Proceedings of the 2006

Australasian Language Technology Workshop

Sancta Sophia College, Sydney
Nov 30-Dec 1, 2006

Sponsored by
HCSNet ALTA

Editors:
Lawrence Cavedon
Ingrid Zukerman
http://www.alta.asn.au/events/altw2006/
ISBN: 1-74108-146-7

Preface

This volume contains the papers accepted for presentation at the Australasian Language Technology Workshop (ALTW) 2006, held at the University of Sydney, Sydney, Australia, on November 30 – December 1, 2006. This is the fourth annual installment of the workshop in its most-recent incarnation, and the continuation of an annual workshop series that has existed under various guises since the early 90s.

The goals of the workshop are:

- to bring together the growing Language Technology (LT) community in Australia and New Zealand and encourage interactions;
- to foster interaction between academic and industrial researchers;
- to encourage dissemination of research results for new and ongoing projects; and
- to increase the visibility of LT research in Australia and New Zealand.

This year's Australasian Language Technology Workshop includes regular talks as well as poster presentations and student posters. Of the 36 papers submitted, 19 papers were selected by the program committee for publication and appear in these proceedings. Of these, 13 are oral presentations papers and 6 are poster presentations. Additionally, we have included 6 student posters to encourage feedback on early results. 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.

We would like to thank all the authors who submitted papers, as well as the members of the program committee for the time and effort they contributed in reviewing the papers. Our thanks also go to local organizer Rolf Schwitter, to Jiawen Rong for installing and configuring the submission management system, and to members of the ALTA executive for their support in organizing the workshop.

This year, ALTW is held as part of the HCSNet (ARC Network in Human Communication Science) SummerFest. We thank HCSNet for its invaluable financial and organisational support, including organising the venue and refreshments, printing the proceedings, handling registration, and particularly for providing travel funds to all presenters, including the invited international speaker: Michael Johnston (AT&T Labs Research). A special thanks goes to Kym Buckley of HCSNet for her immense effort in managing the organisation of the HCSNet events, including ALTW.

Lawrence Cavedon and Ingrid Zukerman
Program Chairs

Committees

Program Chairs:

Lawrence Cavedon (National ICT Australia and RMIT University, Australia)
Ingrid Zukerman (Monash University)

Local Chair:

Rolf Schwitter (Macquarie University)

Program Committee:

Our thanks to the following Program Committee members for their efforts in the reviewing process.

Jan Alexandersson (DFKI)
Timothy Baldwin (University of Melbourne)
Steven Bird (University of Melbourne)
Frances Bond (NTT Communication Science Labs)
Stephen Clark (Oxford University)
Nathalie Colineau (CSIRO ICT Centre)
James Curran (University of Sydney)
Robert Dale (Macquarie University)
Mark Dras (Macquarie University)
Mark Ellison (Analith)
Dominique Estival (Appen)
Tanja Gaustad (Appen)
Graeme Hirst (University of Toronto)
Baden Hughes (University of Melbourne)
Michael Johnston (ATT Labs Research)
Arne Jonsson (Linkoping University)
Dan Jurafsky (Stanford University)
Min-Yen Kan (National University of Singapore)
Alistair Knott (University of Otago)
Kazunori Komatani (Kyoto University)
Kathleen McCoy (University of Delaware)
Chris Manning (Stanford University)
Yuval Marom (Monash University)
Diego Molla (Macquarie University)
Ng Hwee Tou (National University of Singapore)
Cecile Paris (CSIRO ICT Centre)
Matthew Purver (Stanford University)
Tony Smith (Waikato University)
Nicola Stokes (NICTA Victoria)
Seyed M.M. Tahaghoghi (RMIT University)
Takaaki Tanaka (NTT Communication Science Labs)
Stephen Wan (Macquarie University and CSIRO)
Menno van Zaanen (Macquarie University)

Table of Contents

Invited Presentations

Full Papers

Student Posters

Thursday November 30 2006

- **9.00am – 10.45am: Session 1**
  - **Welcome and Opening**
  - Evidence for gradient salience: What happens with competing non-salient referents during pronoun resolution?, *Ralph L. Rose*
  - Error correction using utterance disambiguation techniques, *Peter Vlugter, Edwin van der Ham, and Alistair Knott*
  - Classifying speech acts using verbal response modes, *Andrew Lampert, Robert Dale, and Cecile Paris*
- **10.45am – 11.15am: Coffee break**
- **11.15am – 12.45pm: Session 2**
  - Verb sense disambiguation using selectional preferences extracted with a state-of-the-art semantic role labeler, *Patrick Ye and Timothy Baldwin*
  - Word relatives in context for word sense disambiguation, *David Martinez, Eneko Agirre, and Xinglong Wang*
  - Die morphologie (f): targeted lexical acquisition for languages other than English, *Jeremy Nicholson, Timothy Baldwin, and Phil Blunsom*
- **12.45pm – 2.00pm: Lunch (provided by HCSNet)**
- **2.00pm – 3.30pm: Poster Session**

  Accepted papers
  - Improved default sense selection for word sense disambiguation, *Tobias Hawker and Matthew Honnibal*
  - Automatic mapping clinical notes to medical terminologies, *Jon Patrick, Yefeng Wang and Peter Budd*
  - Pseudo relevance feedback using named entities for Question Answering, *Luiz Augusto Pizzato, Diego Molla, and Cecile Paris*
  - Web readability and Computer-Assisted Language Learning, *Alexandra L. Uitdenbogerd*
  - Questions require an answer: A deductive perspective on questions and answers, *Willemijn Vermaat*
  - Towards the evaluation of referring expression generation, *Jette Viethen and Robert Dale*

  Student posters
  - Analysis and prediction of user behaviour in a museum environment, *Karl Grieser, Tim Baldwin and Steven Bird*
  - Using dialogue acts to suggest responses in support services via Instant Messaging, *Edward Ivanovic*
  - Probabilities improve stress-prediction in a CFG of Hawaiian phonology, *'Oiwi Parker Jones*
  - Towards cognitive optimisation of a search engine interface, *Kenneth Treharne, Darius Pfitzner and David M. W. Powers*
  - Natural language processing and XML retrieval, *Alan Woodley, Xavier Tannier, Marcus Hassler, and Shlomo Geva*
  - Extracting patient clinical profiles from case reports, *Yitao Zhang and Jon Patrick*

- **3.30pm – 4.00pm: Coffee break**
- **4.00pm – 5.30pm: Session 4**
  - Efficient combinatory categorial grammar parsing, *Bojan Djordjevic and James R. Curran*
  - **HCSNet Keynote Presentation**:
    Robust multimodal understanding for interactive systems
    Michael Johnston (AT&T Labs Research)
- **5.30pm—6.00pm: Drinks (courtesy of HCSNet)**
- **6.00pm: Musical Recital (courtesy of HCSNet)**

Friday December 1 2006

- **9.00am—10.00am**:
  - **HCSNet Keynote Presentation**:
    Human-computer interaction based on acoustic signals, muscle movements, and brainwaves, *Tanja Schultz (Carnegie Mellon University)*
- **10.15am—10.45am: Session 5**
  - Computational semantics in the *Natural Language Toolkit*, *Ewan Klein*
- **10.45am—11.15am: Coffee break**
- **11.45am – 12.45pm: Session 6**
  - Experiments with sentence classification, *Anthony Khoo, Yuval Marom and David Albrecht*
  - **Invited Presentation**: User modeling for language technologists
    Judy Kay (University of Sydney)
- **12.45pm – 2.00pm: Lunch (provided by HCSNet)**
- **2.00pm – 3.30pm: Session 7**
  - Named entity recognition for Question Answering, *Diego Molla, Menno van Zaanen, and Daniel Smith*
  - Named entity recognition for astronomy literature, *Tara Murphy, Tara McIntosh, and James R. Curran*
  - This phrase-based SMT system is out of order: Generalised word reordering in machine translation, *Simon Zwarts and Mark Dras*
- **3.30pm – 4.00pm: Coffee break**
- **4.00pm – 4.45pm: Session 8**
  - Using dependency-based features to take the "para-farce" out of paraphrase, *Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris*
  - **Closing Remarks**
- **4.45pm—5.45pm: ALTA Annual General Meeting**

Robust Multimodal Understanding for Interactive Systems

Michael Johnston
AT&T Labs Research

The ongoing convergence of the web with telephony, driven by technologies such as voice over IP, high-speed mobile data networks, and hand-held computers and smartphones, enables widespread deployment of multimodal interfaces which combine graphical user interfaces with natural human input modalities such as speech and pen. In order to support effective multimodal interaction, natural language processing techniques, which have typically been applied to linear sequences of speech or text, need to be extended to support integration and understanding of multimodal language distributed over multiple different simultaneous input modes.

Multimodal grammars (Johnston and Bangalore 2000) combine speech and gesture parsing, integration, and understanding all within a single formalism. Their finite-state implementation enables efficient processing of lattice input from speech and gesture recognition and mutual compensation for errors and ambiguities. However, like other approaches based on hand-crafted rules, multimodal grammars can be brittle with respect to unexpected, erroneous, or disfluent input.

In this talk, I will illustrate and evaluate the use of multimodal grammars to support spoken input combined with complex freehand pen input in the context of a multimodal conversational system, and explore a range of methods for improving their robustness. These include techniques for building effective language models for speech recognition when little or no training data is available and techniques for robust multimodal understanding that draw on classification, machine translation, and sequence edit methods.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 1–1.

User Modelling for Language Technologists

Judy Kay
School of Information Technologies
University of Sydney

This talk overviews some of the potential roles that user modelling can play in improving effective communication between people and machines, with a focus on language technology. Essentially, these relate to
improving the machine's ability to understand a person as well as improved communication from the machine to the person. The talk will present examples of the use of methods of language technology for personalisation and outline some of the barriers to greater user modelling in language-based interfaces.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 2–2.

Efficient Combinatory Categorial Grammar Parsing

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

Abstract

Efficient wide-coverage parsing is integral to large-scale NLP applications. Unfortunately, parsers for linguistically motivated formalisms, e.g. HPSG and TAG, are often too inefficient for these applications.

This paper describes two modifications to the standard CKY chart parsing algorithm used in the Clark and Curran (2006) Combinatory Categorial Grammar (CCG) parser. The first modification extends the tight integration of the supertagger and parser, so that *individual* supertags can be added to the chart, which is then repaired rather than rebuilt. The second modification adds constraints to the chart that restrict which constituents can combine.

Parsing speed is improved by 30–35% without a significant accuracy penalty and a small increase in coverage when both of these modifications are used.

1 Introduction

Parsing is the process of determining the syntactic structure of a sentence. It is an integral part of the deep semantic analysis that any sophisticated Natural Language Processing (NLP) system, such as Question Answering and Information Extraction systems, must perform.

The sentences Bob killed Alice, Alice was killed by Bob and Bob was the man who killed Alice convey the same information. If we treat the sentence as a *bag* or *sequence of words* by assuming limited structure, the sentences appear to be very different. These examples demonstrate that full parsing is necessary for accurate semantic interpretation. Further, sophisticated linguistic analysis capable of modelling a wider range of phenomena should give us the most information.

Unfortunately, parsing is very inefficient because of the large degree of ambiguity present in natural language. This is particularly true for wide-coverage grammars in linguistically expressive formalisms, especially those automatically extracted from a treebank.

Many NLP systems use shallow parsing because full parsing is too slow (Grishman, 1997). To improve the approximate structure identified by shallow parsers, many systems use domain-specific knowledge to extract dependencies (Grishman, 1997; Cole et al., 1997). Ciravegna et al. (1997) show that the accuracy can be improved by using a better parser. The ability of NLP systems to extract useful and correct information could therefore be improved substantially if the speed of full parsing was acceptable.

The C&C CCG parser (Clark and Curran, 2006) is the fastest linguistically motivated parser in the literature, but it is still limited to about 25 sentences per second on commodity hardware.

This paper describes two modifications to the C&C parser that significantly improve parsing efficiency without reducing accuracy or coverage. The first involves *chart repair*, where the CKY chart is repaired when new categories are added, instead of rebuilt from scratch. This allows an even tighter integration of the parser and supertagger (described below) which results in an 11% speed improvement over the original parser.

The second modification involves parsing with *constraints*, that is, requiring certain spans to be constituents. This reduces the search space considerably by eliminating a large number of constituents that cross the boundary of these spans. The best set of constraints results in a 10% improvement over the original parser. These constraints are also useful for other tasks. Finally, when both chart repair and constraints are used, a 30-35% speed improvement is achieved while coverage increases and the accuracy is unchanged.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 3–10.

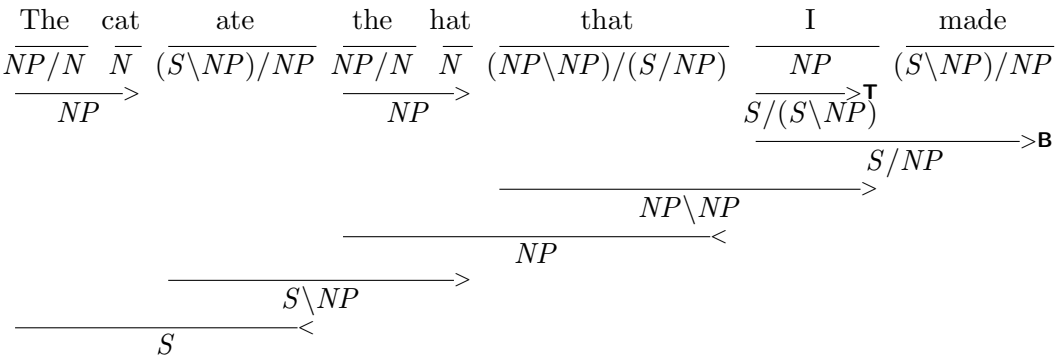

Figure 1: A Combinatory Categorial Grammar derivation

2 Combinatory Categorial Grammar

Context free grammars (CFGs) have traditionally been used for parsing natural language. However, some constructs in natural language require more expressive grammars. *Mildly context sensitive* grammars, e.g. HPSG and TAG, are powerful enough to describe natural language but like CFGs (Younger, 1967) are polynomial time parseable (Vijay-Shanker and Weir, 1990).

Combinatory Categorial Grammar (CCG) is another mildy context sensitive grammar (Steedman, 2000) that has significant advantages over CFGs, especially for analysing constructions involving coordination and long range dependencies. Consider the following sentence:

Give a teacher an apple and a policeman a flower.

There are local dependencies between give and teacher, and between give and apple. The additional dependencies between give, and policeman and flower are long range, but are extracted easily using CCG. a policeman a flower is a non-standard constituent that CCG deals with very elegantly, allowing a teacher an apple and a policeman a flower to be coordinated before attachment to the verb give.

In CCG each word is assigned a category which encodes sub-categorisation information. Categories are either *atomic*, such as N, NP and S for noun, noun phrase or sentence; or *complex*, such as NP/N for a word that combines with a noun on the right to form a noun phrase. $NP\backslash N$ is similarly a word that combines with a noun on the left to create an NP. The categories are then combined using combinatory rules such as *forward application* $>$ $(X/Y\ Y \Rightarrow X)$ and *forward composition* $>\mathbf{B}$ $(X/Y\ Y/Z \Rightarrow_{\mathbf{B}} X/Z)$.

An example derivation that uses a number of combination rules is shown in Figure 1. The example demonstrates how CCG handles long range dependencies such as the hat ... I made. Type raising ($>\mathbf{T}$) and forward composition ($>\mathbf{B}$) on I made are used to derive the same predicate-argument structure as if it was written as I made the hat.

A derivation creates predicate-argument dependencies, which are 5-tuples $\langle h_f, f, s, h_a, l\rangle$, where h_f is the word of the category representing the relationship, f is the category itself, s is the argument slot, h_a is the head word of the argument and l indicates if the dependency is local.

The argument slot is used to identify the different arguments of words like bet in $[\text{Alice}]_1$ bet $[\text{Bob}]_2$ $[\text{five dollars}]_3$ $[\text{that they win}]_4$. The dependency between bet and Bob would be represented as $\langle \text{bet}, (((S\backslash NP_1)/NP_2)/NP_3)/S[em]_4, 2, \text{Bob}, -\rangle$. The C&C parser is evaluated by comparing extracted dependencies against the gold standard. Derivations are not compared directly because different derivations can produce the same dependency structure.

The parser is trained on CCGbank, a version of Penn Treebank translated semi-automatically into CCG derivations and predicate-argument dependencies (Hockenmaier and Steedman, 2006). The resulting corpus contains 99.4% of the sentences in the Penn Treebank. Hockenmaier and Steedman also describe how a large CCG grammar can be extracted from CCGbank. A grammar automatically extracted from CCGbank is used in the C&C parser and supertagger.

3 C&C CCG Parser

The C&C parser takes one or more syntactic structures (categories) assigned to each word and attempts to build a spanning analysis of the sentence. Typically every category that a word was seen with in the training data is assigned.

Supertagging (Bangalore and Joshi, 1999) was introduced for Lexicalized Tree Adjoining Grammar (LTAG) as a way of assigning fewer categories to each word thus reducing the search space of the parser and improving parser efficiency.

Clark (2002) introduced supertagging for CCG parsing. The supertagger used in the C&C parser is a maximum entropy (Berger et al., 1996) sequence tagger that uses words and part of speech (POS) tags in a five word window as features. The label set consists of approximately 500 categories (or supertags) so the task is significantly harder than other NLP sequence tagging tasks.

The supertagger assigns one or more possible categories to each word together with the probability for each of the guesses. Clark and Curran (2004) discovered that initially assigning a very small number of categories per word and then attempting to parse was not only faster but more accurate than assigning many categories. If no spanning analysis could be found the parser requested more categories from the supertagger, and the parsing process was repeated until the number of attempts exceeded a limit (typically 5 levels of supertagger ambiguity). This tight integration of the supertagger and the parser resulted in state of the art accuracy and a massive improvement in efficiency, reaching up to 25 sentences a second.

Up until now, when a spanning analysis was not found the chart was destroyed, then extra categories are assigned to each word, and the chart is built again from scratch. However the chart rebuilding process is very wasteful because the new chart is always a superset of the previous one and could be created by just updating the old chart instead of rebuilding it.

This has limited how small the initial ambiguity levels can be set and thus how closely the parser and supertagger can interact. The first modification we describe below is to implement *chart repair* which allows additional categories to be assigned to an existing chart and the CKY algorithm to run efficiently over just the modified section.

4 Chart Parsing

Given a sentence of n words, we define position $pos \in \{0, \ldots, n-1\}$ to be the starting position of a span (contiguous sequence of words), and $span$, its size. So the hat in the cat ate the hat would have $pos = 3$ and $span = 2$. Each span can be parsed in a number of ways so a set of derivations will be created for each valid $(pos, span)$ pair. Let $(pos, span)$ represent this set of derivations. Then, the derivations for $(pos, span)$ will be combinations of derivations in (pos, k) and $(pos+k, span-k)$ for all $k \in \{1, \ldots, span-1\}$. The naïve way to parse a sentence using these definitions is to find the derivations that span the whole sentence $(0, n)$ by recursively finding derivations in $(0, k)$ and $(k, n-k)$ for all $k \in \{1, \ldots, n-1\}$. However, this evaluates derivations for each $(pos, span)$ pair multiple times, making the time complexity exponential in n.

To make the algorithm polynomial time, dynamic programming can be used by storing the derivations for each $(pos, span)$ when they are evaluated, and then reusing the stored values. The *chart* data structure is used to store the derivations. The chart is a two dimensional array indexed by pos and $span$. The valid pairs correspond to $pos + span \leq n$, that is, to spans that do not extend beyond the end of the sentence. The squares represent valid cells in Figure 2. The location of $cell(3, 4)$ is marked with a diamond. $cell(3, 4)$ stores the derivations whose yield is the four word sequence indicated.

The CKY (also called CYK or Cocke-Younger-Kasami) algorithm used in the C&C parser has an $O(n^3)$ worst case time complexity. Sikkel and Nijholt (1997) give a formal description of CKY (Younger, 1967) and similar parsing algorithms, such as the Earley parser (Earley, 1970).

The CKY algorithm is a bottom up algorithm and works by combining adjacent words to give a span of size two (second row from the bottom in Figure 2). It then combines adjacent spans in the first two rows to create all allowable spans of size three in the row above. This process is continued until a phrase that spans the whole sentence (top row) is reached.

The (lexical) categories in the bottom row (on the lexical items themselves) are assigned by the supertagger (Clark and Curran, 2004). The number of categories assigned to each word can be varied dynamically. Assigning a small number of categories (i.e. keeping the level of lexical category ambiguity low) increases the parsing speed significantly but does not always produce a spanning derivation. The original C&C parser uses a small number of categories first, and if no spanning tree is found the process is repeated with a larger number of categories.

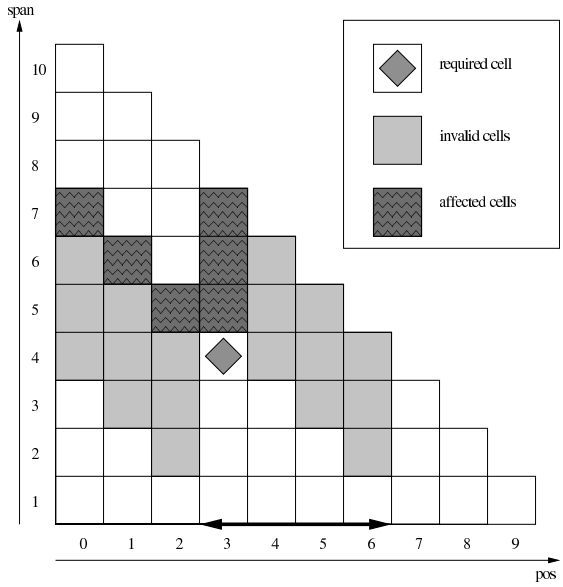

Figure 2: Cells affected by adding a constraint. The axes are the cell indices in the chart with **pos** the starting position, and **span** the length of the span, of constituents in the cell.

5 Constraints

The original C&C parser uses a supertagger to assign a number of categories to each word, together with the probability of each category (Clark and Curran, 2004). In the first iteration, only categories with $\beta \geq 0.075$ are used, where β is the ratio of the probability of the category and the probability of the most likely category for that word. For example, if the categories for dog are N, NP and N/N with probabilities 0.8, 0.12 and 0.08 then β value of N/N is $0.08/0.8 = 0.1$. If there is no spanning tree, a lower value is used for the cutoff β in the next iteration so that more tags are assigned to each word. The previous chart is destroyed and the new chart is built from scratch. Since the assigned categories are always a superset of the previously assigned ones, the derivations in the new chart will include all the derivations in the previous chart.

Instead of rebuilding the chart when new categories are added it can be simply repaired by modifying cells that are affected by the new tags. Considering the case where a single tag is added to the ith word in an n word sentence, the new tag can only affect the cells that satisfy $pos \leq i$ and $pos + span > i$. These cells are shown in Figure 3. The chart can therefore be repaired bottom up by updating a third of the cells on average.

The number of affected cells is $(n - pos) \times pos$ and the total number of cells is approximately $\frac{n^2}{2}$. The average number of affected cells is approximately $\frac{1}{n}\int_0^n (n-p)p \; dp = \frac{n^2}{6}$, so on average a third of the cells are affected.

The chart is repaired bottom up. A new category is added to one word by adding it to the list of categories in the appropriate cell in the bottom row. The list is marked so that we know which categories are new. For each cell C in the second row we look for each pair of cells A and B whose spans combine to create the span of C. In the original algorithm all categories from A are combined with all categories from B, but during the repair this is only done if at least one of them is new because otherwise the resulting category would already be in C. Again the list of categories in C is marked so that cells higher in the chart know which categories are new. This is repeated for all affected cells.

This speeds up the parser not only because previous computations are reused, but also because categories can be added one at a time until a spanning derivation is found. This increases coverage slightly because the number of categories can be varied one by one. In the original parser it was possible to have sentences that a spanning tree cannot be found for using for example 20 categories, but increasing the number of categories to 25 causes the total number of derivations in the chart to exceed a predefined limit, so the sentence does not get parsed even if 23 categories would produce a spanning tree without exceeding the limit.

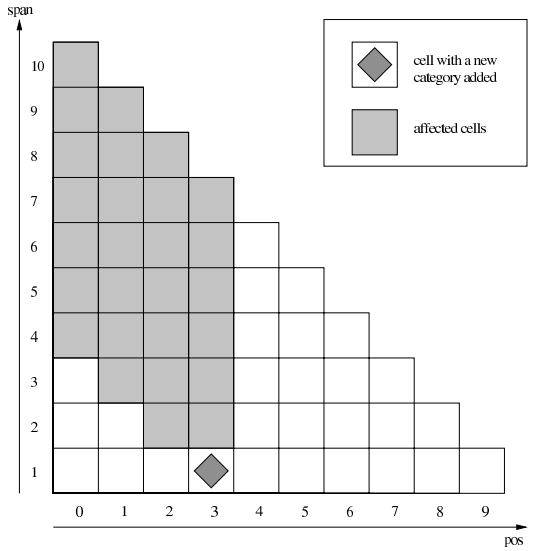

Figure 3: Cells affected by chart repair.

6 Chart Repair

Adding constraints to the parser was originally designed to enable more efficient manual annotation. The parser has been used for semi-automatic parsing of sentences to create gold standard derivations. It is used to produce a guess, which is then manually corrected. Adding a constraint could speed this process up by allowing annotators to quickly tell the parser which part it got wrong.

For example if the Royal Air Force contract was parsed as (the Royal (Air Force contract)), meaning that the Air Force contract was royal, instead of manually annotating the category for each word the annotator could simply create a constraint on Royal Air Force requiring it to be a constituent, and thus making the previous derivation impossible. The correct derivation, (the ((Royal Air Force) contract)) would then very likely be produced without further effort from the annotator.

However, parsing would be much faster in general if the search space could be constrained by requiring *known* spans P to be a single constituent. This reduces the search space because P must be the yield of a single cell $C_P(pos_P, span_P)$, so the cells with yields that cross the boundary of P do not need to be considered at all (grey squares in Figure 2). The problem of what spans are known in advance is described below.

In addition, if a cell contains P as a *prefix* or *suffix* (wavy pattern cells in Figure 2) then it also has constraints on how it can be created. In Figure 2, $P = cell(3, 4)$ is required, i.e. the span starting at word 3 of length 4 containing words 3, 4, 5 and 6 is a constituent. Consider $cell(3, 7)$. It includes words 3 to 9 and contains P as the prefix. Normally $cell(3, 7)$ can be created by combining $cell(3, 1)$ with $cell(4, 6)$, ..., $cell(pos, s)$ with $cell(pos + s, span - s)$, ..., and $cell(3, 6)$ with $cell(9, 1)$. However the first three of these combinations are not allowed because the second component would cross the boundary of P. This gives a lower limit for the span of the left component. Similarly if P is the suffix of the span of a cell then there is a lower limit on the span of the right component.

This eliminates a lot of work during parsing and can provide a significant speed increase. In the quick brown foxes like dogs, the following phrases would all be possible constituents: foxes like dogs, brown foxes like dogs, ..., and the quick brown foxes like dogs. Since the chart is built bottom up the parser has no knowledge of the surrounding words so each of those appears like a valid constituent and would need to be created. However if the quick brown foxes is known to be a constituent then only the last option becomes possible.

7 Creating Constraints

How can we know that specific spans must be yielded by constituents in advance? Surely the parsing is already solved if we have this information? In this paper, we have experimented with constraints determined from shallow parsing and hints in the sentence itself.

Chunk tags (gold standard and from the C&C chunker) were used to create constraints. Only NPs

were used because the accuracy for other chunks is low. The chunks required modification because the Penn Treebank has different analyses to CCGbank, e.g. Larry's dog is chunked as [Larry]$_{NP}$ ['s dog]$_{NP}$, which is different to the CCGbank analysis. Adjacent NPs of this form are concatenated.

A number of *punctuation constraints* were used and had a significant impact especially for longer sentences. The punctuation rules in CCGbank are very productive. For example the final punctuation in The dog ate a bone. will create all of these constituents: bone., a bone., ..., The dog ate a bone. However, in CCGbank the sentence final punctuation is always attached at the root. A constraint on the the first $n-1$ words was added to force the parser to only attach the sentence final punctuation once the rest of the sentence has been parsed.

Constraints are also placed on parenthesised expressions. In The dog (a hungry Pomeranian) ate a bone, the phrase a hungry Pomeranian clearly needs to be parsed first and then attached to the rest of the sentence. However, CCGbank would allow the right parenthesis to be absorbed by Pomeranian before hungry is attached. The same would apply to quoted expressions but CCGbank has removed the quotation marks. However, the parser must still deal with quoted expressions in practical systems.

Finally constraints are placed on phrases bounded by semicolons, colons and hyphens. This is especially useful with longer sentences of the form Alice paid $5; Bob paid $6; ..., where many clauses are separated by semicolons. This reduces the sentence to a number of smaller units which significantly improves parsing efficiency.

If the parser cannot parse the sentence with constraints then they are removed and the sentence is parsed again. This increases the coverage because the reduced search space means that longer sentences can be parsed without exceeding the memory or time limit. However if the constraint accuracy is low a large number of sentences will need to be parsed twice which would cancel out anything gained from using them.

8 Experiments

The parser is trained on CCGbank sections 02-21, with section 00 being used for development. The performance is measured in terms of coverage, accuracy and parsing time. The time reported includes loading the grammar and statistical model, which is $\sim$5 seconds, and parsing the 1913 sentences in section 00. Accuracy is measured in terms of the dependency F-score.

The failure rate (the opposite of coverage) is broken down into sentences with length up to 40 and with length over 40 because the longer sentences are the most problematic ones and the original parser already has high coverage on sentences with up to 40 words. There are 1784 1-40 word sentences and 129 41+ word sentences. The average length and standard deviation in 41+ are 50.8 and 31.5 respectively.

All experiments used gold standard POS tags. Some experiments use gold standard chunks to determine an upper bound on the utility of chunk constraints. *Original* and *Original+repair* do not use any constraints. *NP(gold)* indicates that gold standard noun phrase constraints are used. *NP* uses the C&C chunker, and *punctuation* adds punctuation constraints. The times reported for *NP* (using the C&C chunker) include the time to load the chunker model ($\sim$1.3 seconds).

Finally the best performing system was compared against the original on section 23, which has 2257 sentences of length 1-40 and 153 of length 41+. The maximum sentence length is only 65, which explains the high coverage for the 41+ section.

9 Results

The results in Table 1 show that using gold standard noun phrases does not improve efficiency, while using noun phrases identified by the chunker decreases speed by 10.8%. This is not surprising because the chunk data was not obtained from CCGbank and the chunker is not very accurate. Some frequent problems were fixed in a preprocessing step as explained in Section 5, but there could be less frequent constructions that cause problems. A more detailed analysis of these constructions is required.

Chart repair (without constraints) gave an 11.1% improvement in speed and 0.21% improvement in accuracy. The accuracy was improved because of the way the repair process adds new categories. Categories are added in decreasing order of probability and parsing stops once a spanning tree is found. This effectively allows the parser to use the probabilities which the supertagger assigns, which are not directly modelled in the parser. Once supertagger probabilities are added to the parser statistical model there should be no

	TIME		ACC	COVER	FAIL RATE %	
	secs	%	%	%	$n \leq 40$	$n > 40$
Original	88.3	—	86.54	98.85	0.392	11.63
punctuation	79.1	10.4	86.56	99.22	**0.168**	9.30
NP(gold)	88.4	-0.1	86.27	99.06	0.224	10.85
NP	97.8	-10.8	86.31	99.16	0.224	9.30
NP(gold) + punctuation	69.8	20.5	86.24	99.27	**0.168**	8.53
NP + punctuation	97.0	-9.9	86.31	99.16	**0.168**	10.08
Original + repair	78.5	11.1	**86.75**	99.01	0.336	10.08
NP(gold) + repair	65.0	26.4	86.04	99.37	0.224	6.20
NP + repair	77.5	12.2	86.35	99.37	0.224	6.20
punctuation + repair	57.2	35.2	86.61	99.48	**0.168**	5.43
NP(gold) + punctuation + repair	**48.2**	**45.4**	86.14	99.48	**0.168**	5.43
NP + punctuation + repair	63.2	28.4	86.43	**99.53**	0.163	**3.88**

Table 1: Parsing performance on section 00 with constraints and chart repair.

accuracy difference between the original method and chart repair.

The best results for parsing with constraints (without repair) were with gold standard noun phrase and punctuation constraints, with 20.5% improvement in speed and 0.42% in coverage. In that case, however the accuracy decreases by 0.3% which is again possibly because CCG constituents do not match up with the chunks every time. The best results obtained without a decrease in accuracy is using only punctuation constraints, with 10.4% increase in speed and 0.37% in coverage.

The best overall result was obtained when gold standard noun phrase and punctuation constraints were used with chart repair, with a 45.4% improvement in speed and 0.63% in coverage, and a 0.4% drop in accuracy. Again the best results without a drop in accuracy were with only punctuation constraints and chart repair, with improvements of 35.2% and 0.63%.

The results also show that coverage of both short and long sentences is improved using these methods. For example the best results show a 43% and 67% decrease in failure rate for sentence lengths in the ranges 1-40 and 41+.

Comparing the last three rows allows us to guess how accurate the chunker will need to be to achieve a faster speed than just using punctuation constraints. Noun phrases clearly have an impact on speed because using gold standard chunks gives a significant improvement, however the C&C chunker is currently not accurate enough. The chunker would need to have about half the error rate it currently has in order to be useful.

Table 2 shows the performance of the punctuation constraints and chart repair system on section 23. The results are consistent with previous results, showing a 30.9% improvement in speed and 0.29% in coverage, with accuracy staying at roughly the same level.

10 Future Work

A detailed analysis of where NPs chunks do not match the CCG constituents is required if NPs are to be used as constraints. The results show that NPs can provide a large improvement in efficiency if identified with sufficient precision.

The chart repair has allowed an even greater level of integration of the supertagger and parser. We intend to explore strategies for determining which category to add next if a parse fails.

Constraints and chart repair both manipulate the chart for more efficient parsing. Other methods of chart manipulation for pruning the search space will be investigated. Agenda based parsing, in particular A* parsing (Klein and Manning, 2003), will be implemented in the C&C parser, which will allow only the most probable parts of the chart to be built, improving efficiency while guaranteeing the optimal derivation is found.

11 Conclusion

We have introduced two modifications to CKY parsing for CCG that significantly increase parsing efficiency without an accuracy or coverage penalty.

Chart repair improves efficiency by reusing the partial CKY chart from the previous parse at-

	TIME		ACC	COVER	FAIL RATE %	
	secs	%	%	%	$n \leq 40$	$n > 40$
Original	91.9	—	**86.92**	99.29	0.621	1.961
punctuation + repair	**63.5**	**30.9**	86.89	**99.58**	**0.399**	**0.654**

Table 2: Parsing performance on Section 23 with constraints and chart repair.

tempts. This allows us to further exploit the tight integration of the supertagger and parser by adding one lexical category at a time until a parse of the sentence is found. Chart repair alone gives an 11% improvement in speed.

Constraints improve efficiency by avoiding the construction of sub-derivations that will not be used. They have a significant impact on parsing speed and coverage without reducing the accuracy, provided the constraints are identified with sufficient precision.

When both methods are used the speed increases by 30-35%, the failure rate decreases by 40-65%, both for sentences of length 1-40 and 41+, while the accuracy is not decreased. The result is an even faster state-of-the-art wide-coverage CCG parser.

12 Acknowledgments

We would like to thank the anonymous reviewers for their feedback. This research was funded under Australian Research Council Discovery grants DP0453131 and DP0665973.

References

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Fabio Ciravegna, Alberto Lavelli, and Giorgio Satta. 1997. Efficient full parsing for Information Extraction. In *Proceedings of the Meeting of the Working Groups on Automatic Learning and Natural Language of the Associazione Italiana per l'Intelligenza Artificale (AI*IA)*, Turin, Italy.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *20th International Conference on Computational Linguistics*, pages 282–288, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2006. Wide-coverage statistical parsing with CCG and log-linear models. (submitted).

Stephen Clark. 2002. A supertagger for Combinatory Categorial Grammar. In *Proceedings of the TAG+ Workshop*, pages 19–24, Venice, Italy.

Ronald Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue. 1997. *Survey of the state of the art in human language technology*. Cambridge University Press, New York, NY, USA.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Ralph Grishman. 1997. Information Extraction: Techniques and challenges. In *SCIE '97: International Summer School on Information Extraction*, pages 10–27, London, UK. Springer-Verlag.

Julia Hockenmaier and Mark Steedman. 2006. CCGbank - a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. (submitted).

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of Human Language Technology and the North American Chapter of the Association for Computational Linguistics Conference*, pages 119–126, Edmond, Canada.

Klass Sikkel and Anton Nijholt. 1997. Parsing of Context-Free languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 2: Linear Modelling: Background and Application*, pages 61–100. Springer-Verlag, New York.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

K. Vijay-Shanker and David J. Weir. 1990. Polynomial time parsing of combinatory categorial grammars. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, pages 1–8, Pittsburgh, Pennsylvania.

Daniel H. Younger. 1967. Recognition and parsing of contex-free languages in time n^3. *Information and Control*, 10(2):189–208.

Improved Default Sense Selection for Word Sense Disambiguation

Tobias HAWKER and **Matthew HONNIBAL**
Language Technology Research Group
School of Information Technologies
University of Sydney
{toby,mhonn}@it.usyd.edu.au

Abstract

Supervised word sense disambiguation has proven incredibly difficult. Despite significant effort, there has been little success at using contextual features to accurately assign the sense of a word. Instead, few systems are able to outperform the default sense baseline of selecting the highest ranked WordNet sense. In this paper, we suggest that the situation is even worse than it might first appear: the highest ranked WordNet sense is not even the best default sense classifier. We evaluate several default sense heuristics, using supersenses and SemCor frequencies to achieve significant improvements on the WordNet ranking strategy.

1 Introduction

Word sense disambiguation is the task of selecting the sense of a word intended in a usage context. This task has proven incredibly difficult: in the SENSEVAL 3 all words task, only five of the entered systems were able to use the contextual information to select word senses more accurately than simply selecting the sense listed as most likely in WordNet (Fellbaum, 1998). Successful WSD systems mostly fall back to the first-sense strategy unless the system was very confident in over-ruling it (Hoste et al., 2001).

Deciding which sense of a word is most likely, irrespective of its context, is therefore a crucial task for word sense disambiguation. The decision is complicated by the high cost (Chklovski and Mihalcea, 2002) and low inter-annotator agreement (Snyder and Palmer, 2004) of sense-tagged corpora. The high cost means that the corpora are small, and most words will have only a few examples. The low inter-annotator agreement exacerbates this problem, as the already small samples are thus also somewhat noisy. These difficulties mean that different sense frequency heuristics can significantly change the performance of a 'baseline' system. In this paper we discuss several such heuristics, and find that most outperform the commonly used first-sense strategy, one by as much as 1.3%.

The sense ranks in WordNet are derived from semantic concordance texts used in the construction of the database. Most senses have explicit counts listed in the database, although sometimes the counts will be reported as 0. In these cases, the senses are presumably ranked by the lexicographer's intuition. Usually these counts are higher than the frequency of the sense in the SemCor sense-tagged corpus (Miller et al., 1993), although not always. This introduces the first alternative heuristic: using SemCor frequencies where available, and using the WordNet sense ranking when there are no examples of the word in SemCor. We find that this heuristic performs significantly better than the first-sense strategy.

Increasing attention is also being paid to coarse grained word senses, as it is becoming obvious that WordNet senses are too fine grained (Hovy et al., 2006). Kohomban and Lee (2005) explore finding the most general hypernym of the sense being used, as a coarser grained WSD task. Similarly, Ciaramita and Altun (2006) presents a system that uses sequence tagging to assign 'supersenses' — lexical file numbers — to words. Both of these systems compare their performance to a baseline of selecting the coarse grained parent of the first-ranked fine grained sense. Ciaramita and Altun also use this baseline as a feature in their

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 11–17.

model. We explore different estimates of the most frequent super-sense, and find that aggregating the counts of the fine-grained senses is a significantly better heuristic for this task.

We believe that one of the reasons coarse grained senses are useful is that they largely ameliorate the inter-annotator agreement issues of fine grained sense tagging. Not only are there fewer senses to choose from, but the senses are more distinct, and therefore should be less easily confused. The super-sense tags are therefore probably less noisy than the fine-grained senses, which would make corpus-based estimates of their frequency more reliable. The best performing fine grained frequency heuristic we present exploits this property of supersenses, by making the assumption that the most frequent sense of a word will be the most frequent member of the most frequent supersense. Essentially, when the overall most frequent sense of a word is a member of a minority supersense, we find that it is better to avoid selecting that sense. This system scores 63.8% on the SENSEVAL 3 all words task, significantly higher than the relevant baseline of 62.5% — using no contextual information at all.

2 Preliminaries: WordNet Sense Ranks, Frequencies and Supersenses

This paper discusses different methods of selecting a 'default' sense of a word from the WordNet (Fellbaum, 1998) sense inventory. These methods draw on four different sources of information associated with the lexicon: WordNet sense ranks, WordNet sense counts, the SemCor sense tagged corpus, and WordNet lexical file numbers.

Each word entry in WordNet consists of a lemma under a part-of-speech and an inventory of its senses. These senses are ranked by the lexicographers according to their frequencies in "various semantic concordance texts" (Fellbaum, 1998). These frequencies are often given in the database. We refer to them as *WordNet counts* to distinguish them from the frequencies we obtain from the SemCor corpus. The SemCor corpus is a subset of the semantic concordance texts used to calculate WordNet counts.

Each WordNet sense is categorised under one of forty-five lexicographer files. Each lexicographer file covers only one part of speech. The main categorisation is applied to nouns and verbs, as there is only one file for adverbs, and three for adjectives. Lexical files are interesting because they represent broad, or coarse-grained, semantic categories; and therefore a way around the commonly noted problem that WordNet senses are generally too fine grained. We describe a first-sense heuristic that takes advantage of this property of the lexicographer files (often referred to as 'supersenses' (Ciaramita and Johnson, 2003) — we use both terms interchangeably). We also discuss first supersense heuristics, as increasing attention is being paid to supervised supersense tagging (Ciaramita and Altun, 2006).

3 First Order Models for Word Sense Disambiguation

Supervised word sense disambiguation (WSD) is the task of finding the most likely sense s from a sense inventory S given a usage context C:

$$\arg\max_{s \in S} P(s|C) \quad (1)$$

These models are usually compared to models of the form:

$$\arg\max_{s \in S} P(s) \quad (2)$$

in order to evaluate how much the context is informing the model. Since we are primarily interested in the argmax, it is usually sufficient to simply define a function that selects the most likely sense, even if a full probability distribution is not defined. Selecting the sense listed first in WordNet is one such function.

As a WSD system, a first order model has an inherent upper bound, as if a word is used with more than one sense, a first order model cannot get all examples correct. However, Figure 1 shows that on the SENSEVAL 3 data, this upper bound is far higher than the performance of state-of-the-art WSD systems. The upper bound was calculated by selecting the most frequent sense of each word in the *test* data. It is effectively a system with oracle frequency information. Because the correct sense will always be given to words that only occur once, it is interesting to see how the upper bound decays if the system is forced to use the first-sense heuristic instead for words that occur less than n times in the test data. For $n > 14$, the oracle system either falls back to or makes the same prediction as the first-sense system for every instance, and so the systems are effectively identical.

McCarthy et al. (2004) described a first order word sense disambiguation system that ac-

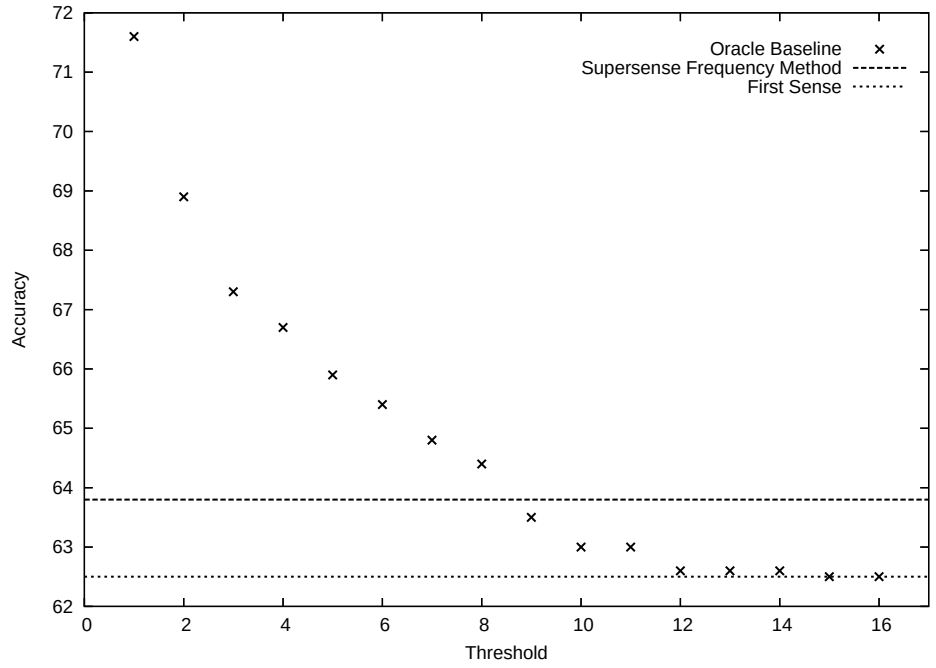

Figure 1: Upper performance bound of one-sense per term strategy for SenseEval

quired 'predominant' senses automatically from un-annotated data using distributional similarity as a proxy for context, and WordNet-based semantic similarity as a sense distinction heuristic. The authors reported an accuracy of 64%, but their system was evaluated on the nouns from the SENSEVAL 2 all words task, and hence cannot be directly compared with the results we report.

4 Experiments

In this section we describe several default sense heuristics and evaluate them on the SENSEVAL 3 English all words test data set. All of the systems use the tokenisation, lemmatisation and part-of-speech tags supplied in the data.

Table 1 presents the results for the systems described below. Each system selects a source of information to select a supersense (None, WordNet, SemCor) and a fine grained sense (WordNet or SemCor). When supersenses are being used, the fine grained sense is chosen from within the selected supersense, effectively filtering out the senses that belong to other lexical files.

4.1 Supersense: None; Fine Sense: WordNet

This is the default sense heuristic used as the baseline in SENSEVAL 3, and is the most common heuristic used for WSD. The heuristic involves simply choosing the lowest numbered sense in the WordNet sense inventory. As this is the only heuristic we explore that does not require extra data, it is the only one that has perfect coverage of WordNet's vocabulary — there is guaranteed to be a sense rank for every WordNet lemma. When there is a coverage problem for one of the other heuristics, such as a case where a word has not occurred in the frequency estimation corpus, that heuristic is allowed to fall back to the WordNet sense rank, rather than being forced to select a sense arbitrarily.

4.2 Supersense: None; Fine Sense: SemCor

As noted in Section 2, SemCor is effectively a subset of the information used to produce the WordNet sense ranks. We evaluated a default sense heuristic that preferred SemCor-only frequncy estimates for words that occurred at least once in the SemCor corpus. This only results in a different prediction from the first-sense heuristic 7% of the time. Nevertheless, the systems perform significantly differently.

4.3 Supersense: WordNet; Fine Sense: WordNet

WordNet sense ranks can also be straightforwardly used as the basis of a default supersense heuristic. Ciaramita and Altun (2006) select the supersense of the first fine grained sense

S.S. Source	Sense Source	S.S. Acc.	WSD Acc.	Δ Coverage	Δ Acc.	Δ Baseline
None	WordNet	79.5	62.5	N/A	N/A	N/A
None	SemCor	80.6	63.4	**7.0%**	36.2	23.2
WordNet	WordNet	79.9	62.3	3.6%	25.3	30.1
WordNet	SemCor	79.9	62.3	3.5%	25.3	**31.0**
SemCor	WordNet	**81.3**	**63.8**	5.9%	**42.3**	19.5
SemCor	SemCor	**81.3**	63.1	5.7%	31.0	20.3

Table 1: Disambiguation Performance for Frequency Estimation Strategies

in WordNet as the baseline system for supersense tagging. A slightly more motivated default supersense heuristic is to aggregate the WordNet counts for each supersense, and select the overall most frequent one. If there are more than two senses, there may be multiple senses after the first that share the same lexicographer file, and together their counts may outweigh the supersense of the first-ranked sense. This situation — having a minority supersense for the first-ranked sense — is quite rare: this heuristic only makes a different prediction from the baseline in 3.6% of cases.

The fine-grained WSD performance of this system is evaluated by choosing the sense with the highest WordNet rank from among the members of the default supersense.

4.4 Supersense: WordNet; Fine Sense: SemCor

Default supersenses in this system are again obtained from the sum of WordNet counts. The sense with the highest count in SemCor from the members of that supersense is then used as the fine-grained sense. The difference from the baseline for this system is even slighter — a different prediction is made in only 3.5% of cases. We would expect this system to have low fine-grained sense accuracy, as the data used to determine the fine-grained sense is effectively a subset of that used for the previous system.

4.5 Supersense: SemCor; Fine Sense: WordNet

The SemCor frequencies can be substituted for WordNet counts to form an alternative supersense heuristic, contrasting with the system described in Section 4.3. The frequency of each supersense is estimated as the sum of the frequencies of its member senses. The supersense with the highest frequency is deemed the default supersense.

In this system, WordNet sense rankings are used to choose a fine-grained sense from among the members of the default supersense as selected from SemCor frequencies.

4.6 SuperSense: SemCor; Fine Sense: SemCor

We also evaluated the fine-grained WSD performance of SemCor-based supersense selection using the counts from SemCor itself.

5 Results

Table 1 gives the WSD and supersense accuracies of the methods outlined in Section 4. Accuracy was calculated with the `scorer2` program provided for evaluation of SENSEVAL 3 systems. The best results for each measure are highlighted in **bold**.

The *S.S. Acc.* column shows the accuracy of supersense predictions as obtained from the supersense of the default sense, over the SENSEVAL 3 test set. The *WSD Acc.* column shows the accuracy at fine-grained WSD. Δ *Coverage* indicates the proportion of content tokens in the test data where the heuristic makes a different prediction from the first-sense baseline. The Δ *Acc.* column shows the accuracy of the strategy on these tokens, while the Δ *Baseline* is the performance of the baseline on these same tokens.

First, it is apparent that the SemCor derived heuristics outperform those calculated from the WordNet counts. This is slightly surprising, as the SemCor frequencies are a subset of the information represented by the WordNet counts, which are used to create the sense rankings. The first sense baseline is also far more widely used, and is the comparison point for SENSEVAL 3 all words systems. The best system at SENSEVAL 3 (Decadt et al., 2004) scored only 2.7% higher than this baseline.

The SemCor strategies 'cover' tokens where the sense distributions in the WordNet counts and

Token Type	SenseEval Counts	SemCor Counts	Newly Wrong	Newly Correct	Net Gain
feel.v	12	207	4	1	-3
state.n	12	184	2	10	8
time.n	11	511	3	3	0
take.v	10	357	0	4	4
policy.n	6	59	0	6	6
thing.n	6	271	1	0	-1
hold.v	5	143	0	0	0
trouble.n	4	52	2	2	0
appear.v	3	152	1	1	0
rate.n	3	108	0	3	3
cloud.n	2	26	2	0	-2
couple.n	2	29	0	2	2
line.n	2	124	0	0	0
suppose.v	2	53	2	0	-2
tremor.n	2	1	0	2	2
hapax legomena	34	927	6	16	10
not in SenseEval	-	5,022	-	-	-
Totals	116 (5.9%)	8,226 (4.4%)	23 (1.1%)	50 (2.4%)	27 (1.3%)

Table 2: Performance Change by Token Type

the SemCor frequencies disagree. The Δ *Baseline* column shows that the first-sense strategy performs very poorly on these tokens. It is unsurprising that these tokens are difficult cases. The fact that a different sense is most frequent in a subset of the WordNet concordance data from the total sample is a good indication that the sense frequencies might be highly domain dependent. It is possible that the SemCor corpus better matches the domains of the SENSEVAL texts, producing more useful sense frequencies for these volatile cases.

No SemCor information is represented in the supersense with highest WordNet count heuristic described in Section 4.3. This heuristic has substantially lower coverage than the SemCor methods, and the baseline performs much higher on the tokens that it does make a prediction for. This supports the interpretation that it is the SemCor frequencies that are the important factor in the improved results.

The highest performance, however, is achieved by calculating the most frequent supersense with the SemCor information, and then using that to exclude senses which belong to a minority lexicographer file. This is statistically significant compared to the baseline (paired t-test, $p < 0.01$), and is only 1.4% lower than Decadt et al. (2004)'s system. The baseline performs particularly poorly on the samples these strategies (described in Section 4.5) cover, suggesting that having the first sense belong to a minority supersense is a good indication that the WordNet sense rank is suboptimal. One of these systems performs significantly better than the other, however (paired t-test, $p < 0.01$). It seems that having identified a volatile example, and a vague concept area the default sense should belong to, it is then best to use all of the available information to choose a sense.

This would explain why the system that uses SemCor counts to choose a supersense and then the WordNet sense-rank to choose a fine grained sense from within it performs the best. This system has the advantage of the SemCor data and the supersense to identify the best subset of volatile examples — the baseline performs at only 19.5% on the examples this system makes a different prediction on, roughly the same number as the other system that uses Semcor supersenses, on which the baseline performs at 20.3%. However, once this subset has been identified, selecting the fine grained sense with the sense rank produces 42% accuracy on the covered tokens, while using the SemCor frequency achieves only 31% Δ *Acc.*.

The performance of the first sense baseline and S.S by SemCor strategies are shown in Figure 1 for comparison with oracle one-sense-per-word accu-

racy.

The difference in correctly assigning senses between the baseline and best-performing systems is statistically significant (paired t-test, $p < 0.01$).

5.1 Token Types

Table 2 shows the behaviour of the best performing system for the tokens it covers. The *hapax legomena* row aggregates scores for content words in this category that occur only once in the test data. The *Newly Wrong* and *Newly Correct* columns refer to the number of instances where the change of default sense has changed from correct to incorrect or vice versa, as compared to the first-sense baseline. The *Net Gain* column indicates the overall contribution from this token type to the performance of the system. The *Not in Senseval* row indicates the tokens where the default sense would be changed, but did not occur in the SENSEVAL test data. Including these tokens in the count allows an accurate comparison of the total coverage of the changed tokens for both corpora.

The table shows that there are both gains and losses for the strategy when compared to the baseline, as should be expected for a classifier limited to assigning only one sense per term. However, the net effect is significantly positive. The table also shows that this positive performance is not simply down to one or two frequent words the heuristic happens to make the right decision on. Almost one third of the newly correct tokens come from decisions made on words that occur only once in the test data. This supports the suggestion above that the heuristic is identifying a range of volatile terms.

6 Conclusions

We have evaluated several heuristics for assigning a default WordNet sense to terms. Our results consistently showed that heuristics which were more sensitive to frequencies in the SemCor corpus outperformed heuristics exclusively based on WordNet sense rankings — suggesting that the stated baselines for SENSEVAL 3 are actually lower than they should be. This is somewhat alarming, considering that systems struggle to make even marginal improvements over the first sense baseline. Since the SemCor data is used to train the supervised systems, the most frequent sense can be inferred — allowing a system to compare favourably with the baseline even if it does not actually gain anything significant from the context of the word.

We have shown that a more nuanced default sense heuristic can achieve some performance gains over simple frequency heuristics, as sense tagged corpora are not large enough to produce entirely reliable fine-grained sense frequencies. By using the frequency of coarse-grained senses, in the form of the lexicographer file number, we are able to identify instances where these frequencies are particularly suspect, thus making slightly more accurate default sense predictions. We have also shown that a system limited to selecting default senses still has an upper bound far beyond current state of the art — even excluding rare words.

This is consistent with the results reported by McCarthy et al. (2004), who show that a classifier limited to selecting one sense per word was able to perform well if the sense was chosen intelligently. Their method, which relies on distributional similarity, might be adopted as a supersense selection heuristic. This might prove useful, as we have shown that using a different method to choose a supersense can be used to change the default prediction in cases where the simple baseline system performs poorly.

7 Acknowledgements

We would like to thank James Curran, James Gorman and Jon Patrick from the University of Sydney for their invaluable insights.

References

Timothy Chklovski and Rada Mihalcea. 2002. Building a sense tagged corpus with open mind word expert. In *Proceedings of the Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions"*, pages 116–122.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of EMNLP 2006*, pages 594–602.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense taggin of unknown nouns in wordnet. In *Proceedings of EMNLP 2003*.

Bart Decadt, Véronique Hoste, Walter Daelemans, and Antal van den Bosch. 2004. GAMBL, genetic algorithm optimization of memory-based

WSD. In *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, pages 108–112. Barcelona, Spain.

Christiane Fellbaum, editor. 1998. *Wordnet: An Electronic Lexical Database*. MIT Press.

Véronique Hoste, Anne Kool, and Walter Daelemans. 2001. Classifier optimization and combination in the english all words task. In *Proceedings of the* SENSEVAL*-2 workshop*, pages 84–86.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of HLT-NAACL 2006, New York*, pages 57–60.

Upali S. Kohomban and Wee Sun Lee. 2005. Learning semantic classes for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the ACL, Ann Arbor*, pages 34–41.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. Using automatically acquired predominant senses for word sense disambiguation. In *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, pages 151–154. Barcelona, Spain.

G. A. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3 DARPA Workshop on Human Language Technology*, pages 303–308.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Proceedings of the* SENSEVAL*-3 Workshop, Barcelona*, pages 41–43.

Experiments with Sentence Classification

Anthony Khoo, Yuval Marom and David Albrecht
Faculty of Information Technology, Monash University
Clayton, VICTORIA 3800, AUSTRALIA
nthony_ak@yahoo.com, {yuvalm,dwa}@csse.monash.edu.au

Abstract

We present a set of experiments involving sentence classification, addressing issues of representation and feature selection, and we compare our findings with similar results from work on the more general text classification task. The domain of our investigation is an email-based help-desk corpus. Our investigations compare the use of various popular classification algorithms with various popular feature selection methods. The results highlight similarities between sentence and text classification, such as the superiority of Support Vector Machines, as well as differences, such as a lesser extent of the usefulness of features selection on sentence classification, and a detrimental effect of common preprocessing techniques (stop-word removal and lemmatization).

1 Introduction

Classification tasks applied to textual data have been receiving increasing attention due to the explosion in digital presentation and storage of textual information, such as web pages, emails, publications, and discussion forums. The bulk of the research concerns the classification of complete documents, such as spam detection in emails (Drucker et al., 1999), and the classification of news articles (Yang and Pedersen, 1997; Joachims, 1998). These kinds of tasks are widely known as text classification (TC). A text document is best characterized by the words and terms it contains, and consequently the representation of textual data is often of a very high dimensionality. Thus, an important aspect of TC is feature selection (Yang and Pedersen, 1997; Forman, 2003).

There are numerous examples of textual documents whose content conveys communication between multiple parties. In such documents, it may be useful to classify individual sentences that express communicative acts, either to obtain a more meaningful description of the documents, or simply to extract meaningful components, such as action items or opinions. The computational linguistics community devotes considerable research into speech and dialogue acts, and has developed a markup convention for coding both spoken and written language (Core and Allen, 1997, for example). The classifications we use for sentences are inspired by such conventions.

Although there are existing implementations of sentence classification (SC) (Zhou et al., 2004; Wang et al., 2005; McKnight and Srinivasan, 2003), including ones where sentences convey communicative acts (Cohen et al., 2004; Corston-Oliver et al., 2004; Ivanovic, 2005), comparatively little attention has been given to SC in general. In particular, there are no empirical demonstrations of the effect of feature selection in SC tasks, to the best of our knowledge.

This paper presents a study into sentence classification, with particular emphasis on representational issues of extracting features from sentences, and applying feature selection (FS) methods. We experiment with various widely accepted FS methods and classification algorithms, and relate our findings to results from TC reported in the literature. Note that we do not offer any new methods in this paper. Rather, we offer some insight into the characteristics of SC and what distinguishes it from the more general TC, and this insight is driven by empirical findings. We believe that sen-

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 18–25.

Sentence Class	Frequency	Percentage	Sentence Class	Frequency	Percentage
APOLOGY	23	1.5%	SALUTATION	129	8.7%
INSTRUCTION	126	8.5%	SIGNATURE	32	2.2%
INSTRUCTION-ITEM	94	6.3%	SPECIFICATION	41	2.8%
OTHERS	22	1.5%	STATEMENT	423	28.5%
QUESTION	24	1.6%	SUGGESTION	55	3.7%
REQUEST	146	9.8%	THANKING	228	15.3%
RESPONSE-ACK	63	4.2%	URL	80	5.4%

Table 1: Sentence class distribution.

tence classification is emerging as an important task to investigate, due to the increasing interest in detecting intentional units at a sub-document level.

The rest of the paper is organized as follows. In the next section we present our domain of investigation. In Section 3 we discuss the experiments that we carried out, and we conclude the paper in Section 4.

2 Domain

Our corpus consists of 160 email dialogues between customers and operators at Hewlett-Packard's help-desk. These deal with a variety of issues, including requests for technical assistance, inquiries about products, and queries about how to return faulty products or parts. As an initial step in our study, we decided to focus only on the response emails, as they contain well-formed grammatical sentences, as opposed to the customers' emails. In future work we intend to extend our study to include both types of emails. The response emails contain 1486 sentences overall, which we have divided into the classes shown in Table 1. The classes are inspired by the SWBD-DAMSL tag set (Jurafsky et al., 1997), an adaptation of the Dialog Act Markup in Several Layers (DAMSL) annotation scheme (Core and Allen, 1997) for switchboard conversations. For example, RESPONSE-ACK refers to an acknowledgement by the operator of receiving the customer's request: *Your email was submitted to the HP eServices Commercial Support group*; INSTRUCTION-ITEM is similar to INSTRUCTION but appears as part of a list of instructions.

We can see from Table 1 that there is a high distribution skew, where some classes are very small. This means that many of the classes have very few positive examples to learn from. We will see various implications of this high skew in our investigation (Section 3).

When annotating the sentences, problems arose when a sentence was of compound form, which consisted of multiple independent clauses connected by conjunctions, like *"and"*, *"but"*, and *"or"*. For example, the sentence *"Please send us the error message and we will be able to help."*. The two clauses could be labeled as REQUEST and STATEMENT respectively. As our study considered only one tag per sentence, the annotators were asked to consider the most dominant clause to tag the sentence as a whole. Another tricky problem when tagging the sentences dealt with the complex sentences, which contained one independent clause and one or more dependent clauses, for example *"If you see any error message, please forward it to us"*. The first clause is a dependent clause, while the second one is an independent clause. To solve this problem, the annotators were asked to consider only the independent clause to determine which tag to use. Despite these difficulties, we obtained a high inter-tagger agreement, measured with the widely used Kappa statistic (Carletta, 1996) as 0.85. We had three annotators, and we considered only the sentences on which at least two of the annotators agreed. This was the case in all but 21 of the sentences.

3 Experiments

Our experiments involve three classification algorithms, Naive Bayes (NB), Decision Tree (DT), and Support Vector Machine (SVM). The evaluation platform is the machine learning software toolkit WEKA (Witten and Frank, 2005). For the SVM, the multi-class task is implemented as a series of binary classification tasks. We employ a stratified 10-fold validation procedure, where the labelled sentences are randomly allocated to training and testing data splits.

A standard measure for classification performance is classification accuracy. However, for

datasets with skewed distribution this measure can be misleading, and so instead we have used the F_1 measure, derived from precision and recall (Salton and McGill, 1983), as follows. The precision of a class i is defined as

$$\text{Precision} = \frac{\text{\# sentences correctly classified into class i}}{\text{\# of sentences classified into class i}}$$

and the recall of class i is defined as

$$\text{Recall} = \frac{\text{\# sentences correctly classified into class i}}{\text{\# of sentences that are truly in class i}}$$

and then F_1, the harmonic mean between precision and recall, is defined as

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Once the F_1 measure is calculated for all the classes, we average it to get an overall indication of performance, and also look at the standard deviation as an indication of consistency. The average can be computed in two different methods to reflect the importance of small classes. The first method, called *macro averaging*, gives an equal weight to each class. The second, called *micro averaging*, gives proportional weight according to the proportion of the classes in the dataset. For classes with only a few positive training data, it is generally more difficult to achieve good classification, and their poor performance will have a larger effect on the overall performance when the macro average is used. The choice between the two measures depends on the relative preference that an experimenter places on the smaller classes. Since the classes in our corpus have unbalanced distributions (Table 1) we consider both alternatives and discuss their differences.

3.1 Experiments with representation

Before looking at feature selection, we investigate different techniques for extracting features from sentences. Finding a useful representation for textual data can be very challenging, and the success of classification hinges on this crucial step. Many different techniques have been suggested for text classification, and we have investigated the most common ones.

3.1.1 Representation techniques

Bag-of-words (BoW). Each distinct word in the text corresponds to a feature, and the text is transformed to a vector of N weights ($< w_1, w_2, \ldots, w_N >$), where N is the total number of distinct words in the entire corpus, and w_k is the weight of the k^{th} word in the vector. Information about sentence order, word order and the structure of the text and sentence are discarded. The BoW representation is widely used due to its simplicity and computational efficiency (Cardoso-Cachopo and Oliveira, 2003). There are various methods for setting the weights, for example, solely taking into account the presence of the word, or also considering the frequency of the word. Since we are dealing with sentences that are usually quite short, we do not believe that the frequency of each word conveys any meaning. This is in contrast to typical text classification tasks. We use a binary word-presence representation, indicating whether a word is present or absent from the sentence.[1]

Stop-word removal. Generally, the first step to reduce the feature space is to remove the stop-words (connective words, such as *"of"*, *"the"*, *"in"*). These words are very common words and are conjectured in TC to provide no information to the classifier. Stop-word removal is said to be used in almost all text classification experiments (Scott and Matwin, 1999).

Tokenization. This involves separating any symbols from the numbers or alphabets. For example, the word *"(manual12.txt)"* is separated into five tokens, *"("*, *"manual12"*, *"."*, *"txt"* and *")"*, all considered as features. Without tokenization, a word that is coupled with different symbols may lose its discriminative power because the BoW treats each coupling as a distinct feature. Similarly, the symbols lose any discriminative power.

Lemmatization. The process of mapping words into their base form. For example, the words *"installed"*, *"installs"* and *"installing"* are mapped to *"install"*. This mapping makes the bag-of-words approach treat words of different forms as a single feature, hence reducing the total number of features. This mapping can increase the discriminative power of a word if that word appears in a particular sentence class but in different forms.

Grouping. This involves grouping certain types of words into a single feature. For instance, all words that are valid numbers, like *"1"*, *"444"* and

[1] We have also attempted a bigram representation, however, our results so far are inconclusive and require further investigation.

Representation	Num Features	Measure	NB	DT	SVM
Basic	2710	micro-F_1 ave	0.481	0.791	0.853
		macro-F_1 ave	0.246	0.693	0.790
Best	1622	micro-F_1 ave	0.666	0.829	0.883
		macro-F_1 ave	0.435	0.803	0.866

Table 2: Classification performance using different representations.

"9834", are grouped to represent a single feature. Grouping was also applied to email addresses, phone numbers, URLs, and serial numbers. As opposed to the other techniques mentioned above, grouping is a domain-specific preprocessing step.

3.1.2 Results.

We begin our investigation by looking at the most basic representation, involving a binary BoW without any further processing. The first row in Table 2 shows the results obtained with this basic setup. The second column shows the number of features resulting from this representation, the third column shows the performance measure, and the last three columns show the results for the three classifiers. We see that SVM outperforms the other classifiers on both measures. We also inspected the standard deviations of the macro averages and observed that SVM is the most consistent (0.037 compared to 0.084 and 0.117 for DT and NB, respectively). This means the SVM's performance is most consistent across the different classes. These results are in line with observations reported in the literature on the superiority of SVMs in classification tasks involving text, where the dimensionality is high. We will return to this issue in the next sub-section when we discuss feature selection. We can also see from Table 2 that the micro F_1 average consistently reports a better performance than the macro F_1 average. This is expected due to the existence of very small classes: their performance tends to be poorer, but their influence on the micro average is proportional to their size, as opposed to the macro average which takes equal weights.

We have experimented with different combinations of the various representation techniques mentioned above (Anthony, 2006). The best one turned out to be one that uses tokenization and grouping, and its results are shown in the second row of Table 2. We can see that it results in a significant reduction in the number of features (approximately 40%). Further, it provides a consistent improvement in all performance measures for all classifiers, with the exception of NB, for which the standard deviation is slightly increased. We see that the more significant improvements are reported by the macro F_1 average, which suggests that the smaller classes are particularly benefiting from this representation. For example, serial numbers occur often in SPECIFICATION class. If grouping was not used, serial numbers often appear in different variation, making them distinct from each other. Grouping makes them appear as a single more predictive feature. To test this further, the SPECIFICATION class was examined with and without grouping. Its classification performance improved from 0.64 (no grouping) to 0.8 (with grouping) with SVM as the classifier. An example of the effect of tokenization can be observed for the QUESTION class, which improved largely because of the question mark symbol '?' being detected as a feature after the tokenization process. Notice that there is a similar increase in performance for NB when considering either the micro or macro average. That is, NB has a more general preference to the second representation, and we conjecture that this is due to the fact that it does not deal well with many features, because of the strong assumption it makes about the independence of features.

The surprising results from our investigations are that two of the most common preprocessing techniques, stop-word removal and lemmatization, proved to be harmful to performance. Lemmatization can harm classification when certain classes rely on the raw form of certain words. For example, the INSTRUCTION class often has verbs in imperative form, for example, "install the driver", but these same verbs can appear in a different form in other classes, for example the SUGGESTION sentence "I would try installing the driver", or the QUESTION sentence "Have you installed the driver?". Stop-words can also carry crucial information about the structure of the sentence, for example, "what", "how", and "please". In fact, often the words in our stop-list appeared in the top list of

words produced by the feature selection methods. We conclude that unlike text classification tasks, where each item to be classified is rich with textual information, sentence classification involves small textual units that contain valuable cues that are often lost when techniques such as lemmatization and stop-word removal are employed.

3.2 Experiments with feature selection

Since there can be thousands or even tens of thousands of distinct words in the entire email corpus, the feature space can be very large, as we have seen in our baseline experiments (Table 2). This means that the computational load on a classification algorithm can be very high. Thus feature selection (FS) is desirable for reducing this load. However, it has been demonstrated in text classification tasks that FS can in fact improve classification performance as well (Yang and Pedersen, 1997).

We investigate four FS methods that have been shown to be competitive in text classification (Yang and Pedersen, 1997; Forman, 2003; Gabrilovich and Markovitch, 2004), but have not been investigated in sentence classification.

3.2.1 Feature selection algorithms

Chi-squared (χ^2). Measures the lack of statistical independence between a feature and a class (Seki and Mostafa, 2005). If the independence is high, then the feature is considered not predictive for the class. For each word, χ^2 is computed for each class, and the maximum score is taken as the χ^2 statistic for that word.

Information Gain (IG). Measures the entropy when the feature is present versus the entropy when the feature is absent (Forman, 2003). It is quite similar to χ^2 in a sense that it considers the usefulness of a feature not only from its presence, but also from its absence in each class.

Bi-Normal Separation (BNS). This is a relatively new FS method (Forman, 2003). It measures the separation along a Standard Normal Distribution of two thresholds that specify the prevalence rate of the feature in the positive class versus the negative class. It has been shown to be as competitive as χ^2 and IG (Forman, 2003; Gabrilovich and Markovitch, 2004), and superior when there is a large class skew, as there is in our corpus.

Sentence Frequency (SF). This is a baseline FS method, which simply removes features that are infrequent. The sentence frequency of a word is the number of sentences in which the word appears. Thus this method is much cheaper computationally than the others, but has been shown to be as competitive when at least 10% of the words are kept (Yang and Pedersen, 1997).

3.2.2 Results.

We evaluate the various FS methods by inspecting the performance of the classifiers when trained with increasing number of features, where we retain the top features as determined by each FS method. Figure 1(a) shows the results obtained with the χ^2 method, reported using the macro F_1 average, where the error bars correspond to the 95% confidence intervals of these averages. We can see from the figure that SVM and DT are far less sensitive to feature selection than NB. As conjectured in the previous sub-section, NB does not deal well with many features, and indeed we can see here that it performs poorly and inconsistently when many of the features are retained. As we filter out more features, its performance starts to improve and become more consistent. In contrast, the SVM seems to prefer more features: its performance degrades slightly if less than 300 features are retained (although it still outperforms the other classifiers), and levels out when at least 300 features are used. As well as having an overall better performance than the other two classifiers, it also has the smallest variability, indicating a more consistent and robust behaviour. SVMs have been shown in text classification to be more robust to many features (Joachims, 1998).

When comparing the FS methods against each other, it seems their performance is not significantly distinguishable. Figure 1(b) shows the performance of the four methods for the NB classifier. We see that when at least 300 features are retained, the performances of the FS methods are indistinguishable, with IG and χ^2 slightly superior. When less than 300 features are retained, the performance of the SF method deteriorates compared to the others. This means that if we only want very few features to be retained, a frequency-based method is not advisable. This is due to the fact that we have small classes in our corpus, whose cue words are therefore infrequent, and therefore we need to select features more carefully. However, if we can afford to use many features, then this

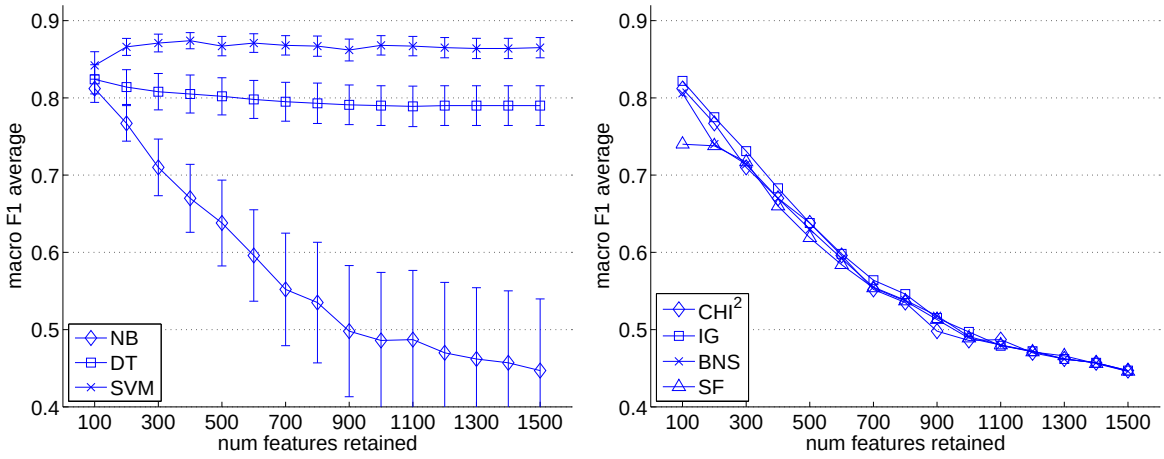

Figure 1: Results from feature selection: (a) the effect of the χ^2 method on different classifiers, (b) the effect of different feature selection methods on the Naive Bayes classifier.

simple method is adequate. We have observed the pattern seen in Figure 1(b) also with the other classifiers, and with the micro F_1 average (Anthony, 2006).

Our observations are in line with those from text classification experiments: the four FS methods perform similarly, except when only a small proportion is retained, when the simple frequency-based method performs worse. However, we expected the BNS method to outperform the others given that we are dealing with classes with a high distributional skew. We offer two explanations for this result. First, the size of our corpus is smaller than the one used in the text classification experiments involving skewed datasets (Forman, 2003) (these experiments use established benchmark datasets consisting of large sets of labelled text documents, but there are no such datasets with labelled sentences). Our smaller corpus therefore results in a substantially fewer number of features (1622 using the our "best" representation in Table 2 compared with approximately 5000 in the text classification experiments). Thus, it is possible that the effect of BNS can only be observed when a more substantial number of features is presented to the selection algorithm. The second explanation is that sentences are less textually rich than documents, with fewer irrelevant and noisy features. They might not rely on feature selection to the extent that text classification tasks do. Indeed, our results show that as long as we retain a small proportion of the features, a simple FS method suffices. Therefore, the effect of BNS cannot be observed.

3.3 Class-by-class analysis

So far we have presented average performances of the classifiers and FS methods. It is also interesting to look at the performance individually for each class. Table 3 shows how well each class was predicted by each classifier, using the macro F_1 average and standard deviation in brackets. The standard deviation was calculated over 10 cross-validation folds. These results are obtained with the "best" representation in Table 2, and with the χ^2 feature selection method retaining the top 300 features.

We can see that a few classes have F_1 of above 0.9, indicating that they were highly predictable. Some of these classes have obvious cue words to distinguish them from other classes. For instance, *"inconvenience"*, *"sorry"*, *"apologize"* and *"apology"* to discriminate APOLOGY class, *"?"* to discriminate QUESTION, *"please"* to discriminate REQUEST and *"thank"* to discriminate THANKING.

It is more interesting to look at the less predictable classes, such as INSTRUCTION, INSTRUCTION-ITEM, SUGGESTION and SPECIFICATION. They are also the sentence classes that are considered more useful to know than some others, like THANKING, SALUTATION and so on. For instance, by knowing which sentences are instructions in the emails, they can be extracted into a to-do list of the email recipient. We have inspected the classification confusion matrix to better understand the less predictable classes. We saw that INSTRUCTION, INSTRUCTION-ITEM,

Sentence Class	NB	DT	SVM
Apology	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Instruction	0.593 (0.109)	0.619 (0.146)	0.675 (0.126)
Instruction-item	0.718 (0.097)	0.582 (0.141)	0.743 (0.127)
Others	0.117 (0.249)	0.411 (0.283)	0.559 (0.282)
Question	0.413 (0.450)	1.000 (0.000)	1.000 (0.000)
Request	0.896 (0.042)	0.930 (0.046)	0.940 (0.047)
Response-ack	0.931 (0.061)	0.902 (0.037)	0.942 (0.057)
Salutation	0.908 (0.029)	0.972 (0.028)	0.981 (0.020)
Signature	0.370 (0.362)	0.960 (0.064)	0.986 (0.045)
Specification	0.672 (0.211)	0.520 (0.218)	0.829 (0.151)
Statement	0.837 (0.042)	0.843 (0.040)	0.880 (0.035)
Suggestion	0.619 (0.206)	0.605 (0.196)	0.673 (0.213)
Thanking	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Url	0.870 (0.071)	0.970 (0.041)	0.988 (0.025)

Table 3: Class-by-class performance

SUGGESTION and STATEMENT were often misclassified as one another. This means that there were not enough distinguishing features to clearly separate these classes. The highest confusion was between INSTRUCTION and STATEMENT, and indeed, sentences of the form *"the driver must be installed before the device will work"* can be interpreted as both an instruction and a general statement. This suggests that the usage of some of these sentence classes may need to be revised.

4 Conclusions

We have presented a set of experiments involving sentence classification. While the successful deployment of classification algorithms for sentences has been demonstrated previously, this kind of classification has received far less attention than the one involving complete documents. In particular, the usefulness of feature selection for sentence classification has not been investigated, to the best of our knowledge.

There are many types of documents where individual sentences carry important information regarding communicative acts between parties. In our experiments this corresponds to email responses to technical help-desk inquiries. However, there are many more examples of such documents, including different kinds of emails (both personal and professional), newsgroup and forum discussions, on-line chat, and instant messaging. Therefore, sentence classification is a useful task that deserves more investigation. In particular, such investigations need to relate results to the more well established ones from text classification experiments, and thus highlight the significant differences between these two tasks.

Our results confirm some observations made from text classification. The SVM classification algorithm generally outperforms other common ones, and is largely insensitive to feature selection. Further, the effect of non-trivial feature selection algorithms is mainly observed when an aggressive selection is required. When a less aggressive selection is acceptable (that is, retaining more features), a simple and computationally cheap frequency-based selection is adequate. Our results also show some important differences between text and sentence classification. Sentences are much smaller than documents, and less rich with textual information. This means that in pruning the feature space one needs to be very careful not to eliminate strong discriminative features, especially when there is a large class distribution skew. We saw that lemmatization and stopword removal proved detrimental, whereas they have been demonstrated to provide a useful dimensionality reduction in text classification. This difference between sentences and documents may also be responsible for obscuring the effect of a particular feature selection method (BNS), which has been demonstrated to outperform others when there is a large distribution skew. We conclude from these observations that while feature selection is useful for reducing the dimensionality of the classification task and even improving the performance of some classifiers, the extent of its usefulness is not as large as in text classification.

Acknowledgments

This research was supported in part by grant LP0347470 from the Australian Research Council and by an endowment from Hewlett-Packard. The authors also thank Hewlett-Packard for the extensive help-desk data, Ingrid Zukerman, Peter

Tischer and anonymous reviewers for their valuable feedback, and Edward Ivanovic for providing a prototype for our tagging software.

References

Anthony. 2006. Sentence classifier for helpdesk emails. Honours Thesis, Clayton School of Information Technology, Monash University.

Ana Cardoso-Cachopo and Arlindo Limede Oliveira. 2003. An empirical comparison of text categorization methods. In *String Processing and Information Retrieval, 10th International Symposium*, pages 183–196, Brazil, October.

Jean Carletta. 1996. Assessing agreement on classification tasks: The Kappa statistic. *Computational Linguistics*, 22(2):249–254.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "Speech Acts". In *Proceedings of Empirical Methods in Natural Language Processing*, pages 309–316, Barcelona, Spain, July. Association for Computational Linguistics.

Mark G. Core and James F. Allen. 1997. Coding dialogues with the DAMSL annotation scheme. In David Traum, editor, *Working Notes: AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Menlo Park, California. American Association for Artificial Intelligence.

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

Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. 1999. Support Vector Machines for spam categorization. *IEEE Transactions on Neural Network*, 10(5):1048–1054.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(7–8):1289–1305. Cambridge, MA, MIT Press.

Evgeniy Gabrilovich and Shaul Markovitch. 2004. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of the 21st International Conference on Machine Learning*, pages 321–328, Alberta, Canada.

Edward Ivanovic. 2005. Dialogue act tagging for instant messaging chat sessions. In *Proceedings of the ACL Student Research Workshop*, pages 79–84, Ann Arbor, Michigan.

Thorsten Joachims. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, pages 137–142.

Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Andreas Stolcke, Paul Taylor, and Carol Van Ess-Dykema. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Proceedings of the IEEE Workshop on Speech Recognition and Understanding*, pages 88–95, Santa Barbara, CA, December.

Larry McKnight and Padmini Srinivasan. 2003. Categorization of sentence types in medical abstracts. In *Proceedings of the American Medical Informatics Association Annual Symposium*, pages 440–444, Washington D.C.

Gerard Salton and Michael J. McGill. 1983. *An Introduction to Modern Information Retrieval*. McGraw-Hill.

Sam Scott and Stan Matwin. 1999. Feature engineering for text classification. In *Proceedings of ICML-99: The 16th International Conference on Machine Learning*, pages 379–388, Slovenia.

Kazuhiro Seki and Javed Mostafa. 2005. An application of text categorization methods to gene ontology annotation. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 138–145, Salvador, Brazil.

Chao Wang, Jie Lu, and Guangquan Zhang. 2005. A semantic classification approach for online product reviews. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 276–279, Compiegne, France.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Fransisco, 2nd edition.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97: The 14th International Conference on Machine Learning*, pages 412–420, Nashville, US.

Liang Zhou, Miruna Ticrea, and Eduard Hovy. 2004. Multi-document biography summarization. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 434–441, Barcelona, Spain.

Computational Semantics in the *Natural Language Toolkit*

Ewan Klein
School of Informatics
University of Edinburgh
Scotland, UK
ewan@inf.ed.ac.uk

Abstract

NLTK, the Natural Language Toolkit, is an open source project whose goals include providing students with software and language resources that will help them to learn basic NLP. Until now, the program modules in NLTK have covered such topics as tagging, chunking, and parsing, but have not incorporated any aspect of semantic interpretation. This paper describes recent work on building a new semantics package for NLTK. This currently allows semantic representations to be built compositionally as a part of sentence parsing, and for the representations to be evaluated by a model checker. We present the main components of this work, and consider comparisons between the Python implementation and the Prolog approach developed by Blackburn and Bos (2005).

1 Introduction

NLTK, the Natural Language Toolkit,[1] is an open source project whose goals include providing students with software and language resources that will help them to learn basic NLP. NLTK is implemented in Python, and provides a set of modules (grouped into packages) which can be imported into the user's Python programs.

Up till now, the modules in NLTK have covered such topics as tagging, chunking, and parsing, but have not incorporated any aspect of semantic interpretation. Over the last year, I have been working on remedying this lack, and in this paper I will describe progress to date. In combination with the NLTK parse package, NLTK's semantics package currently allow semantic representations to be built compositionally within a feature-based chart parser, and allows the representations to be evaluated by a model checker.

One source of inspiration for this work came from Blackburn and Bos's (2005) landmark book *Representation and Inference for Natural Language* (henceforth referred to as B&B). The two primary goals set forth by B&B are (i) automating the association of semantic representations with expressions of natural language, and (ii) using logical representations of natural language to automate the process of drawing inferences. I will be focussing on (i), and the related issue of defining satisfaction in a model for the semantic representations. By contrast, the important topic of (ii) will not be covered—as yet, there are no theorem provers in NLTK. That said, as pointed out by B&B, for many inference problems in NLP it is desirable to call external and highly sophisticated first-order theorem provers.

One notable feature of B&B is the use of Prolog as the language of implementation. It is not hard to defend the use of Prolog in defining logical representations, given the presence of first-order clauses in Prolog and the fundamental role of resolution in Prolog's model of computation. Nevertheless, in some circumstances it may be helpful to offer students access to an alternative framework, such as the Python implementation presented here. I also hope that the existence of work in both programming paradigms will turn out to be mutually beneficial, and will lead to a broader community of upcoming researchers becoming involved in the area of computational semantics.

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

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 26–33.

2 Building Semantic Representations

The initial question that we faced in NLTK was how to induce semantic representations for English sentences. Earlier efforts by Edward Loper and Rob Speer had led to the construction of a chart parser for (untyped) feature-based grammars, and we therefore decided to introduce a `sem` feature to hold the semantics in a parse tree node. However, rather than representing the value of `sem` as a feature structure, we opted for a more traditional (and more succinct) logical formalism. Since the λ calculus was the pedagogically obvious choice of 'glue' language for combining the semantic representations of subconstituents in a sentence, we opted to build on church.py,[2] an independent implementation of the untyped λ calculus due to Erik Max Francis. The NLTK module semantics.logic extends church.py to bring it closer to first-order logic, though the resulting language is still untyped. (1) illustrates a representative formula, translating *A dog barks*. From a Python point of view, (1) is just a string, and has to be parsed into an instance of the `Expression` class from semantics.logic.

(1) `some x.(and (dog x) (bark x))`

The string `(dog x)` is analyzed as a function application. A statement such as *Suzie chases Fido*, involving a binary relation *chase*, will be translated as another function application: `((chase fido) suzie)`, or equivalently `(chase fido suzie)`. So in this case, `chase` is taken to denote a function which, when applied to an argument yields the second function denoted by `(chase fido)`. Boolean connectives are also parsed as functors, as indicated by `and` in (1). However, infix notation for Boolean connectives is accepted as input and can also be displayed.

For comparison, the Prolog counterpart of (1) on B&B's approach is shown in (2).

(2) `some(X,(and(dog(X),bark(X))`

(2) is a Prolog term and does not require any additional parsing machinery; first-order variables are treated as Prolog variables.

(3) illustrates a λ term from semantics.logic that represents the determiner *a*.

(3) `\Q P.some x.(and (Q x) (P x))`

[2] `http://www.alcyone.com/pyos/church/`.

`\Q` is the ascii rendering of λQ, and `\Q P` is shorthand for $\lambda Q \lambda P$.

For comparison, (4) illustrates the Prolog counterpart of (3) in B&B.

(4)
```
lam(Q,lam(P,some(X,
      and(app(Q,X),app(P,X)))))
```

Note that `app` is used in B&B to signal the application of a λ term to an argument. The right-branching structure for λ terms shown in the Prolog rendering can become fairly unreadable when there are multiple bindings. Given that readability is a design goal in NLTK, the additional overhead of invoking a specialized parser for logical representations is arguable a cost worth paying.

Figure 1 presents a minimal grammar exhibiting the most important aspects of the grammar formalism extended with the `sem` feature. Since the values of the `sem` feature have to handed off to a separate processor, we have adopted the convention of enclosing the values in angle brackets, except in the case of variables (e.g., `?subj` and `?vp`), which undergo unification in the usual way. The `app` relation corresponds to function application;

In Figure 2, we show a trace produced by the NLTK module parse.featurechart. This illustrates how variable values of the `sem` feature are instantiated when completed edges are added to the chart. At present, β reduction is not carried out as the `sem` values are constructed, but has to be invoked after the parse has completed.

The following example of a session with the Python interactive interpreter illustrates how a grammar and a sentence are processed by a parser to produce an object `tree`; the semantics is extracted from the root node of the latter and bound to the variable `e`, which can then be displayed in various ways.

```
>>> gram = GrammarFile.read_file('sem1.cfg')
>>> s = 'a dog barks'
>>> tokens = list(tokenize.whitespace(s))
>>> parser = gram.earley_parser()
>>> tree = parser.parse(tokens)
>>> e = root_semrep(tree)
>>> print e
(\Q P.some x.(and (Q x) (P x)) dog \x.(bark x))
>>> print e.simplify()
some x.(and (dog x) (bark x))
>>> print e.simplify().infixify()
some x.((dog x) and (bark x))
```

Apart from the pragmatic reasons for choosing a functional language as our starting point,

```
S[sem = <app(?subj,?vp)>] -> NP[sem=?subj] VP[sem=?vp]
VP[sem=?v] -> IV[sem=?v]
NP[sem=<app(?det,?n)>] -> Det[sem=?det] N[sem=?n]

Det[sem=<\Q P. some x. ((Q x) and (P x))>] -> 'a'
N[sem=<dog>] -> 'dog'
IV[sem=<\x.(bark x)>] -> 'barks'
```

Figure 1: Minimal Grammar with Semantics

```
Predictor |> . . .| S[sem='(?subj ?vp)'] -> * NP[sem=?subj] VP[sem=?vp]
Predictor |> . . .| NP[sem='(?det ?n)'] -> * Det[sem=?det] N[sem=?n]
Scanner   |[-] . .| [0:1] 'a'
Completer |[-> . .| NP[sem='(\\Q P.some x.(and (Q x) (P x)) ?n)']
                    -> Det[sem='\\Q P.some x.(and (Q x) (P x))'] * N[sem=?n]
Scanner   |. [-] .| [1:2] 'dog'
Completer |[---] .| NP[sem='(\\Q P.some x.(and (Q x) (P x)) dog)']
                    -> Det[sem='\\Q P.some x.(and (Q x) (P x))'] N[sem='dog'] *
Completer |[--->.| S[sem='(\\Q P.some x.(and (Q x) (P x)) dog ?vp)']
                    -> NP[sem='(\\Q P.some x.(and (Q x) (P x)) dog)'] * VP[sem=?vp]
Predictor |. . > .| VP[sem=?v] -> * V[sem=?v]
Scanner   |. . [-]| [2:3] 'barks'
Completer |. . [-]| VP[sem='bark'] -> V[sem='bark'] *
Completer |[=====]| S[sem='(\\Q P.some x.(and (Q x) (P x)) dog bark)']
                    -> NP[sem='(\\Q P.some x.(and (Q x) (P x)) dog)'] VP[sem='bark'] *
Completer |[=====]| [INIT] -> S *
```

Figure 2: Parse tree for *a dog barks*

there are also theoretical attractions. It helps introduce students to the tradition of Montague Grammar (Montague, 1974; Dowty et al., 1981), which in turn provides an elegant correspondence between binary syntax trees and semantic composition rules, in the style celebrated by categorial grammar. In the next part of the paper, I will turn to the issue of how to represent models for the logical representations.

3 Representing Models in Python

Although our logical language is untyped, we will interpret it as though it were typed. In particular, expressions which are intended to translate unary predicates will be interpreted as functions of type $e \rightarrow \{0, 1\}$ (from individuals to truth values) and expressions corresponding to binary predicates will be interpreted as though they were of type $e \rightarrow (e \rightarrow \{0, 1\})$. We will start out by looking at data structures which can be used to provide denotations for such expressions.

3.1 Dictionaries and Boolean Types

The standard mapping type in Python is the dictionary, which associates keys with arbitrary values. Dictionaries are the obvious choice for representing various kinds of functions, and can be specialized by user-defined classes. This means that it is possible to benefit from the standard Python operations on dictionaries, while adding additional features and constraints, or in some cases overriding the standard operations. Since we are assuming that our logical language is based on function application, we can readily construct the interpretation of n-ary relations in terms of dictionaries-as-functions.

Characteristic functions (i.e., functions that correspond to sets) are dictionaries with Boolean values:

```
cf = {'d1': True,
      'd2': True,
      'd3': False}
```

`cf` corresponds to the set $\{d_1, d_2\}$. Since functions are being implemented as dictionaries, func-

tion application is implemented as indexing (e.g., `cf['d1']` applies `cf` to argument `'d1'`). Note that `True` and `False` are instances of the Python built-in `bool` type, and can be used in any Boolean context. Since Python also includes `and` and `not`, we can make statements (here, using the Python interactive interpreter) such as the following:

```
>>> cf['d1'] and not cf['d3']
True
```

As mentioned earlier, relations of higher arity are also modeled as functions. For example, a binary relation will be a function from entities to a characteristic function; we can call these 'curryed characteristic functions'.

```
cf2 = {'d2': {'d1': True},
       'd3': {'d2': True}}
```

`cf2` corresponds to the relation $\{(d_1, d_2), (d_2, d_3)\}$, on two assumptions. First, we are allowed to omit values terminating in `False`, since arguments that are missing the function will be taken to yield `False`. Second, as in Montague Grammar, the 'object' argument of a binary relation is consumed before the 'subject' argument. Thus we write $((love\ m)\ j)$ in place of $love(j, m)$. Recall that we also allow the abbreviated form $(love\ m\ j)$

Once we have curryed characteristic functions in place, it is straightforward to implement the valuation of non-logical constants as a another dictionary-based class `Valuation`, where constants are the keys and the values are functions (or entities in the case of individual constants).

While variable assignments could be treated as a list of variable-value pairs, as in B&B, an alternative is again to use a dictionary-based class. This approach makes it relatively easy to impose further restrictions on assignments, such as only assigning values to strings of the form `x`, `y`, `z`, `x0`, `x1`, ...

3.2 Sets

Python provides support for sets, including standard operations such as intersection and subset relationships. Sets are useful in a wide variety of contexts. For example, instances of the class `Valuation` can be given a property `domain`, consisting of the `set` of entities that act as keys in curryed characteristic functions; then a condition on objects in the `Model` class is that the `domain` of some model `m` is a superset of `m`'s `valuation.domain`:

```
m.domain.issuperset
      (m.valuation.domain)
```

For convenience, `Valuation` objects have a `read` method which allows n-ary predicates to be specified as relations (i.e., sets of tuples) rather than functions. In the following example, `rel` is a set consisting of the pairs `('d1', 'd2')` and `('d2', 'd3')`.

```
val = Valuation()
rel = set([('d1', 'd2'),('d2', 'd3')])
val.read([('love', rel)])
```

`read` converts `rel` internally to the curryed characteristic function `cf2` defined earlier.

4 Key Concepts

4.1 Satisfaction

The definition of satisfaction presupposes that we have defined a first-order language, and that we have a way of parsing that language so that satisfaction can be stated recursively. In the interests of modularity, it seems desirable to make the relationship between language and interpretation less tightly coupled than it is on the approach of B&B; for example, we would like to be able apply similar evaluation techniques to different logical representations. In the current NLTK implementation, the `nltk_lite.semantics.evaluate` module imports a second module `logic`, and calls a `parse` method from this module to determine whether a given Python string can be analysed as first-order formula. However, `evaluate` tries to make relatively weak assumptions about the resulting parse structure. Specifically, given a parsed expression, it tries to match the structure with one of the following three kinds of pattern:

```
(binder, body)
(op, arg_list)
(fun, arg)
```

Any string which cannot be decomposed is taken to be a primitive (that is, a non-logical constant or individual variable).

A `binder` can be a λ or a quantifier (existential or universal); an `op` can be a Boolean connective or the equality symbol. Any other paired expression is assumed to be a function application. In principle, it should be possible to interface the `evaluate` module with any parser for first-order formulas which can deliver these structures. Although the model checker expects predicate-argument structure as function applications, it would be straightforward to accept atomic clauses that have been parsed into a predicate and a list of arguments.

Following the functional style of interpretation, Boolean connectives in `evaluate` are interpreted as truth functions; for example, the connective `and` can be interpreted as the function `AND`:

```
AND = {True:  {True: True,
               False: False},
       False: {True: False,
               False: False}}
```

We define `OPS` as a mapping between the Boolean connectives and their associated truth functions. Then the simplified clause for the satisfaction of Boolean formulas looks as follows:[3]

```
def satisfy(expr, g):
   if parsed(expr) == (op, args)
      if args == (phi, psi):
         val1 = satisfy(phi, g)
         val2 = satisfy(psi, g)
         return OPS[op][val1][val2]
```

In this and subsequent clauses for `satisfy`, the return value is intended to be one of Python's Boolean values, `True` or `False`. (The exceptional case, where the result is undefined, is discussed in Section 4.3.)

An equally viable (and probably more efficient) alternative to logical connnectives would be to use the native Python Boolean operators. The approach adopted here was chosen on the grounds that it conforms to the functional framework adopted elsewhere in the semantic representations, and can be expressed succinctly in the satisfaction clauses. By contrast, in the B&B Prolog implementation, `and` and `or` each require five clauses in the satisfaction definition (one for each combination of Boolean-valued arguments, and a fifth for the 'undefined' case).

We will defer discussion of the quantifiers to the next section. The `satisfy` clause for function application is similar to that for the connectives. In order to handle type errors, application is delegated to a wrapper function `app` rather than by directly indexing the curryed characteristic function as described earlier.

```
   ...
   elif parsed(expr) == (fun, arg):
      funval = satisfy(fun, g)
      argval = satisfy(psi, g)
      return app(funval, argval)
```

4.2 Quantifers

Examples of quantified formulas accepted by the `evaluate` module are pretty unexceptional. *Some boy loves every girl* is rendered as:

```
'some x.((boy x) and
      all y.((girl y) implies
                     (love y x)))'
```

The first step in interpreting quantified formulas is to define the *satisfiers* of a formula that is open in some variable. Formally, given an open formula $\phi[x]$ dependent on x and a model with domain D, we define the set $sat(\phi[x], g)$ of satisfiers of $\phi[x]$ to be:

$$\{u \in D : satisfy(\phi[x], g[u/x]) = True\}$$

We use '$g[u/x]$' to mean that assignment which is just like g except that $g(x) = u$. In Python, we can build the set $sat(\phi[x], g)$ with a `for` loop.[4]

```
def satisfiers(expr, var, g):
    candidates = []
    if freevar(var, expr):
        for u in domain:
            g.add(u, var)
            if satisfy(expr, g):
                candidates.append(u)
    return set(candidates)
```

An existentially quantified formula $\exists x.\phi[x]$ is held to be true if and only if $sat(\phi[x], g)$ is nonempty. In Python, `len` can be used to return the cardinality of a set.

[3] In order to simplify presentation, tracing and some error handling code has been omitted from definitions. Object-oriented uses of `self` have also been suppressed.

[4] The function `satisfiers` is an instance method of the `Models` class, and `domain` is an attribute of that class.

```
...
elif parsed(expr) == (binder, body):
    if binder == ('some', var):
        sat = satisfiers(body, var, g)
        return len(sat) > 0
```

In other words, a formula $\exists x.\phi[x]$ has the same value in model M as the statement that the number of satisfiers in M of $\phi[x]$ is greater than 0.

For comparison, Figure 3 shows the two Prolog clauses (one for truth and one for falsity) used to evaluate existentially quantified formulas in the B&B code `modelChecker2.pl`. One reason why these clauses look more complex than their Python counterparts is that they include code for building the list of satisfiers by recursion. However, in Python we gain bivalency from the use of Boolean types as return values, and do not need to explicitly mark the polarity of the satisfaction clause. In addition, processing of sets and lists is supplied by a built-in Python library which avoids the use of predicates such as `memberList` and the `[Head|Tail]` notation.

A universally quantified formula $\forall x.\phi[x]$ is held to be true if and only if every u in the model's domain D belongs to $sat(\phi[x], g)$. The `satisfy` clause above for existentials can therefore be extended with the clause:

```
...
elif parsed(expr) == (binder, body):
    ...
    elif binder == ('all', var):
        sat = self.satisfiers(body,var,g)
        return domain.issubset(sat)
```

In other words, a formula $\forall x.\phi[x]$ has the same value in model M as the statement that the domain of M is a subset of the set of satisfiers in M of $\phi[x]$.

4.3 Partiality

As pointed out by B&B, there are at least two cases where we might want the model checker to yield an 'Undefined' value. The first is when we try to assign a semantic value to an unknown vocabulary item (i.e., to an unknown non-logical constant). The second arises through the use of partial variable assignments, when we try to evaluate $g(x)$ for some variable x that is outside g's domain. We adopt the assumption that if any subpart of a complex expression is undefined, then the whole expression is undefined.[5] This means that an 'undefined' value needs to propagate through all the recursive clauses of the `satisfy` function. This is potentially quite tedious to implement, since it means that instead of the clauses being able to expect return values to be Boolean, we also need to allow some alternative return type, such as a string. Fortunately, Python offers a nice solution through its exception handling mechanism.

It is possible to create a new class of exceptions, derived from Python's `Exception` class. The `evaluate` module defines the class `Undefined`, and any function called by `satisfy` which attempts to interpret unknown vocabulary or assign a value to an out-of-domain variable will raise an `Undefined` exception. A recursive call within `satisfy` will automatically raise an `Undefined` exception to the calling function, and this means that an 'undefined' value is automatically propagated up the stack without any additional machinery. At the top level, we wrap `satisfy` with a function `evaluate` which handles the exception by returning the string `'Undefined'` as value, rather than allowing the exception to raise any higher.

EAFP stands for 'Easier to ask for forgiveness than permission'. According to van Rossum (2006), "this common Python coding style assumes the existence of valid keys or attributes and catches exceptions if the assumption proves false." It contrasts with LBYL ('Look before you leap'), which explicitly tests for pre-conditions (such as type checks) before making calls or lookups. To continue with the discussion of partiality, we can see an example of EAFP in the definition of the `i` function, which handles the interpretion of non-logical constants and individual variables.

```
try:
    return self.valuation[expr]
except Undefined:
    return g[expr]
```

We first try to evaluate `expr` as a non-logical constant; if `valuation` throws an `Undefined` exception, we check whether `g` can assign a value. If the latter also throws an `Undefined` excep-

[5] This is not the only approach, since one could adopt the position that a tautology such as $p \vee \neg p$ should be true even if p is undefined.

```
satisfy(Formula,model(D,F),G,pos):-
   nonvar(Formula),
   Formula = some(X,SubFormula),
   var(X),
   memberList(V,D),
   satisfy(SubFormula,model(D,F),[g(X,V)|G],pos).

satisfy(Formula,model(D,F),G,neg):-
   nonvar(Formula),
   Formula = some(X,SubFormula),
   var(X),
   setof(V,memberList(V,D),All),
   setof(V,
         (
          memberList(V,D),
          satisfy(SubFormula,model(D,F),[g(X,V)|G],neg)
         ),
         All).
```

Figure 3: Prolog Clauses for Existential Quantification

tion, this will automatically be raised to the calling function.

To sum up, an attractive consequence of this approach in Python is that no additional stipulations need to be added to the recursive clauses for interpreting Boolean connectives. By contrast, in the B&B `modelChecker2.pl` code, the clauses for existential quantification shown in Figure 3 need to be supplemented with a separate clause for the 'undefined' case. In addition, as remarked earlier, each Boolean connective receives an additional clause when undefined.

5 Specifying Models

Models are specified by instantiating the `Model` class. At initialization, two parameters are called, determining the model's domain and valuation function. In Table 4, we start by creating a `Valuation` object `val` (line 1), we then specify the valuation as a list `v` of *constant-value* pairs (lines 2–9), using relational notation. For example, the value for `'adam'` is the individual `'d1'` (i.e., a Python string); the value for `'girl'` is the set consisting of individuals `'g1'` and `'g1'`; and the value for `'love'` is a set of pairs, as described above. We use the `parse` method to update `val` with this information (line 10). As mentioned earlier, a `Valuation` object has a `domain` property (line 11), and in this case `dom` will evaluate to the set `set(['b1', 'b2', 'g1', 'g2', 'd1'])`. It is convenient to use this set as the value for the model's domain when it is initialized (line 12). We also declare an `Assignment` object (line 13), specifying that its domain is the same as the model's domain.

Given model `m` and assignment `g`, we can evaluate `m.satisfiers(formula, g)`, for various values of `formulas`. This is quite a handy way of getting a feel for how connectives and quantifiers interact. A range of cases is illustrated in Table 5. As pointed out earlier, all formulas are represented as Python strings, and therefore need to be parsed before being evaluated.

6 Conclusion

In this paper, I have tried to show how various aspects of Python lend themselves well to the task of interpreting first-order formulas, following closely in the footsteps of Blackburn and Bos. I argue that at least in some cases, the Python implementation compares quite favourably to a Prolog-based approach. It will be observed that I have not considered efficiency issues. Although these cannot be ignored (and are certainly worth exploring), they are not a priority at this stage of development. As discussed at the outset, our main goal is develop a framework that can be used to communicate key ideas of formal semantics to students, rather than to build systems which can scale easily to tackle large problems.

Clearly, there are many design choices to be made in any implementation, and an alternative framework which overlaps in part with what I have presented can be found in the Python code supplement to (Russell and Norvig, 2003).[6] One important distinction is that the approach adopted here

[6] `http://aima.cs.berkeley.edu/python`

```
val = Valuation()
v = [('adam', 'b1'), ('betty', 'g1'), ('fido', 'd1'),\
     ('girl', set(['g1', 'g2'])),\
     ('boy', set(['b1', 'b2'])),\
     ('dog', set(['d1'])),\
     ('love', set([('b1', 'g1'),\
                   ('b2', 'g2'),\
                   ('g1', 'b1'),\
                   ('g2', 'b1')]))]
val.parse(v)
dom = val.domain
m = Model(dom, val)
g = Assignment(dom, {'x': 'b1', 'y': 'g2'})
```

Figure 4: First-order model `m`

Formula open in x	Satisfiers
`'(boy x)'`	`set(['b1', 'b2'])`
`'(x = x)'`	`set(['b1', 'b2', 'g2', 'g1', 'd1'])`
`'((boy x) or (girl x))'`	`set(['b2', 'g2', 'g1', 'b1'])`
`'((boy x) and (girl x))'`	`set([])`
`'(love x adam)'`	`set(['g1'])`
`'(love adam x)'`	`set(['g2', 'g1'])`
`'(not (x = adam))'`	`set(['b2', 'g2', 'g1', 'd1'])`
`'some y.(love x y)'`	`set(['g2', 'g1', 'b1'])`
`'all y.((girl y) implies (love y x))'`	`set([])`
`'all y.((girl y) implies (love x y))'`	`set(['b1'])`
`'((girl x) implies (dog x))'`	`set(['b1', 'b2', 'd1'])`
`'all y.((dog y) implies (x = y))'`	`set(['d1'])`
`'(not some y.(love x y))'`	`set(['b2', 'd1'])`
`'some y.((love y adam) and (love x y))'`	`set(['b1'])`

Figure 5: Satisfiers in model `m`

is explicitly targeted at students learning computational linguistics, rather than being intended for a more general artificial intelligence audience.

While I have restricted attention to rather basic topics in semantic interpretation, there is no obstacle to addressing more sophisticated topics in computational semantics. For example, I have not tried to address the crucial issue of quantifier scope ambiguity. However, work by Peter Wang (author of the NLTK module `nltk_lite.contrib.hole`) implements the Hole Semantics of B&B. This module contains a 'plugging' algorithm which converts underspecified representations into fully-specified first-order logic formulas that can be displayed textually or graphically. In future work, we plan to extend the semantics package in various directions, in particular by adding some basic inferencing mechanisms to NLTK.

Acknowledgements

I am very grateful to Steven Bird, Patrick Blackburn, Alex Lascarides and three anonymous reviewers for helpful feedback and comments.

References

Steven Bird. 2005. NLTK-Lite: Efficient scripting for natural language processing. In *Proceedings of the 4th International Conference on Natural Language Processing (ICON)*, pages 11–18, New Delhi, December. Allied Publishers.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Publications.

D. R. Dowty, R. E. Wall, and S. Peters. 1981. *Introduction to Montague Semantics*. Studies in Linguistics and Philosophy. Reidel, Dordrecht.

Richard Montague. 1974. The proper treatment of quantification in ordinary English. In R. H. Thomason, editor, *Formal Philosphy: Selected Papers of Richard Montague*, pages 247–270. Yale University Press, New Haven.

Stuart Russell and Peter Norvig. 2003. *Artifical Intelligence: A Modern Approach*. Prentice Hall. 2nd edition.

Guido van Rossum. 2006. *Python Tutorial*. March. Release 2.4.3.

Classifying Speech Acts using Verbal Response Modes

Andrew Lampert †‡ **Robert Dale** ‡ **Cécile Paris** †

†CSIRO ICT Centre
Locked Bag 17, North Ryde
NSW 1670 Australia
firstname.lastname@csiro.au

‡Centre for Language Technology
Macquarie University
NSW 2109 Australia
[alampert,rdale]@ics.mq.edu.au

Abstract

The driving vision for our work is to provide intelligent, automated assistance to users in understanding the status of their email conversations. Our approach is to create tools that enable the detection and connection of speech acts across email messages. We thus require a mechanism for tagging email utterances with some indication of their dialogic function. However, existing dialog act taxonomies as used in computational linguistics tend to be too task- or application-specific for the wide range of acts we find represented in email conversation. The Verbal Response Modes (VRM) taxonomy of speech acts, widely applied for discourse analysis in linguistics and psychology, is distinguished from other speech act taxonomies by its construction from cross-cutting principles of classification, which ensure universal applicability across any domain of discourse. The taxonomy categorises on two dimensions, characterised as literal meaning and pragmatic meaning. In this paper, we describe a statistical classifier that automatically identifies the literal meaning category of utterances using the VRM classification. We achieve an accuracy of 60.8% using linguistic features derived from VRM's human annotation guidelines. Accuracy is improved to 79.8% using additional features.

1 Introduction

It is well documented in the literature that users are increasingly using email for managing requests and commitments in the workplace (Bellotti et al., 2003). It has also been widely reported that users commonly feel overloaded when managing multiple ongoing tasks through email communication e.g. (Whittaker and Sidner, 1996).

Given significant task-centred email usage, one approach to alleviating email overload in the workplace is to draw on Speech Act Theory (Searle, 1969) to analyse the intention behind email messages and use this information to help users process and prioritise their email. The basic tenet of Speech Act Theory is that when we utter something, we also act. Examples of such acts can include stating, questioning or advising.

The idea of identifying and exploiting patterns of communicative acts in conversations is not new. Two decades ago, Flores and Winograd (1986) proposed that workplace workflow could be seen as a process of creating and maintaining networks of conversations in which requests and commitments lead to successful completion of work.

Recently, these ideas have begun to be applied to email messages. Existing work analysing speech acts in email messages differs as to whether speech acts should be annotated at the message level, e.g., (Cohen et al., 2004; Leuski, 2004), or at the utterance or sentence level, e.g., (Corston-Oliver et al., 2004). Our thesis is that a single email message may contain multiple commitments on a range of tasks, and so our work focuses on utterance-level classification, with the aim of being able to connect together the rich tapestry of threads that connect individual email messages.

Verbal Response Modes (VRM) (Stiles, 1992) is a principled taxonomy of speech acts for classifying the literal and pragmatic meaning of utterances. The hypothesis we pose in this work is that VRM annotation can be learned to create a classi-

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 34–41.

fier of literal utterance meaning.

The driving vision for our work is to eventually provide intelligent, automated assistance to email users in understanding the status of their current email conversations and tasks. We wish to assist users to identify outstanding tasks easily (both for themselves and their correspondents) through automatically flagging incomplete conversations, such as requests or commitments that remain unfulfilled. This capability should lead to novel forms of conversation-based search, summarisation and navigation for collections of email messages and for other textual, computer-mediated conversations. The work described here represents our first steps towards this vision.

This paper is structured as follows. First, in Section 2, we describe related work on automatically classifying speech and dialogue acts. In Section 3 we introduce the VRM taxonomy, comparing and contrasting it with other speech act taxonomies in Section 4. Then, in Section 5, we describe our statistical VRM classifier, and in Section 6 we present what we believe are the first results in the field for automatic VRM classification of the literal meaning of utterances. Finally, in Section 7 we discuss our results. Section 8 presents some concluding remarks and pointers to future work.

2 Related Work

There is much existing work that explores automated processing of speech and dialogue acts. This collection of work has predominantly focused around two related problems: *dialogue act prediction* and *dialogue act recognition*. Our work focuses on the second problem, and more specifically on speech act recognition.

Examples of dialogue act recognition include work by Core (1998) which uses previous and current utterance information to predict possible annotations from the DAMSL scheme (Core and Allen, 1997). Similar work by Chu-Carroll (1998) on statistical "discourse act" recognition also uses features from the current utterance and discourse history to achieve accuracy of around 51% for a set of 15 discourse acts. In particular, Chu-Carroll's results were significantly improved by taking into account the syntactic form of each utterance.

The use of n-gram language models is also a popular approach. Reithinger and Klesen (1997) apply n-gram language models to the VERBMOBIL corpus (Alexandersson et al., 1998) and report tagging accuracy of 74.7% for a set of 18 dialogue acts. In common with our own work, Webb et al. (2005) approach dialogue act classification using only intra-utterance features. They found that using only features derived from n-gram cue phrases performed moderately well on the SWITCHBOARD corpus of spoken dialogue (Godfrey et al., 1992).

To our knowledge, however, there has been no previous work that attempts to identify VRM categories for utterances automatically.

3 Verbal Response Modes

Verbal Response Modes (VRM) is a principled taxonomy of speech acts that can be used to classify literal and pragmatic meaning within utterances. Each utterance is coded twice: once for its *literal meaning*, and once for its communicative intent or *pragmatic meaning*. The same VRM categories are used in each case.

Under the VRM system, every utterance from a speaker can be considered to concern either the *speaker's* or the *other's* experience. For example, in the utterance "I like pragmatics.", the *source of experience* is the speaker. In contrast, the source of experience for the utterance "Do you like pragmatics?" is the other interlocutor.

Further, in making an utterance, the speaker may need to make presumptions about experience. For example, in saying "Do you like pragmatics?", the speaker does not need to presume to know what the other person is, was, will be, or should be thinking, feeling, perceiving or intending. Such utterances require a *presumption of experience* of the speaker only. In contrast, the utterance "Like pragmatics!" attempts to impose an experience (a liking for pragmatics) on the other interlocutor, and has a presumption of experience for the other.

Finally a speaker may represent the experience either from their own personal point of view, or from a viewpoint that is shared or held in common with the other interlocutor. The three example utterances above all use the speaker's *frame of reference* because the experience is understood from the speaker's point of view. In contrast, the utterance "You like pragmatics." takes the other's frame of reference, representing the experience as the other interlocutor views it.

These three principles — *source of experience*, *presumption about experience* and *frame of reference* — form the basis of the VRM taxonomy.

The principles are dichotomous — each can take the value of *speaker* or *other* (other interlocutor) — and thus define eight mutually exclusive VRM categories, as shown in Table 1.

With 8 VRM modes and 2 separate dimensions of coding (literal and pragmatic meaning), there are 64 possible form-intent combinations. The 8 combinations of codings in which the literal and pragmatic meanings coincide are referred to as *pure modes*. The other 56 modes are labelled *mixed modes*. An example of a mixed-mode utterance is "Can you pass the sugar?" which is coded QA. This is read *Question in service of Advisement*, meaning that the utterance has a Question form (literal meaning) but Advisement intent (pragmatic meaning). In this way, the VRM taxonomy is designed to simply and consistently classify and distinguish direct and indirect speech acts.

4 Comparison with Other Speech Act Taxonomies

There are, of course, alternate speech and dialogue act taxonomies, some of which have been applied within natural language processing applications. Unfortunately, many of these taxonomies tend to offer competing, rather than complementary approaches to classifying speech acts, making it difficult to compare experimental results and analyses that are based on different taxonomies. It would clearly be desirable to unambiguously relate categories between different taxonomies.

One specific drawback of many speech and dialogue act taxonomies, including taxonomies such as those developed in the VERBMOBIL project (Alexandersson et al., 1998), is that they are domain or application specific in their definition and coverage of speech act categories. This often stems from the taxonomy being developed and used in a rather *ad hoc*, empirical manner for analysing discourse and utterances from a single or small set of application domains.

While the VRM research grew from studying therapist interventions in psychotherapy (Stiles, 1992; Wiser and Goldfried, 1996), the VRM system has been applied to a variety of discourse genres. These include: American Presidential speeches (Stiles et al., 1983), doctor-patient interactions (Meeuswesen et al., 1991), courtroom interrogations (McGaughey and Stiles, 1983), business negotiations (Ulijn and Verweij, 2000), persuasive discourse (Kline et al., 1990) and television commercials (Rak and McMullen, 1987). VRM coding assumes only that there is a *speaker* and an intended audience (*other*), and thus can be applied to any domain of discourse.

The wide applicability of VRM is also due to its basis of clearly defined, domain-independent, systematic principles of classification. This ensures that the VRM categories are both extensive and exhaustive, meaning that all utterances can be meaningfully classified with exactly one VRM category[1]. In contrast, even widely-applied, complex taxonomies such as DAMSL (Core and Allen, 1997) resort to the inclusion of an *other* category within the speech act component, to be able to classify utterances across domains.

In addition, the VRM principles facilitate more rigorous and comparable coding of utterances from which higher-level discourse properties can be reliably calculated, including characterisation of the roles played by discourse participants. If required, the eight VRM modes can also be further divided to identify additional features of interest (for example, the Question category could be split to distinguish open and closed questions). Importantly, this can be done within the existing framework of categories, without losing the principled basis of classification, or the ability to compare directly with other VRM analyses.

Table 2 compares the VRM categories with Searle's five major speech act categories (1969; 1979). Searle's categories are largely subsumed under the subset of VRM categories that offer the *speaker's* source of experience and/or frame of reference (Disclosure, Edifications, Advisements and Questions). The coverage of Searle's speech acts seems more limited, given that the other VRMs (Reflection, Interpretation, Confirmation and Acknowledgement), all *other* on at least two principles, have no direct equivalents in Searle's system, except for some Interpretations which might map to specific subcategories of Declaration.

5 Building a VRM Classifier

As discussed earlier, the VRM system codes both the literal and pragmatic meaning of utterances. The pragmatic meaning conveys the speaker's actual intention, and such meaning is often hidden or

[1]The only exceptions are utterances that are inaudible or incomprehensible in spoken dialogue, which are coded Uncodable (U).

Source of Experience	*Presumption about Experience*	*Frame of Reference*	*VRM Mode*	*Description*
Speaker	Speaker	Speaker	Disclosure (D)	Reveals thoughts, feelings, perceptions or intentions. E.g., *I like pragmatics.*
		Other	Edification (E)	States objective information. E.g., *He hates pragmatics.*
	Other	Speaker	Advisement (A)	Attempts to guide behaviour; suggestions, commands, permission, prohibition. E.g., *Study pragmatics!*
		Other	Confirmation (C)	Compares speaker's experience with other's; agreement, disagreement, shared experience or belief. E.g., *We both like pragmatics.*
Other	Speaker	Speaker	Question (Q)	Requests information or guidance. E.g., *Do you like pragmatics?*
		Other	Acknowledgement (K)	Conveys receipt of or receptiveness to other's communication; simple acceptance, salutations. E.g., *Yes.*
	Other	Speaker	Interpretation (I)	Explains or labels the other; judgements or evaluations of the other's experience or behaviour. E.g., *You're a good student.*
		Other	Reflection (R)	Puts other's experience into words; repetitions, restatements, clarifications. E.g., *You dislike pragmatics.*

Table 1: The Taxonomy of Verbal Response Modes from (Stiles, 1992)

Searle's Classification	*Corresponding VRM*
Commissive	Disclosure
Expressive	Disclosure
Representative	Edification
Directive	Advisement; Question
Declaration	Interpretation; Disclosure; Edification

Table 2: A comparison of VRM categories with Searle's speech acts

highly dependent on discourse context and background knowledge. Because we classify utterances using only intra-utterance features, we cannot currently encode any information about the discourse context, so could not yet plausibly tackle the prediction of pragmatic meaning. Discerning literal meaning, while somewhat simpler, is akin to classifying direct speech acts and is widely recognised as a challenging computational task.

5.1 Corpus of VRM Annotated Utterances

Included with the VRM coding manual (Stiles, 1992) is a VRM coder training application for training human annotators. This software, which is freely available online[2], includes transcripts of spoken dialogues from various domains segmented into utterances, with each utterance annotated with two VRM categories that classify both its literal and pragmatic meaning.

These transcripts were pre-processed to remove instructional text and parenthetical text that was not actually part of a spoken and coded utter-

[2]The VRM coder training application and its data files are available to download from http://www.users.muohio.edu/stileswb/archive.htmlx

ance. Several additional example utterances were extracted from the coding manual to increase the number of instances of under-represented VRM categories (notably *Confirmations* and *Interpretations*).

The final corpus contained 1368 annotated utterances from 14 dialogues and several sets of isolated utterances. Table 3 shows the frequency of each VRM mode in the corpus.

VRM	*Instances*	*Percentage*
Disclosure	395	28.9%
Edification	391	28.6%
Advisement	73	5.3%
Confirmation	21	1.5%
Question	218	15.9%
Acknowledgement	97	7.1%
Interpretation	64	4.7%
Reflection	109	8.0%

Table 3: The distribution of VRMs in the corpus

5.2 Features for Classification

The VRM annotation guide provides detailed instructions to guide humans in correctly classifying the literal meaning of utterances. These suggested features are shown in Table 4.

We have attempted to map these features to computable features for training our statistical VRM classifier. Our resulting set of features is shown in Table 5 and includes several additional features not identified by Stiles that we use to further characterise utterances. These additional features include:

- **Utterance Length:** The number of words in the utterance.
- **First Word:** The first word in each utterance, represented as a series of independent boolean features (one for each unique first word present in the corpus).
- **Last Token:** The last token in each utterance – either the final punctuation (if present) or the final word in the utterance. As for the First Word features, these are represented as a series of independent boolean features.
- **Bigrams:** Bigrams extracted from each utterance, with a variable threshold for including only frequent bigrams (above a specified threshold) in the final feature set.

VRM Category	*Form Criteria*
Disclosure	Declarative; 1st person singular or plural where *other* is not a referent.
Edification	Declarative; 3rd person.
Advisement	Imperative or 2nd person with verb of permission, prohibition or obligation.
Confirmation	1st person plural where referent includes the other (i.e., "we" refers to both *speaker* and *other*).
Question	Interrogative, with inverted subject-verb order or interrogative words.
Acknowledgement	Non-lexical or contentless utterances; terms of address or salutation.
Interpretation	2nd person; verb implies an attribute or ability of the other; terms of evaluation.
Reflection	2nd person; verb implies internal experience or volitional action.

Table 4: VRM form criteria from (Stiles, 1992)

The intuition for including the utterance length as a feature is that different VRMs are often associated with longer or shorter utterances - e.g., Acknowledgement utterances are often short, while Edifications are often longer.

To compute our utterance features, we made use of the Connexor Functional Dependency Grammar (FDG) parser (Tapanainen and Jarvinen, 1997) for grammatical analysis and to extract syntactic dependency information for the words in each utterance. We also used the morphological tags assigned by Connexor. This information was used to calculate utterance features as follows:

- **Functional Dependencies:** Dependency functions were used to identify main subjects and main verbs within utterances, as required for features including the 1st/2nd/3rd person subject, inverted subject-verb order and imperative verbs.
- **Syntactic Functions:** Syntactic function in-

formation was determined using the Connexor parser. This information was used to identify the main utterance subject where dependency information was not available.

- **Morphology:** Morphological tags, also generated by Connexor, were used to distinguish between first and third person pronouns, as well as between singular and plural forms of first person pronouns. Additionally, we used morphological tags from Connexor to identify imperative verbs.

- **Hand-constructed word lists:** Several of the features used relate to closed sets of common lexical items (e.g., verbs of permission, interrogative words, variations of "yes" and "no"). For these features, we employ hand-constructed simple lists, using online thesauri to expand our lists from an initial set of seed words. While some of the lists are not exhaustive, they seem to help our results and involved only a small amount of effort; none took more than an hour to construct.

Feature	*Likely VRM*
1st person singular subject	D,Q
1st person plural singular subject	D,C
3rd person subject	E,Q
2nd person subject	A,Q,I,R
Inverted subject-verb order	Q
Imperative verb	A
Verbs of permission, prohibition, obligation	A
Interrogative words	Q
Non-lexical content	K
Yes/No variants	K
Terms of evaluation	I
Utterance length	all
First word	all
Last token	all
Bi-grams	all

Table 5: Features used in VRM Classifier

6 Results

Our classification results using several different learning algorithms and variations in feature sets are summarised in Table 6. We experimented with using only the linguistic features suggested by Stiles, using only the additional features we identified, and using a combination of all features shown in Table 5. All our results were validated using stratified 10-fold cross validation.

We used supervised learning methods implemented in Weka (Witten and Frank, 2005) to train our classifier. Through experimentation, we found that Weka's Support Vector Machine implementation (SMO) provided the best classification performance. Encouragingly, other relatively simple approaches, such as a Bayesian Network classifier using the K2 hill-climbing search algorithm, also performed reasonably well.

The baseline against which we compare our classifier's performance is a OneR (one rule) classifier using an identical feature set. This baseline system is a one-level decision tree, (i.e., based on a set of rules that test only the single most discriminative feature). As shown in Table 6, the accuracy of this baseline varies from 42.76% to 49.27%, depending on the exact features used. Regardless of features or algorithms, our classifier performs significantly better than the baseline system.

Algorithm	*Feature Set*	*Accuracy*	*Mean Abs Error*
SVM	All	79.75%	0.19
SVM	Only Stiles'	60.82%	0.20
SVM	No Stiles'	74.49%	0.19
Bayes Net	All	78.51%	0.06
Bayes Net	Only Stiles'	60.16%	0.12
Bayes Net	No Stiles'	75.68%	0.07
Baseline	All	49.27%	0.36
Baseline	Only Stiles'	49.27%	0.36
Baseline	No Stiles'	42.76%	0.38

Table 6: VRM classifier results

Another tunable parameter was the level of pruning of n-grams from our feature set according to their frequency of occurrence. Heuristically, we determined that a cut-off of 5 (i.e., only n-grams that occur five or more times in our corpus of utterances were included as features) gave us the highest accuracy for the learning algorithms tested.

7 Discussion

This work appears to be the first attempt to automatically classify utterances according to their literal meaning with VRM categories. There are thus no direct comparisons to be easily drawn for

our results. In classifying only the literal meaning of utterances, we have focused on a simpler task than classifying in-context meaning of utterances which some systems attempt.

Our results do, however, compare favourably with previous dialogue act classification work, and clearly validate our hypothesis that VRM annotation can be learned. Previous dialogue act classification results include Webb et al. (2005) who reported peak accuracy of around 71% with a variety of n-gram, word position and utterance length information on the SWITCHBOARD corpus using the 42-act DAMSL-SWBD taxonomy. Earlier work by Stolcke et al. (2000) obtained similar results using a more sophisticated combination of hidden markov models and n-gram language models with the same taxonomy on the same corpus. Reithinger and Klesen (1997) report a tagging accuracy of 74.7% for a set of 18 dialogue acts over the much larger VERBMOBIL corpus (more than 223,000 utterances, compared with only 1368 utterances in our own corpus).

VRM	*Instances*	*Precision*	*Recall*
D	395	0.905	0.848
E	391	0.808	0.872
A	73	0.701	0.644
C	21	0.533	0.762
Q	218	0.839	0.885
K	97	0.740	0.763
I	64	0.537	0.453
R	109	0.589	0.514

Table 7: Precision and recall for each VRM

In performing an error analysis of our results, we see that classification accuracy for Interpretations and Reflections is lower than for other classes, as shown in Table 7. In particular, our confusion matrix shows a substantial number of transposed classifications between these two VRMs. Interestingly, Stiles makes note that these two VRMs are very similar, differing only on one principle (frame of reference), and that they are often difficult to distinguish in practice. Additionally, some Reflections repeat all or part of the *other's* utterance, or finish the *other's* previous sentence. It is impossible for our current classifier to detect such phenomena, since it looks at utterances in isolation, not in the context of a larger discourse. We plan to address this in future work.

Our results also provide support for using both linguistic and statistical features in classifying VRMs. In the cases where our feature set consists of only the linguistic features identified by Stiles, our results are substantially worse. Similarly, when only n-gram, word position and utterance length features are used, classifier performance also suffers. Table 6 shows that our best results are obtained when both types of features are included.

Finally, another clear trend in the performance of our classifier is that the VRMs for which we have more utterance data are classified substantially more accurately.

8 Conclusion

Supporting the hypothesis posed, our results suggest that classifying utterances using Verbal Response Modes is a plausible approach to computationally identifying literal meaning. This is a promising result that supports our intention to apply VRM classification as part of our longer-term aim to construct an application that exploits speech act connections across email messages.

While difficult to compare directly, our classification accuracy of 79.75% is clearly competitive with previous speech and dialogue act classification work. This is particularly encouraging considering that utterances are currently being classified in isolation, without any regard for the discourse context in which they occur.

In future work we plan to apply our classifier to email, exploiting features of email messages such as header information in the process. We also plan to incorporate discourse context features into our classification and to explore the classification of pragmatic utterance meanings.

References

Jan Alexandersson, Bianka Buschbeck-Wolf, Tsutomu Fujinami, Michael Kipp, Stephan Koch, Elisabeth Maier, Norbert Reithinger, Birte Schmitz, and Melanie Siegel. 1998. Dialogue acts in VERBMOBIL-2. Technical Report Verbmobil-Report 226, DFKI Saarbruecken, Universitt Stuttgart, Technische Universitt Berlin, Universitt des Saarlandes. 2nd Edition.

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, Ft Lauderdale, Florida, USA, April 5-10.

Jennifer Chu-Carroll, 1998. *Applying Machine Learning to Discourse Processing. Papers from the 1998 AAAI Spring Symposium.*, chapter A statistical model for discourse act recognition in dialogue interactions, pages 12–17. The AAAI Press, Menlo Park, California.

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. Association for Computational Linguistics.

Mark Core and James Allen. 1997. Coding dialogs with the DAMSL annotation scheme. In *AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Cambridge, MA, November.

Mark Core. 1998. Predicting DAMSL utterance tags. In *Proceedings of the AAAI-98 Spring Symposium on Applying Machine Learning to Discourse Processing*.

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*, July.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 517–520, San Francisco, CA, March.

Susan L Kline, C L Hennen, and K M Farrell. 1990. Cognitive complexity and verbal response mode use in discussion. *Communication Quarterly*, 38:350–360.

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*, Sheffi eld, UK.

Karen J McGaughey and William B Stiles. 1983. Courtroom interrogation of rape victims: Verbal response mode use by attourneys and witnesses during direct examination vs. cross-examination. *Journal of Applied Social Psychology*, 13:78–87.

Ludwein Meeuswesen, Cas Schaap, and Cees van der Staak. 1991. Verbal analysis of doctor-patient communication. *Social Science and Medicine*, 32(10):1143–50.

Diana S Rak and Linda M McMullen. 1987. Sex-role stereotyping in television commercials: A verbal response mode and content analysis. *Canadian Journal of Behavioural Science*, 19:25–39.

Norbert Reithinger and Martin Klesen. 1997. Dialogue act classifi cation using language models. In *Proceedings of Eurospeech ’97*, pages 2235–2238, Rhodes, Greece.

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

John R Searle. 1979. *Expression and Meaning*. Cambridge University Press.

William B Stiles, Melinda L Au, Mary Ann Martello, and Julia A Perlmutter. 1983. American campaign oratory: Verbal response mode use by candidates in the 1980 american presidential primaries. *Social Behaviour and Personality*, 11:39–43.

William B Stiles. 1992. *Describing Talk: a taxonomy of verbal response modes*. SAGE Series in Interpersonal Communication. SAGE Publications. ISBN: 0803944659.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Marie Meteer, and Carol Van Ess-Dykema. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–371.

Pasi Tapanainen and Timo Jarvinen. 1997. A nonprojective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington D.C. Association for Computational Linguistics.

Jan M Ulijn and Maurits J Verweij. 2000. Question behaviour in monocultural and intercultural business negotiations: the Dutch-Spanish connection. *Discourse Studies*, 2(1):141–172.

Nick Webb, Mark Hepple, and Yorick Wilks. 2005. Dialogue act classifi cation based on intra-utterance features. In *Proceedings of the AAAI Workshop on Spoken Language Understanding*.

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.

Terry Winograd and Fernando Flores. 1986. *Understanding Computers and Cognition*. Ablex Publishing Corporation, Norwood, New Jersey, USA, 1st edition. ISBN: 0-89391-050-3.

Susan Wiser and Marvin R Goldfried. 1996. Verbal interventions in signifi cant psychodynamic-interpersonal and cognitive-behavioral therapy sessions. *Psychotherapy Research*, 6(4):309–319.

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

Word Relatives in Context for Word Sense Disambiguation

David Martinez
Computer Science and
Sofware Engineering
University of Melbourne
Victoria 3010 Australia
davidm@csse.unimelb.edu.au

Eneko Agirre
IXA NLP Group
Univ. of the Basque Country
Donostia, Basque Country
e.agirre@ehu.es

Xinglong Wang
School of Informatics
University of Edinburgh
EH8 9LW, Edinburgh, UK
xwang@inf.ed.ac.uk

Abstract

The current situation for Word Sense Disambiguation (WSD) is somewhat stuck due to lack of training data. We present in this paper a novel disambiguation algorithm that improves previous systems based on acquisition of examples by incorporating local context information. With a basic configuration, our method is able to obtain state-of-the-art performance. We complemented this work by evaluating other well-known methods in the same dataset, and analysing the comparative results per word. We observed that each algorithm performed better for different types of words, and each of them failed for some particular words. We proposed then a simple unsupervised voting scheme that improved significantly over single systems, achieving the best unsupervised performance on both the Senseval 2 and Senseval 3 lexical sample datasets.

1 Introduction

Word Sense Disambiguation (WSD) is an intermediate task that potentially can benefit many other NLP systems, from machine translation to indexing of biomedical texts. The goal of WSD is to ground the meaning of words in certain contexts into concepts as defined in some dictionary or lexical repository.

Since 1998, the Senseval challenges have been serving as showcases for the state-of-the-art WSD systems. In each competition, Senseval has been growing in participants, labelling tasks, and target languages. The most recent Senseval workshop (Mihalcea et al., 2004) has again shown clear superiority in performance of supervised systems, which rely on hand-tagged data, over other kinds of techniques (knowledge-based and unsupervised). However, supervised systems use large amounts of accurately sense-annotated data to yield good results, and such resources are very costly to produce and adapt for specific domains. This is the so-called knowledge acquisition bottleneck, and it has to be tackled in order to produce technology that can be integrated in real applications. The challenge is to make systems that disambiguate all the words in the context, as opposed to techniques that work for a handful of words.

As shown in the all-words tasks in Senseval-3 (B. Snyder and M. Palmer, 2004), the current WSD techniques are only able to exceed the most frequent sense baseline by a small margin. We believe the main reason for that is the lack of large amounts of training material for English words (not to mention words in other languages). Unfortunately developing such resources is difficult and sometimes not feasible, which has been motivating us to explore unsupervised techniques to open up the knowledge acquisition bottleneck in WSD.

The unsupervised systems that we will apply on this paper require raw corpora and a thesaurus with relations between word senses and words. Although these resources are not available for all languages, there is a growing number of WordNets in different languages that can be used[1]. Other approach would be to apply methods based on distributional similarity to build a thesaurus automatically from raw corpora (Lin, 1998). The relations can then be applied in our algorithm. In this paper we have focused on the results we can obtain for

[1] http://www.globalwordnet.org/gwa/wordnet_table.htm

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 42–50.

English, relying on WordNet as thesaurus (Fellbaum, 1998).

A well known approach for unsupervised WSD consists of the automatic acquisition of training data by means of monosemous relatives (Leacock et al., 1998). This technique roughly follows these steps: (i) select a set of monosemous words that are related to the different senses of the target word, (ii) query the Internet to obtain examples for each relative, (iii) create a collection of training examples for each sense, and (iv) use an ML algorithm trained on the acquired collections to tag the test instances. This method has been used to bootstrap large sense-tagged corpora (Mihalcea, 2002; Agirre and Martinez, 2004).

Two important shortcomings of this method are the lack of monosemous relatives for some senses of the target words, and the noise introduced by some distant relatives. In this paper we directly address those problems by developing a new method that makes use of polysemous relatives and relies on the context of the target word to reduce the presence of noisy examples.

The remaining of the paper is organised as follows. In Section 2 we describe related work in this area. Section 3 briefly introduces the monosemous relatives algorithm, and our novel method is explained in Section 4. Section 5 presents our experimental setting, and in Section 6 we report the performance of our technique and the improvement over the monosemous relatives method. Section 7 is devoted to compare our system to other unsupervised techniques and analyse the prospects for system combination. Finally, we conclude and discuss future work in Section 8.

2 Related Work

The construction of unsupervised WSD systems applicable to all words in context has been the goal of many research initiatives, as can be seen in special journals and devoted books - see for instance (Agirre and Edmonds, 2006) for a recent book. We will now describe different trends that are being explored.

Some recent techniques seek to alleviate the knowledge acquisition bottleneck by combining training data from different words. Kohomban and Lee (2005) build semantic classifiers by merging data from words in the same semantic class. Once the class is selected, simple heuristics are applied to obtain the fine-grained sense. The classifier follows memory-based learning, and the examples are weighted according to their semantic similarity to the target word. Niu et al. (2005) use all-words training data to build a word-independent model to compute the similarity between two contexts. A maximum entropy algorithm is trained with the all-words corpus, and the model is used for clustering the instances of a given target word. One of the problems of clustering algorithms for WSD is evaluation, and in this case they map the clusters to Senseval-3 lexical-sample data by looking at 10% of the examples in training data. One of the drawbacks of these systems is that they still require hand-tagged data.

Parallel corpora have also been widely used to avoid the need of hand-tagged data. Recently Chan and Ng (2005) built a classifier from English-Chinese parallel corpora. They grouped senses that share the same Chinese translation, and then the occurrences of the word on the English side of the parallel corpora were considered to have been disambiguated and "sense tagged" by the appropriate Chinese translations. The system was successfully evaluated in the all-words task of Senseval-2. However, parallel corpora is an expensive resource to obtain for all target words. A related approach is to use monolingual corpora in a second language and use bilingual dictionaries to translate the training data (Wang and Carroll, 2005). Instead of using bilingual dictionaries, Wang and Martinez (2006) tried to apply machine translation on translating text snippets in foreign languages back into English and achieved good results on English WSD.

Regarding portability, methods to automatically rank the senses of a word given a raw corpus, such as (McCarthy et al., 2004), have shown good flexibility to adapt to different domains, which is a desirable feature of all-words systems. We will compare the performance of the latter two systems and our approach in Section 7.

3 Monosemous Relatives method

The "monosemous relatives" approach is a technique to acquire training examples automatically and then feed them to a Machine Learning (ML) method. This algorithm is based on (Leacock et al., 1998), and follows these steps: (i) select a set of monosemous words that are related to the different senses of the target word, (ii) query the Internet to obtain examples for each relative, (iii)

create a collection of training examples for each sense, and (iv) use an ML algorithm trained on the acquired collections to tag the test instances. This method has been applied in different works (Mihalcea, 2002; Agirre and Martinez, 2004). We describe here the approach by Agirre and Martinez (2004), which we will apply to the same datasets as the novel method described in Section 4.

In this implementation, the monosemous relatives are obtained using WordNet, and different relevance weights are assigned to these words depending on the distance to the target word (synonyms are the closest, followed by immediate hypernyms and hyponyms). These weights are used to determine an order of preference to construct the training corpus from the queries, and 1,000 examples are then retrieved for each query. As explained in (Agirre and Martinez, 2004), the number of examples taken for each sense has a big impact in the performance, and information on the expected distribution of senses influences the results. They obtain this information using different means, such as hand-tagged data distribution (from Semcor), or a prior algorithm like (McCarthy et al., 2004). In this paper we present the results of the basic approach that uses all the retrieved examples per sense, which is the best standalone unsupervised alternative.

The ML technique Agirre and Martinez (2004) applied is Decision Lists (Yarowsky, 1994). In this method, the sense s_k with the highest weighted feature f_i is selected, according to its log-likelihood (see Formula 1). For this implementation, they used a simple smoothing method: the cases where the denominator is zero are smoothed by the constant 0.1.

$$weight(s_k, f_i) = \log(\frac{Pr(s_k|f_i)}{\sum_{j \neq k} Pr(s_j|f_i)}) \quad (1)$$

The feature set consisted of local collocations (bigrams and trigrams), bag-of-words features (unigrams and salient bigrams), and domain features from the WordNet Domains resource (Magnini and Cavagliá, 2000). The Decision List algorithm showed good comparative performance with the monosemous relatives method, and it had the advantage of allowing hands-on analysis of the different features.

4 Relatives in Context

The goal of this new approach is to use the WordNet relatives and the contexts of the target words to overcome some of the limitations found in the "monosemous relatives" technique. One of the main problems is the lack of close monosemous relatives for some senses of the target word. This forces the system to rely on distant relatives whose meaning is far from the intended one. Another problem is that by querying only with the relative word we do not put any restrictions on the sentences we retrieve. Even if we are using words that are listed as monosemous in WordNet, we can find different usages of them in a big corpus such as Internet (e.g. Named Entities, see example below). Including real contexts of the target word in the queries could alleviate the problem.

For instance, let us assume that we want to classify *church* with one of the 3 senses it has in Senseval-2: (1) Group of Christians, (2) Church building, or (3) Church service. When querying the Internet directly with monosemous relatives of these senses, we find the following problems:

- Metaphors: the relative *cathedral* (2nd sense) appears in very different collocations that are not related to any sense of *church*, e.g. *the cathedral of football*.
- Named entities: the relative *kirk* (2nd sense), which is a name for a Scottish church, will retrieve sentences that use Kirk as a proper noun.
- Frequent words as relatives: relatives like *hebraism* (1st sense) could provide useful examples, but if the query is not restricted can also be the source of many noisy examples.

The idea behind the "relatives in context" method is to combine local contexts of the target word with the pool of relatives in order to obtain a better set of examples per sense. Using this approach, we only gather those examples that have a close similarity with the target contexts, defined by a set of pre-defined features. We will illustrate this with the following example from the Senseval-2 dataset, where the goal is to disambiguate the word *church*:

> *The **church** was rebuilt in the 13th century and further modifications and restoration were carried out in the 15th century.*

We can extract different features from this context, for instance using a dependency parser. We can obtain that there is a object-verb relation between *church* and *rebuild*. Then we can incorporate this knowledge to the relative-based query and obtain training examples that are closer to our target sentence. In order to implement this approach with rich features we require tools that allow for linguistic queries, such as the linguist's engine (Resnik and Elkiss, 2005), but other approach would be to use simple features, such as strings of words, in order to benefit directly from the examples coming from search engines in the Internet. In this paper we decided to explore the latter technique to observe the performance we can achieve with simple features. Thus, in the example above, we query the Internet with snippets such as "The *cathedral* was rebuilt" to retrieve training examples. We will go back to the example at the end of this section.

With this method we can obtain a separate training set starting from each test instance and the pool of relatives for each sense. Then, a ML algorithm can be trained with the acquired examples. Alternatively, we can just rank the different queries according to the following factors:

- Length of the query: the longer the match, the more similar the new sentence will be to the target.
- Distance of the relative to the target word: examples that are obtained with synonyms will normally be closer to the original meaning.
- Number of hits: the more common the snippet we query, the more reliable.

We observed a similar performance in preliminary experiments when using a ML method or applying an heuristic on the above factors. For this paper we devised a simple algorithm to rank queries according to the three factors, but we plan to apply other techniques in the acquired training data in the future.

Thus, we build a disambiguation algorithm that can be explained in the following four steps:

1. Obtain pool of relatives: for each sense of the target word we gather its synonyms, hyponyms, and hypernyms. We also take polysemous nouns, as we expect that in similar local contexts the relative will keep its related meaning.

2. Construct queries: first we tokenise each target sentence, then we apply sliding windows of different sizes (up to 6 tokens) that include the target word. For each window and each relative in the pool, we substitute the target word for the relative and query the Internet. Then we store the number of hits for each query. The algorithm stops augmenting the window for the relative when one of its substrings returns zero hits.

3. Ranking of queries: we devised a simple heuristic to rank the queries according to our intuition on the relevant parameters. We chose these three factors (in decreasing order of relevance):

- Number of tokens of the query.
- Type of relative: preference order: (1) synonyms, (2) immediate hyponyms, (3) immediate hypernyms, and (4) distant relatives.
- Number of hits: we choose the query with most hits. For normalisation we divide by the number of hits of the relative alone, which penalises frequent and polysemous relatives.

We plan to improve this ranking approach in the future, by learning the best parameter set on a development corpus. We also would like to gather a training corpus from the returned documents and apply a ML classifier.

4. Assign the sense of the highest ranked query: another alternative that we will explore in the future is to vote among the k highest ranked queries.

We will show how the algorithm works with the example for the target word *church* presented above. Using the relatives (synonyms, hypernyms, and hyponyms) of each sense and the local context we query the Internet. The list of the longest matches that have at least 2 hits is given in Table 1. In this case the second sense would be chosen because the words *nave*, *abbey*, and *cathedral* indicate this sense. In cases where the longest match corresponds to more than one sense the closest relative is chosen; if there is still a tie the number of hits (divided by the number of hits of the relative for normalisation) is used.

5 Experimental setting

For our experiments we relied on the lexical-sample datasets of both Senseval-2 (Kilgarriff, 2001) and Senseval-3 (Mihalcea et al., 2004). We

Query	Sense
The *nave* was rebuilt in the 13th century	2
The *abbey* was rebuilt in the 13th century	2
The *cathedral* was rebuilt in the 13th century	2
The *Catholic Church* was rebuilt in	1
The *Christian church* was rebuilt	1
The *church service* was	3
The *religious service* was	3

Table 1: Longest matches for relative words of *church* in the Senseval-2 example "*The **church** was rebuilt in the 13th century and further modifications and restoration were carried out in the 15th century.*".

will refer to these sets as S2LS and S3LS respectively. This approach will give us the chance to measure the performance on different sets of words, and compare our results to the state of the art. We will focus on nouns in this work, in order to better study the specific problems to be analysed in the error analysis. The test sets consist on 29 nouns in S2LS, and 20 nouns in S3LS. The sense inventory in S2LS corresponds to WordNet 1.7 (pre-release), while for S3LS the senses belong to WordNet 1.7.1.

Our main goal is to build all-words WSD systems, and this preliminary test on lexical-sample datasets will give us a better idea of the performance we can expect. The same algorithms can be used for extending the evaluation to all the words in context by considering each target word separately. We plan to carry out this evaluation in the near future.

Regarding evaluation, we used the scoring software provided by the Senseval organisation to measure the precision and recall of the systems. Precision refers to the ratio of correct answers to the total number of answers given by the system, and recall indicates the ratio of correct answers to the total number of instances. All our algorithms have full coverage (that is, they always provide an answer), and therefore precision equals recall. In some cases we may present the results per sense, and then the precision will refer to the ratio of correct answers to the number of answers given to the sense; recall will be the ratio of correct answers to the number of test instances linked to the sense.

6 Results

The results of applying the "monosemous relatives" (MR) and the "relatives in context" (RC) algorithm are shown in Table 2. The micro-averaged

S2LS			S3LS		
Word	MR	RC	Word	MR	RC
art	**61.1**	40.3	argument	24.7	**38.7**
authority	22.0	**45.1**	arm	10.2	**27.1**
bar	**52.1**	16.9	atmosphere	**31.3**	24.7
bum	18.8	**72.5**	audience	**51.8**	34.0
chair	**62.9**	54.8	bank	32.3	**60.6**
channel	**28.7**	27.9	degree	39.3	**43.8**
child	1.6	**46.8**	difference	**26.4**	23.7
church	**62.1**	58.1	difficulty	13.0	**43.5**
circuit	**52.8**	47.2	disc	**52.2**	45.0
day	2.2	**36.7**	image	4.1	**23.0**
detention	16.7	**62.5**	interest	**26.8**	23.1
dyke	**89.3**	85.7	judgment	20.6	**25.0**
facility	26.8	**50.0**	organization	**71.4**	69.6
fatigue	**73.8**	67.5	paper	25.6	**42.7**
feeling	**51.0**	49.0	party	**67.5**	67.2
grip	8.0	**26.0**	performance	20.5	**33.3**
hearth	37.5	**40.6**	plan	**78.0**	76.2
holiday	7.4	**74.1**	shelter	36.2	**44.9**
lady	**79.3**	8.7	sort	13.5	**65.6**
material	50.8	50.8	source	22.4	**53.1**
mouth	41.2	**43.9**			
nation	**80.6**	36.1			
nature	**44.4**	26.7			
post	**47.4**	36.2			
restraint	9.1	**22.7**			
sense	18.6	**48.8**			
spade	**66.1**	32.3			
stress	**52.6**	21.1			
yew	**85.2**	55.6			
Avg S2	39.9	**41.5**	Avg S3	34.2	**43.2**
Avg S2-S3	36.8	**42.4**			

Table 2: Recall of the "Monosemous Relatives" method (MR) and the "Relatives in Context" (RC) technique in the two Senseval datasets. Best results per word in bold.

results show that the new method clearly outperforms the monosemous relatives in this dataset. However, we can also notice that this improvement does not happen for all the words in the set. One of the problems of unsupervised systems is that they are not able to perform robustly for all words, as supervised can do because of the valuable information contained in the hand-tagged corpora. Thus, we normally see different performances depending on the type of words in the target set, which suggest that the best way to raise unsupervised performance is the combination of algorithms, as we will see in Section 7.

Even if an all-words approach gives a better idea of the performance of different techniques, the Senseval lexical-sample dataset tries to include words with different degrees of polysemy and frequency in order to provide a balanced evaluation. We also show in Section 7 the performance of other techniques previously described in Sec-

S.	Definition
1	beginning, origin, root, rootage - the place where something begins
2	informant - a person who supplies information
3	reference - a publication (or a passage from a publication) that is referred to
4	document (or organization) from which information is obtained
5	facility where something is available
6	seed, germ - anything that provides inspiration for later work
7	generator, author - someone who originates or causes or initiates something

Table 3: Sense inventory for *source* in WordNet 1.7.1.

tion 2.

Sometimes it is worth to "eyeball" the real examples in order to get insight on the algorithms. For that, we chose the word *source* in the S3LS dataset, which clearly improves its performance with the new method. This word has 7 senses in WordNet 1.7.1, shown in Table 3. The Senseval grouping provided by the organisation joins senses 3 and 4, leaving each of the others as separate groups. The coarse inventory of senses has been seen as an alternative to fine-grained WSD (Ciaramita and Altun, 2006).

For this word, we see that the "monosemous relatives" approach achieves a low recall of 22.4%. Analysing the results per sense, we observed that the precision is good for sense 1 (90%), but the recall is as low as 4.7%, which indicates that the algorithm misses many of the instances. The drop in performance seems due to the following reasons: (i) close monosemous relatives found for sense 1 are rare (direct hyponyms such as "headspring" or "provenance" are used), and (ii) far and highly productive relatives are used for senses 2 and 7, which introduce noise (e.g. the related multiword "new edition" for sense 7). In the case of the "relatives in context" algorithm, even if we have a similar set of relatives per each sense, the local context seems to help disambiguate better, achieving a higher recall of 53.1%. In this case the first sense, which is the most frequent in the test set (with 65% of the instances), is better represented and this allows for improved recall.

Following with the target word *source*, we picked a real example from the test set to see the behaviour of the algorithms. This sentence was hand-tagged with sense 1, and we show here a fragment containing the target word:

> *...tax will have been deducted at* ***source****, and this will enable you to sign a Certificate of Deduction...*

The monosemous relatives method is not able to find good collocations in the noisy training data, and it has to rely in bag-of-word features to make its decision. These are not usually as precise as local collocations, and in the example they point to senses 1, 2, and 4. The scorer gives only 1/3 credit to the algorithm in this case. Notice that one of the advantages of using Decision Lists is that it allows us to have a closer look to the features that are applied in each decision. Regarding the "relatives in context" method, in this example it is able to find the correct sense relying in collocations such as *deducted at origin* and *deducted at beginning*.

7 Comparison with other systems

In this section we compared our results with some of the state-of-the-art systems described in Section 2 for this dataset. We chose the Automatic Ranking of Senses by (McCarthy et al., 2004), and the Machine Translation approach by (Wang and Martinez, 2006). These unsupervised systems were selected for a number of reasons: they have been tested in Senseval data with good performance, the techniques are based on different knowledge sources, and the results on Senseval data were available to us. We also devised a simple unsupervised heuristic that would always choose the sense that had a higher number of close relatives in WordNet, picking randomly when there was a tie. We tested this approach in previous work (Wang and Martinez, 2006) and it showed to work well for discarding rare senses. We applied it here as a standalone system. We do not include the results of supervised systems because they can benefit strongly from ready-made hand-tagged data, which is not normally available in a real setting.

The performance of the three systems, together with the previous two, is given in Table 4. We can see that overall the Automatic ranking approach (RK) gives the best performance, with almost the same score as our Relatives in Context (RC) approach. The Machine Translation (MT) method performs 2 points lower overall, but its recall is balanced in the two different datasets. Surprisingly, the simple Number of Relatives (NR) heuristic does better than the Monosemous Relatives (MR), performing specially well in the S3LS

Algorithm	Avg S2LS	Avg S3LS	Avg S2LS-S3LS
RK	39.0	**45.5**	**42.5**
MT	40.8	40.7	40.7
NR	33.6	43.0	38.7
RC	**41.5**	43.2	42.4
MR	39.9	34.2	36.8

Table 4: Recall of different algorithms on Senseval datasets. Best results per column in bold. RK: Automatic Ranking of Senses, MT: Translation-based, NR: Number of Relatives heuristic, RC: Relatives in Context, MR: Monosemous Relatives.

dataset.

We can now analyse the performance of the five systems per word. The results are given in Table 5. We can see that the choice of the target word heavily affects the results of the algorithms, with most of them having very low results for a handful of words, with recall below 20% and even 10%. These very low results make a difference when compared to supervised systems, which do degrade gracefully. None of the algorithms is robust enough to achieve an acceptable performance for all words.

Measuring the agreement of the different algorithms is a way to know if a combined system would improve the results. We calculated the kappa statistic, which has been widely used in NLP tasks (Carletta, 1996), to measure the agreement on the answers of the algorithms in S2LS and S3LS (cf. Table 6). The table shows the averaged results per word of the S2LS dataset in the upper-right side, and the S3LS values in the bottom-left side. We can see that all the results are closer to 0 than 1, indicating that they tend to disagree, and suggesting that the systems offer good prospects for combination. The highest agreement is attained between methods RK and NR in both datasets, and the lowest between RC and MT.

In order to study the potential for combination, we tried the simplest method, that of one system one vote, where each system returns a single vote for the winning sense, and the sense getting most votes wins. In case of ties, all the senses getting the same number of votes are returned. Note that the Senseval scorer penalises systems returning multiple senses (unless all of them are correct).

The results of the ensemble and also of leaving one system out in turn are given in Table 7. The table shows that the best combination (in bold) for each of the datasets varies, which is natural given

Words	Algorithms				
	MR	RC	MT	RK	NR
art	**61.1**	40.3	47.2	**61.1**	**61.1**
authority	22.0	**45.1**	17.6	37.4	37.4
bar	**52.1**	16.9	44.1	14.4	14.4
bum	18.8	72.5	80.0	**85.0**	7.5
chair	62.9	54.8	85.5	**88.7**	**88.7**
channel	28.7	27.9	**47.1**	10.3	2.9
child	1.6	46.8	35.5	43.5	**56.5**
church	**62.1**	58.1	32.3	40.3	40.3
circuit	52.8	47.2	**54.7**	43.4	43.4
day	2.2	**36.7**	32.4	1.4	4.3
detention	16.7	62.5	79.2	**87.5**	12.5
dyke	**89.3**	85.7	67.9	**89.3**	**89.3**
facility	26.8	**50.0**	17.9	26.8	26.8
fatigue	73.8	67.5	67.5	**82.5**	**82.5**
feeling	51.0	49.0	16.3	**59.2**	**59.2**
grip	8.0	**26.0**	14.0	16.0	8.0
hearth	37.5	40.6	56.2	**75.0**	**75.0**
holiday	7.4	74.1	11.1	7.4	**96.3**
lady	**79.3**	8.7	45.7	10.9	10.9
material	**50.8**	**50.8**	19.5	15.3	15.3
mouth	41.2	43.9	41.2	**56.1**	**56.1**
nation	**80.6**	36.1	37.5	**80.6**	19.4
nature	**44.4**	26.7	22.2	17.8	21.1
post	**47.4**	36.2	37.9	43.1	43.1
restraint	9.1	**22.7**	13.6	18.2	9.1
sense	18.6	**48.8**	37.2	11.6	11.6
spade	66.1	32.3	**67.7**	**67.7**	3.2
stress	52.6	21.1	**55.3**	50.0	2.6
yew	**85.2**	55.6	**85.2**	81.5	81.5
argument	24.7	38.7	45.9	**51.4**	21.6
arm	10.2	27.1	71.4	**82.0**	44.0
atmosphere	31.3	24.7	45.7	**66.7**	**66.7**
audience	51.8	34.0	57.0	**67.0**	**67.0**
bank	32.3	60.6	37.1	**67.4**	**67.4**
degree	39.3	**43.8**	41.4	22.7	16.4
difference	26.4	23.7	32.5	**40.4**	16.7
difficulty	13.0	**43.5**	26.1	34.8	34.8
disc	52.2	45.0	**58.0**	27.0	27.0
image	4.1	23.0	21.6	**36.5**	**36.5**
interest	26.8	23.1	31.2	**41.9**	11.8
judgment	20.6	25.0	**40.6**	28.1	28.1
organization	71.4	69.6	19.6	**73.2**	**73.2**
paper	25.6	**42.7**	30.8	23.1	25.6
party	**67.5**	67.2	52.6	6.9	62.1
performance	20.5	33.3	**46.0**	24.1	26.4
plan	78.0	76.2	29.8	**82.1**	**82.1**
shelter	36.2	**44.9**	39.8	33.7	**44.9**
sort	13.5	**65.6**	20.8	**65.6**	**65.6**
source	22.4	53.1	9.4	0.0	**65.6**
Wins	11	12	8	**22**	18
Average	36.8	42.4	40.7	**42.5**	38.7

Table 5: Recall of the 5 algorithms per word and in average, the best results per word are given in bold. The top rows show the S2LS words, and the bottom rows the S3LS words.

the variance of each of the single systems, and that the combination of all 5 systems attains very good performance on both datasets.

In the lower lines, Table 7 shows a number of reference systems: the best unsupervised system that took part in each of the S2LS and S3LS com-

Algorithm	MR	RC	MT	RK	NR
MR	-	0.23	0.28	0.41	0.43
RC	0.13	-	**0.13**	0.31	0.30
MT	0.11	**0.09**	-	0.23	0.35
RK	0.33	0.25	0.26	-	**0.45**
NR	0.23	0.15	0.28	**0.36**	-

Table 6: Averaged kappa agreement between pairs of algorithms. Results on the S2LS dataset are given in the upper-right side, and the S3LS values in the bottom-left side.

System	S2LS	S3LS
All	42.3	51.0
Leave MR out	41.7	**52.1**
Leave RC out	40.3	48.3
Leave MT out	40.0	47.5
Leave RK out	44.5	46.7
Leave NR out	**45.9**	49.9
Best Senseval unsup	35.8	47.5
Best single system	41.5	45.5
Oracle	80.4	84.3

Table 7: Voting systems, best unsupervised systems, best single systems, and oracle on S2LS and S3LS.

petitions, and the best single system in each of the datasets[2]. The combination of the 5 systems is able to beat all of them in both datasets, showing that the simple voting system was effective to improve the single systems and attain the best totally unsupervised system in this dataset. We can also see that the novel technique described in this paper (RC) contributes to improve the ensemble in both datasets. This does not happen for the monosemous relatives approach, which degrades performance in S3LS.

As an upperbound, we also include the oracle combination in Table 7, which determines that an instance has been correctly tagged if any of the algorithms has got it right. This oracle shows that the union of the 5 systems cover 80.4% and 84.3% of the correct solutions for each of the datasets, and that there is ample room for more sophisticated combination strategies.

8 Conclusions and Future work

The current situation for WSD is somewhat stuck due to lack of training data. We present in this paper a novel disambiguation algorithm that improves previous systems based on the acquisition of examples by incorporating local context information. With a basic configuration, our method is able to obtain state-of-the-art performance.

We complemented this work by evaluating other well-known methods in the same dataset, and analysing the comparative results per word. We observed that each algorithm performed better for different types of words, and each of them failed for some particular words. We then proposed a simple unsupervised voting scheme that improved significantly over single systems, achieving the best performance on both the Senseval 2 and Senseval 3 lexical sample datasets.

We have also shown that there is ample room for improvement, as the oracle combination sets an upperbound of around 80% for a perfect combination. This work naturally leads to explore more sophisticated combination strategies, using meta-learning to try to understand which features of each word make a certain WSD system succeed (or fail). We would also like to widen the range of systems, either using existing unsupervised off-the-shelf WSD systems and/or reimplementing them.

Regarding the "Relatives in Context" method, there are different avenues to explore. We plan to use this approach to acquire automatic sense tagged data for training, instead of relying on rules. We also would like to study the use of richer features than the local strings to acquire examples that have similar linguistic structures.

Finally, we want to test the new technique on an all-words corpus. A simple approach would be to process each instance of each word separately as in the lexical sample. However, we could also try to disambiguate all words in the context together, by substituting the target words with their relatives jointly. We are comparing our unsupervised systems in the testbeds where supervised systems are comfortable (lexical-sample tasks). We think that unsupervised systems can have the winning hand in more realistic settings like those posed by Senseval all-words tasks.

[2]For nouns the best scores of the competing systems were obtained by dictionary-based systems in both S2LS (Litkowski, 2001), and S3LS (Pedersen, 2004).

References

E. Agirre and P. Edmonds. 2006. *Word Sense Disambiguation*. Kluwer Academic Publishers.

E. Agirre and D. Martinez. 2004. Unsupervised wsd based on automatically retrieved examples: The importance of bias. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

B. Snyder and M. Palmer. 2004. The English all-words task. In *Proceedings of the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, Barcelona, Spain.

J. Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. In *Computational Linguistics 22(2)*.

Y.S. Chan and H.T. Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, Pittsburgh, Pennsylvania, USA.

M. Ciaramita and Y. Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of EMNLP 2006*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

A. Kilgarriff. 2001. English Lexical Sample Task Description. In *Proceedings of the Second International Workshop on evaluating Word Sense Disambiguation Systems*, Toulouse, France.

U. Kohomban and W.S. Lee. 2005. Learning Semantic Classes for Word Sense Disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

C. Leacock, M. Chodorow, and G. A. Miller. 1998. Using corpus statistics and WordNet relations for sense identification. In *Computational Linguistics*, volume 24, pages 147–165.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *In Proceedings of COLING-ACL*, Montreal, Canada.

K. Litkowski. 2001. Use of Machine-Readable Dictionaries in Word-Sense Disambiguation for Senseval-2. In *Proceedings of the Second International Workshop on evaluating Word Sense Disambiguation Systems*, Toulouse, France.

B. Magnini and G. Cavagliá. 2000. Integrating subject field codes into WordNet. In *Proceedings of the Second International LREC Conference*, Athens, Greece.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. Finding Predominant Word Senses in Untagged Text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

R. Mihalcea, T. Chklovski, and Adam Killgariff. 2004. The Senseval-3 English lexical sample task. In *Proceedings of the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, Barcelona, Spain.

R. Mihalcea. 2002. Bootstrapping large sense tagged corpora. In *Proceedings of the 3rd International Conference on Languages Resources and Evaluations (LREC 2002)*, Las Palmas, Spain.

C. Niu, W. Li, R.K. Srihari, and H. Li. 2005. Word independent context pair classification model for word sense disambiguation. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*.

T. Pedersen. 2004. The duluth lexical sample systems in senseval-3. In *Proceedings of the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, Barcelona, Spain.

P. Resnik and A. Elkiss. 2005. The linguist's search engine: An overview. In *Proceedings of ACL 2005 (Demonstration Section)*.

X. Wang and J. Carroll. 2005. Word sense disambiguation using sense examples automatically acquired from a second language. In *Proceedings of HLT/EMNLP*, Vancouver, Canada.

X. Wang and D. Martinez. 2006. Word sense disambiguation using automatically translated sense examples. In *Proceedings of EACL 2006 Workshop on Cross Language Knowledge Induction*, Trento, Italy.

D. Yarowsky. 1994. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM.

Named Entity Recognition for Question Answering

Diego Mollá and **Menno van Zaanen** and **Daniel Smith**
Centre for Language Technology
Macquarie University
Sydney
Australia
{diego, menno, dsmith}@ics.mq.edu.au

Abstract

Current text-based question answering (QA) systems usually contain a named entity recogniser (NER) as a core component. Named entity recognition has traditionally been developed as a component for information extraction systems, and current techniques are focused on this end use. However, no formal assessment has been done on the characteristics of a NER within the task of question answering. In this paper we present a NER that aims at higher recall by allowing multiple entity labels to strings. The NER is embedded in a question answering system and the overall QA system performance is compared to that of one with a traditional variation of the NER that only allows single entity labels. It is shown that the added noise produced introduced by the additional labels is offset by the higher recall gained, therefore enabling the QA system to have a better chance to find the answer.

1 Introduction

Many natural language processing applications require finding named entities (NEs) in textual documents. NEs can be, for example, person or company names, dates and times, and distances. The task of identifying these in a text is called named entity recognition and is performed by a named entity recogniser (NER).

Named entity recognition is a task generally associated with the area of information extraction (IE). Firstly defined as a separate task in the Message Understanding Conferences (Sundheim, 1995), it is currently being used in a varied range of applications beyond the generic task of information extraction, such as in bioinformatics, the identification of entities in molecular biology (Humphreys et al., 2000), and text classification (Armour et al., 2005).

In this paper we will focus on the use of named entity recognition for question answering. For the purposes of this paper, question answering (QA) is the task of automatically finding the answer to a question phrased in English by searching through a collection of text documents. There has been an increase of research in QA since the creation of the question answering track of TREC (Voorhees, 1999), and nowadays we are starting to see the introduction of question-answering techniques in mainstream web search engines such as Google[1], Yahoo![2] and MSN[3].

An important component of a QA system is the named entity recogniser and virtually every QA system incorporates one. The rationale of incorporating a NER as a module in a QA system is that many fact-based answers to questions are entities that can be detected by a NER. Therefore, by incorporating in the QA system a NER, the task of finding some of the answers is simplified considerably.

The positive impact of NE recognition in QA is widely acknowledged and there are studies that confirm it (Noguera et al., 2005). In fact, virtually every working QA system incorporates a NER. However, there is no formal study of the optimal characteristics of the NER within the context of QA. The NER used in a QA system is typically developed as a stand-alone system designed independently of the QA task. Sometimes

[1] http://www.google.com
[2] http://search.yahoo.com
[3] http://search.msn.com

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 51–58.

it is even used as a black box that is not fine-tuned to the task. In this paper we perform a step towards such a formal study of the ideal characteristics of a NER for the task of QA. In particular, section 2 comments on the desiderata of a NER for QA. Next, section 3 describes the QA system used in the paper, while section 4 describes the NER and its modifications for its use for QA. Section 5 presents the results of various experiments evaluating variations of the NER, and finally Section 6 presents the concluding remarks and lines of further research.

2 Named Entity Recognition for Question Answering

Most QA systems gradually reduce the amount of data they need to consider in several phases. For example, when the system receives a user question, it first selects a set of relevant documents, and then filters out irrelevant pieces of text of these documents gradually until the answer is found.

The NER is typically used as an aid to filter out strings that do not contain the answer. Thus, after a question analysis stage the type of the expected answer is determined and mapped to a list of entity types. The NER is therefore used to single out the entity types appearing in a text fragment. If a piece of text does not have any entity with a type compatible with the type of the expected answer, the text is discarded or heavily penalised. With this in mind, the desiderata of a NER are related with the range of entities to detect and with the recall of the system.

2.1 Range of Entities

Different domains require different types of answers. Typically, the question classification component determines the type of question and the type of the expected answer. For example, the questions used in the QA track of past TREC conferences can be classified following the taxonomy shown in Table 1 (Li and Roth, 2002).

The set of entity types recognised by a standalone NER is typically very different and much more coarse-grained. For example, a typical set of entity types recognised by a NER is the one defined in past MUC tasks and presented in Table 2. The table shows a two-level hierarchy and the types are much more coarse-grained than that of Table 1. Within each of the entity types of Table 2 there are several types of questions of Table 1.

ABBREVIATION abb, exp
ENTITY animal, body, color, creative, currency, dis.med., event, food, instrument, lang, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
DESCRIPTION definition, description, manner, reason
HUMAN group, ind, title, description
LOCATION city, country, mountain, other, state
NUMERIC code, count, date, distance, money, order, other, period, percent, speed, temp, size, weight

Table 1: Complete taxonomy of Li & Roth

Class	Type
ENAMEX	Organization
	Person
	Location
TIMEX	Date
	Time
NUMEX	Money
	Percent

Table 2: Entities used in the MUC tasks

A QA system typically uses both a taxonomy of expected answers and the taxonomy of named entities produced by its NER to identify which named entities are relevant to a question. The question is assigned a type from a taxonomy such as defined in Table 1. This type is then used to filter out irrelevant named entities that have types as defined in Table 2.

A problem that arises here is that the granularity of the NEs provided by a NER is much coarser than the ideal granularity for QA, as the named entity types are matched against the types the question requires. Consequently, even though a question classifier could determine a very specific type of answer, this type needs to be mapped to the types provided by the NER.

2.2 Recall

Given that the NER is used to filter out candidate answers, it is important that only wrong answers are removed, while all correct answers stay in the set of possible answers. Therefore, recall in a NER in question answering is to be preferred above precision. Generally, a NER developed for a generic NE recognition task (or for information extraction) is fine-tuned for a good balance between recall and precision, and this is not necessarily what

we need in this context.

2.2.1 Multi-labelling

Recognising named entities is not a trivial task. Most notably, there can be ambiguities in the detection of entities. For example, it can well happen that a text has two or more interpretations. Notable examples are names of people whose surname takes the form of a geographical location (*Europe*, *Africa*) or a profession (*Smith*, *Porter*). Also, names of companies are often chosen after the name of some of their founders. The problem is that a NER typically only assigns one label to a specific piece of text. In order to increase recall, and given that NE recognition is not an end task, it is therefore theoretically advisable to allow to return multiple labels and then let further modules of the QA system do the final filtering to detect the exact answer. This is the hypothesis that we want to test in the present study. The evaluations presented in this paper include a NER that assigns single labels and a variation of the same NER that produces multiple, overlapping labels.

3 Question Answering

QA systems typically take a question presented by the user posed in natural language. This is then analysed and processed. The final result of the system is an *answer*, again in natural language, to the question of the user. This is different from, what is normally considered, information retrieval in that the user presents a complete question instead of a query consisting of search keywords. Also, instead of a list of relevant documents, a QA system typically tries to find an exact answer to the question.

3.1 AnswerFinder

The experiments discussed in this paper have been conducted within the AnswerFinder project (Mollá and van Zaanen, 2005). In this project, we develop the AnswerFinder question answering system, concentrating on shallow representations of meaning to reduce the impact of paraphrases (different wordings of the same information). Here, we report on a sub-problem we tackled within this project, the actual finding of correct answers in the text.

The AnswerFinder question answering system consists of several phases that essentially work in a sequential manner. Each phase reduces the amount of data the system has to handle from then on. The advantage of this approach is that progressive phases can perform more "expensive" operations on the data.

The first phase is a document retrieval phase that finds documents relevant to the question. This greatly reduces the amount of texts that need to be handled in subsequent steps. Only the best n documents are used from this point on.

Next is the sentence selection phase. From the relevant documents found by the first phase, all sentences are scored against the question. The most relevant sentences according to this score are kept for further processing.

At the moment, we have implemented several sentence selection methods. The most simple one is based on word overlap and looks at the number of words that can be found in both the question and the sentence. This is the method that will be used in the experiments reported in this paper. Other methods implemented, but not used in the experiments, use richer linguistic information. The method based on grammatical relation (Carroll et al., 1998) overlap requires syntactic analysis of the question and the sentence. This is done using the Connexor dependency parser (Tapanainen and Järvinen, 1997). The score is computed by counting the grammatical relations found in both sentence and question. Logical form overlap (Mollá and Gardiner, 2004) relies on logical forms that can be extracted from the grammatical relations. They describe shallow semantics of the question and sentence. Based on the logical form overlap, we have also implemented logical graph overlap (Mollá, 2006). This provides a more fine-grained scoring method to compute the shallow semantic distance between the question and sentence. All of these methods have been used in a full-fledged question answering system (Mollá and van Zaanen, 2006). However, to reduce variables in our experiments, we have decided to use the simplest method only (word overlap) in the experiments reported in this paper.

After the sentence selection phase, the system searches for the exact answers. Some of the sentence selection methods, while computing the distance, already find some possible answers. For example, the logical graphs use rules to find parts of the sentence that may be exact answers to the question. This information is stored together with the sentence. Note that in this article, we are are only interested in the impact of named entity

recognition in QA, so we will not use any sentence selection method that finds possible answers.

The sentences remaining after the sentence selection phase are then analysed for named entities. All named entities found in the sentences are considered to be possible answers to the user question.

Once all possible answers to the questions are found, the actual answer selection phase takes place. For this, the question is analysed, which provides information on what kind of answer is expected. This can be, for example, country, river, distance, person, etc. as described in Table 1. The set of possible answers is now considered preferring answers that match the question type.

The best answer (i.e. with the highest score and matching the question type) is returned to the user, which finishes a typical question answering interaction.

4 Named Entity Recognition

The ability of finding exact answers by the AnswerFinder system relies heavily on the quality of the named entity recognition performed on the sentences that are relevant to the user question. Finding all named entities in the sentences is therefore of utmost importance. Missing named entities may mean that the answer to the question cannot be recovered anymore.

We have tried different NERs in the context of question answering. In addition to a general purpose NER, we have developed our own NER. Even though several high quality NERs are available, we thought it important to have full control over the NER to make it better suited for the task at hand.

4.1 ANNIE

ANNIE is part of the Sheffield GATE (General Architecture for Text Engineering) system (Gaizauskas et al., 1996) and stands for "A Nearly-New IE system". This architecture does much more than we need, but it is possible to only extract the NER part of it. Unfortunately, there is not much documentation on the NER in ANNIE. The named entity types found by ANNIE match up with the MUC types as described in Table 2.

ANNIE was chosen as an example of a typical NER because it is freely available to the research community and the named entity types are a subset of the MUC types.

4.2 AFNER

In addition to ANNIE's NER, we also look at the results from the NER that is developed within the AnswerFinder project, called *AFNER*.

4.2.1 General Approach

The NER process used in AFNER consists of two phases. The first phase uses hand-written regular expressions and gazetteers (lists of named entities that are searched for in the sentences). These information sources are combined with machine learning techniques in the second phase.

AFNER first tokenises the given text, applies the regular expressions to each token, and searches for occurrences of the token in the gazetteers. Regular expression matches and list occurrences are used as features in the machine learning classifier. These features are used in combination with token specific features, as well as features derived from the text as a whole. Using a model generated from the annotated corpus, each token is classified as either the beginning of ('B') or in ('I') a particular type of named entity, or out ('OUT') of any named entity. The classified tokens are then appropriately combined into named entities.

4.2.2 First Phase — Regular Expressions and Gazetteers

Regular expressions are useful for finding named entities following identifiable patterns, such as dates, times, monetary expressions, etc. As a result, the entities that can be discovered using regular expressions are limited. However, matching a particular regular expression is a key feature used in identifying entities of these particular types. Gazetteers are useful for finding commonly referenced names of people, places or organisations, but are by no means exhaustive. The purpose of combining lists with other features is to supplement the lists used.

4.2.3 Second Phase — Machine Learning

The second phase involves the machine learning component of AFNER. The technique used is maximum entropy, and the implementation of the classifier is adapted from Franz Josef Och's *YASMET*.[4] The system is trained on the Remedia Corpus (Hirschman et al., 1999), which contains annotations of named entities.

The regular expression and gazetteer matches are used as features, in combination with others

[4] http://www.fjoch.com/YASMET.html

pertaining to both individual tokens and tokens in context. Features of individual tokens include those such as capitalisation, alpha/numeric information, etc. Contextual features are those that identify a token amongst surrounding text, or relate to tokens in surrounding text. For example, whether a token is next to a punctuation mark or a capitalised word, or whether a token is always capitalised in a passage of text. Contextual features relating to global information have been used as described by Chieu and Ng (2002). In addition, features of previous tokens are included.

The features are then passed to a maximum entropy classifier which, for every token, returns a list of probabilities of the token to pertain to each category. The categories correspond with each type of entity type prepended with 'B' and 'I', and a general 'OUT' category for tokens not in any entity. The list of entity types used is the same as in the MUC tasks (see Table 2).

Preliminary experiments revealed that often the top two or three entity type probabilities have similar values. For this reason the final named entity labels are computed on the basis of the top n probabilities (provided that they meet a defined threshold), where n is a customisable limit. Currently, a maximum of 3 candidate types are allowed per token.

Classified tokens are then combined according to their classification to produce the final list of named entities. We have experimented with two methods named *single* and *multiple*. For single type combination only one entity can be associated with a string, whereas for multiple type combination several entities can be associated. Also, the multiple type combination allows overlaps of entities. The multiple type combination aims at increasing recall at the expense of ambiguous labelling and decrease of precision.

In the case of multiple type combination (see Figure 1 for an example), each label prepended with 'B' signals the beginning of a named entity of the relevant type, and each 'I' label continues a named entity if it is preceded by a 'B' or 'I' label of the same type. If an 'I' label does not appear after a 'B' classification, it is treated as a 'B' label. In addition, if a 'B' label is preceded by an 'I' label, it will be both added as a separate entity (with the previous entity ending) and appended to the previous entity.

The single type combination (Figure 2) is implemented by filtering out all the overlapping entities of the output of the multiple type combination. This is done by selecting the longest-spanning entity and discarding all substring or overlapping strings. If there are two entities associated with exactly the same string, the one with higher probability is chosen.

The probability of a multi-token entity is computed by combining the individual token probabilities. Currently we use the geometric mean but we are exploring other possibilities. If P_i is the probability of token i and $P_{1...n}$ is the probability of the entire sentence, the geometric mean of the probabilities is computed as:

$$P_{1...n} = e^{\frac{\sum_{i=1}^{n} \log P_i}{n}}$$

5 Results

To evaluate the impact of the quality of NER within the context of question answering, we ran the AnswerFinder system using each of the named entity recognisers, ANNIE, AFNER$_s$ and AFNER$_m$. This section first explains the experimental setup we used, then shows and discusses the results.

5.1 Experimental setup

To evaluate AnswerFinder we used the data available for participants of the QA track of the 2005 TREC competition-based conference[5]. This competition provides us with a nice setting to measure the impact of the NERs. We simply use the documents and questions provided during the TREC 2005 competition. To determine whether a document or text fragment contains the answer we use Ken Litkowsky's answer patterns, also available at the TREC website.

The questions in TREC 2005 are grouped by topic. The competition consisted of 75 topics, with a total of 530 questions. These questions are divided into three different types: factoid, list, and other. In this paper, we only consider the factoid questions, that is, questions that require a single fact as answer. List asks for a list of answers and other is answered by giving any additional information about the topic. There are 362 factoid questions in the question set.

In the experiments, AnswerFinder uses the TREC data as follows. First, we apply docu-

[5] http://trec.nist.gov

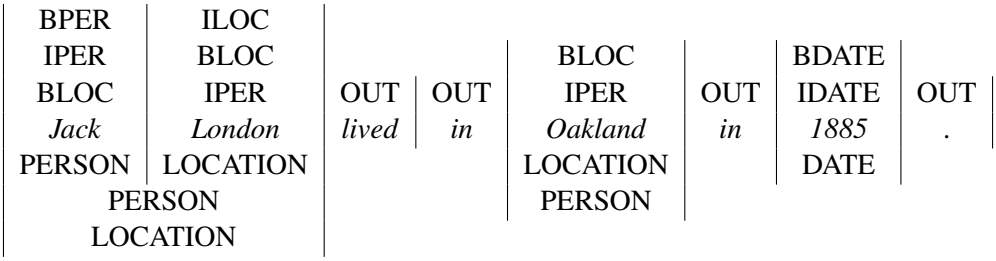

Figure 1: Named entities as multiple labels. The token-based labels appear above the words. The final NE labels appear below the words.

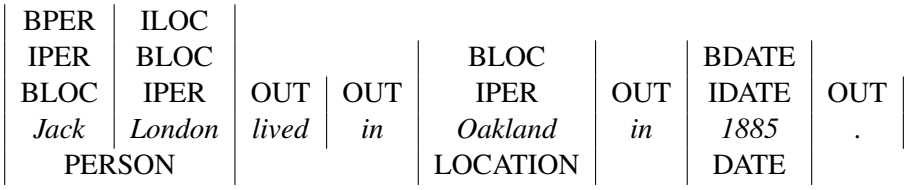

Figure 2: Named entities as single labels. The token-based labels appear above the words. The resulting NE labels appear below the words.

ment selection (using the list of relevant documents for each question provided by TREC). From these documents, we select the n best sentences based on word overlap between the sentence and the question.

We can now compute an upper-bound baseline. By taking the selected sentences as answers, we can compute the maximum score possible from a question answering perspective. By not requiring exactly matching answers, we can count the number of questions that could be answered if the answer selection phase would be perfect. In other words, we measure the percentage of questions that can still be answered if the answer selection part of the system would be perfect.

Next, we run experiments with the same settings, but applying each of the NERs to the relevant sentences. All named entities that are found in these sentences are then considered possible answers to the question and again the percentage of questions that can be answered is computed.

Finally, we embed the NERs in a simplified version of AnswerFinder to test their impact in a baseline QA system.

5.2 Empirical results

In Table 3 we see the percentage of questions that can still be answered after document selection. The table reflects the intuition that, the smaller the number of preselected documents, the more likely it is that the document that contains the answer is left out. The documents are selected using a list of relevant documents provided for the competition.

# of documents	% of questions
10	75.5%
20	81.6%
30	86.9%
40	89.5%
50	92.1%

Table 3: Percentage of factoid questions that can still be answered after document selection

# of sentences	% of questions
5	42.4%
10	49.9%
20	62.0%
30	65.4%
40	68.8%
50	70.8%
60	73.0%
70	73.7%

Table 4: Percentage of factoid questions that can still be answered after sentence selection from the top 50 documents

If we continue with 50 documents after document selection, we can select relevant sentences from the text in these documents using the word overlap metric. We end up with the percentages as given in Table 4.

There is quite a dramatic drop from 92.1% in all the documents to 73.7% with 70 sentences selected. This can be explained from the fact that the

# of sentences	% of questions		
	ANNIE	AFNER$_s$	AFNER$_m$
5	27.9%	11.6%	27.7%
10	33.0%	13.6%	33.3%
20	41.4%	17.7%	41.9%
30	44.3%	19.0%	45.6%
40	46.2%	19.9%	47.4%
50	47.8%	20.5%	48.8%
60	49.3%	21.3%	51.0%
70	50.5%	21.3%	51.5%

Table 5: Percentage of factoid questions that can still be answered after NE recognition from the top 50 documents

word overlap sentence selection is not extremely sophisticated. It only looks at words that can be found in both the question and sentence. In practice, the measure is very coarse-grained. However, we are not particularly interested in perfect answers here, these figures are upper-bounds in the experiment.

From the selected sentences now we extract all named entities. The results are summarised in Table 5.

The figures of Table 5 approximate recall in that they indicate the questions where the NER has identified a correct answer (among possibly many wrong answers).

The best results are those provided by AFNER$_m$ and they are closely followed by ANNIE. This is an interesting result in that AFNER has been trained with the Remedia Corpus, which is a very small corpus on a domain that is different from the AQUAINT corpus. In contrast, ANNIE is fine-tuned for the domain. Given a larger training corpus of the same domain, AFNER$_m$'s results would presumably be much better than ANNIE's.

The results of AFNER$_s$ are much worse than the other two NERs. This clearly indicates that some of the additional entities found by AFNER$_m$ are indeed correct.

It is expected that precision would be different in each NER and, in principle, the noise introduced by the erroneous labels may impact the results returned by a QA system integrating the NER. We have tested the NERs extrinsically by applying them to a baseline setting of AnswerFinder. In particular, the baseline setting of AnswerFinder applies the sentence preselection methods described above and then simply returns the most frequent entity found in the sentences preselected. If there are several entities sharing the top position then one is chosen randomly. In other words, the baseline ignores the question type and the actual context of the entity. We decided to use this baseline setting because it is more closely related to the precision of the NERs than other more sophisticated settings. The results are shown in Table 6.

# of sentences	% of questions		
	ANNIE	AFNER$_s$	AFNER$_m$
10	6.2%	2.4%	5.0%
20	6.2%	1.9%	7.0%
30	4.9%	1.4%	6.8%
40	3.7%	1.4%	6.0%
50	4.0%	1.2%	5.1%
60	3.5%	0.8%	5.4%
70	3.5%	0.8%	4.9%

Table 6: Percentage of factoid questions that found an answer in a baseline QA system given the top 50 documents

The figures show a drastic drop in the results. This is understandable given that the baseline QA system used is very basic. A higher-performance QA system would of course give better results.

The best results are those using AFNER$_m$. This confirms our hypothesis that a NER that allows multiple labels produces data that are more suitable for a QA system than a "traditional" single-label NER. The results suggest that, as long as recall is high, precision does not need to be too high. Thus there is no need to develop a high-precision NER.

The table also indicates a degradation of the performance of the QA system as the number of preselected sentences increases. This indicates that the baseline system is sensitive to noise. The bottom-scoring sentences are less relevant to the question and therefore are more likely not to contain the answer. If these sentences contain highly frequent NEs, those NEs might displace the correct answer from the top position. A high-performance QA system that is less sensitive to noise would probably produce better results as the number of preselected sentences increases (possibly at the expense of speed). The fact that AFNER$_m$, which produces higher recall than AFNER$_s$ according to Table 5, still obtains the best results in the baseline QA system according to Table 6, suggests that the amount

of noise introduced by the additional entities does not affect negatively the process of extracting the answer.

6 Summary and Conclusion

In this paper we have focused on the impact of introducing multiple labels with the aim to increase recall in a NER for the task of question answering. In our experiments we have tested the impact of the ANNIE system, and two variations of AFNER, our custom-built system that can be tuned to produce either single labels or multiple labels. The experiments confirm the hypothesis that allowing multiple labelling in order to increase recall of named entities benefits the task of QA. In other words, if the NER has several candidate labels for a string (or a substring of it), it pays off to output the most plausible alternatives. This way the QA system has a better chance to find the answer. The noise introduced by returning more (possibly wrong) entities is offset by the increase of recall.

Further work includes the evaluation of the impact of multi-label NE recognition on higher-performance QA systems. In particular we plan to test various versions of the complete AnswerFinder system (not just the baseline setting) with each of the NERs. In addition, we plan to retrain AFNER using more data and more relevant data and explore the impact of the single and multiple methods on the resulting higher-performance NER.

Acknowledgements

This work is supported by the Australian Research Council under the ARC Discovery grant DP0450750.

References

[Armour et al.2005] Quintin Armour, Nathalie Japkowicz, and Stan Matwin. 2005. The role of named entities in text classification. In *Proceedings CLiNE 2005*, Gatineau, Canada.

[Carroll et al.1998] John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proc. LREC98*.

[Chieu and Ng2002] Haoi Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *Proceedings COLING 2002*.

[Gaizauskas et al.1996] Robert Gaizauskas, Hamish Cunningham, Yorick Wilks, Peter Rodgers, and Kevin Humphreys. 1996. GATE: an environment to support research and development in natural language engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France.

[Hirschman et al.1999] Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep Read: A reading comprehension system. In *Proc. ACL'99*. University of Maryland.

[Humphreys et al.2000] Kevin Humphreys, George Demetriou, and Robert Gaizauskas. 2000. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proceedings of the Pacific Symposium on Biocomputing' 00 (PSB'00)*, pages 502–513. Honolulu, Hawaii.

[Li and Roth2002] Xin Li and Dan Roth. 2002. Learning question classifiers. *Proc. COLING 02*.

[Mollá and Gardiner2004] Diego Mollá and Mary Gardiner. 2004. Answerfinder - question answering by combining lexical, syntactic and semantic information. In Ash Asudeh, Cécile Paris, and Stephen Wan, editors, *Proc. ALTW 2004*, pages 9–16, Sydney, Australia. Macquarie University.

[Mollá and van Zaanen2005] Diego Mollá and Menno van Zaanen. 2005. Learning of graph rules for question answering. In Tim Baldwin and Menno van Zaanen, editors, *Proc. ALTW 2005*. ALTA.

[Mollá and van Zaanen2006] Diego Mollá and Menno van Zaanen. 2006. Answerfinder at TREC 2005. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proc. TREC 2005*. NIST.

[Mollá2006] Diego Mollá. 2006. Learning of graph-based question answering rules. In *Proc. HLT/NAACL 2006 Workshop on Graph Algorithms for Natural Language Processing*, pages 37–44.

[Noguera et al.2005] Elisa Noguera, Antonio Toral, Fernando Llopis, and Rafael Muñoz. 2005. Reducing question answering input data using named entity recognition. In *Proceedings of the 8th International Conference on Text, Speech & Dialogue*, pages 428–434.

[Sundheim1995] Beth M. Sundheim. 1995. Overview of results of the MUC-6 evaluation. In *Proc. Sixth Message Understanding Conference MUC-6*. Morgan Kaufmann Publishers, Inc.

[Tapanainen and Järvinen1997] Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proc. ANLP-97*. ACL.

[Voorhees1999] Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC-8*, number 500-246 in NIST Special Publication. NIST.

Named Entity Recognition for Astronomy Literature

Tara Murphy and **Tara McIntosh** and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{tm,tara,james}@it.usyd.edu.au

Abstract

We present a system for named entity recognition (NER) in astronomy journal articles. We have developed this system on a NE corpus comprising approximately 200,000 words of text from astronomy articles. These have been manually annotated with ∼40 entity types of interest to astronomers.

We report on the challenges involved in extracting the corpus, defining entity classes and annotating scientific text. We investigate which features of an existing state-of-the-art Maximum Entropy approach perform well on astronomy text. Our system achieves an F-score of 87.8%.

1 Introduction

Named entity recognition (NER) involves assigning broad semantic categories to entity references in text. While many of these categories do in fact refer to *named* entities, e.g. PERSON and LOCATION, others are not proper nouns, e.g. DATE and MONEY. However, they are all syntactically and/or semantically distinct and play a key role in Information Extraction (IE). NER is also a key component of Question Answering (QA) systems (Hirschman and Gaizauskas, 2001). State-of-the-art QA systems often have custom-built NER components with finer-grained categories than existing corpora (Harabagiu et al., 2000). For IE and QA systems, generalising entity references to broad semantic categories allows shallow extraction techniques to identify entities of interest and the relationships between them.

Another recent trend is to move beyond the traditional domain of newspaper text to other corpora. In particular, there is increasing interest in extracting information from scientific documents, such as journal articles, especially in biomedicine (Hirschman et al., 2002).

A key step in this process is understanding the entities of interest to scientists and building models to identify them in text. Unfortunately, existing models of language perform very badly on scientific text even for the categories which map directly between science and newswire, e.g. PERSON. Scientific entities often have more distinctive orthographic structure which is not exploited by existing models.

In this work we identify entities within astronomical journal articles. The astronomy domain has several advantages: firstly, it is representative of the physical sciences; secondly, the majority of papers are freely available in a format that is relatively easy to manipulate (LaTeX); thirdly, there are many interesting entity types to consider annotating; finally, there are many databases of astronomical objects that we will eventually exploit as gazetteer information.

After reviewing comparable named entity corpora, we discuss aspects of astronomy that make it challenging for NLP. We then describe the corpus collection and extraction process, define the named entity categories and present some examples of interesting cases of ambiguity that come up in astronomical text.

Finally, we describe experiments with retraining an existing Maximum Entropy tagger for astronomical named entities. Interestingly, some feature types that work well for newswire significantly degrade accuracy here. We also use the tagger to detect errors and inconsistencies in the annotated corpus. We plan to develop a much larger freely available astronomy NE corpus based on our experience described here.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 59–66.

2 Existing annotated corpora

Much of the development in NER has been driven by the corpora available for training and evaluating such systems. This is because the state-of-the-art systems rely on statistical machine learning approaches.

2.1 Message Understanding Conference

The MUC named entity recognition task (in MUC 6/7) covered three types of entities:

names PERSON, LOCATION, ORGANISATION;

temporal expressions DATE, TIME;

numeric expressions MONEY, PERCENT.

The distribution of these types in MUC 6 was: names 82%, temporal 10% and numeric 8%, and in MUC 7 was: names 67%, temporal 25% and numeric 6%.

The raw text for the MUC 6 NER corpus consisted of 30 Wall Street Journal articles, provided by the Linguistic Data Consortium (LDC). The text used for the English NER task in MUC 7 was from the New York Times News Service, also from the LDC. There are detailed annotation guidelines available.[1]

2.2 GENIA corpus

The GENIA corpus (Kim et al., 2003) is a collection of 2000 abstracts from the National Library of Medicine's MEDLINE database. The abstracts have been selected from search results for the keywords human, blood cells and transcription factors. GENIA is annotated with a combination of part of speech (POS) tags based on the Penn Treebank set (Marcus et al., 1994) and a set of biomedical named entities described in the GENIA ontology. One interesting aspect of the GENIA corpus is that some named entities are syntactically nested. However, most statistical NER systems are sequence taggers which cannot easily represent hierarchical tagging.

2.3 Astronomy Bootstrapping Corpus

The Astronomy Bootstrapping Corpus (Becker et al., 2005; Hachey et al., 2005) is a small corpus consisting of 209 abstracts from the NASA Astronomical Data System Archive. The corpus was developed as part of experiments into efficient methods for developing new statistical models for NER. The abstracts were selected using the query quasar + line from articles published between 1997 and 2003. The corpus was annotated with the following named entity types:

1. INSTRUMENT NAME (136 instances)
2. SOURCE NAME (111 instances)
3. SOURCE TYPE (499 instances)
4. SPECTRAL FEATURE (321 instances)

The seed and test sets (50 and 159 abstracts) were annotated by two astronomy PhD students. The abstracts contained on average 10 sentences with an average length of 30 tokens, implying an *tag density* (the percentage of words tagged as a named entity) of $\sim 2\%$.

3 NLP for astronomy

Astronomy is a broad scientific domain combining theoretical, observational and computational research, which all differ in conventions and jargon. We are interested in NER for astronomy within a larger project to improve information access for scientists.

There are several comprehensive text and scientific databases for astronomy. For example, NASA Astrophysics Data System (ADS, 2005) is a bibliographic database containing over 4 million records (journal articles, books, etc) covering the areas of astronomy and astrophysics, instrumentation, physics and geophysics. ADS links to various external resources such as electronic articles, data catalogues and archives.

3.1 IAU naming conventions

The naming of astronomical objects is specified by the International Astronomical Union's (IAU) Commission 5, so as to minimise confusing or overlapping designations in the astronomical literature. The most common format for object names is a catalogue code followed by an abbreviated position (Lortet et al., 1994). Many objects still have common or historical names (e.g. the Crab Nebula). An object that occurs in multiple catalogues will have a separate name in each catalogue (e.g. PKS 0531+21 and NGC 1952 for the Crab Nebula).

[1] www.cs.nyu.edu/cs/faculty/grishman/muc6.html

1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
2	90	218	421	1909	3221	4320	5097	5869	6361	6556	7367	7732	3495

Table 1: Number of LaTeX astro-ph articles extracted for each year.

3.2 The Virtual Observatory

There is a major effort in astronomy to move towards integrated databases, software and telescopes. The umbrella organisation for this is the *International Virtual Observatory Alliance* (Hanisch and Quinn, 2005). One of the aims is to develop a complete ontology for astronomical data which will be used for Unified Content Descriptors (Martinez et al., 2005).

4 Collecting the raw corpus

The process of collecting the raw text for named entity annotation involved first obtaining astronomy text, extracting the raw text from the document formatting and splitting it into sentences and tokens.

4.1 arXiv

arXiv (arXiv, 2005) is an automated distribution system for research articles, started in 1991 at the Los Alamos National Laboratory to provide physicists with access to prepublication materials. It rapidly expanded to incorporate many domains of physics and thousands of users internationally (Ginsparg, 2001).

The astrophysics section (astro-ph) is used by most astronomers to distribute papers before or after publication in a recognised journal. It contains most of astrophysics publications from the last five to ten years. Table 1 shows that the number of articles submitted to astro-ph has increased rapidly since 1995.

The articles are mostly typeset in LaTeX. We have downloaded 52 658 articles from astro-ph, totalling approximately 180 million words. In creating the NE corpus we limited ourselves to articles published since 2000, as earlier years had irregular LaTeX usage.

4.2 LaTeX conversion

After collecting of LaTeX documents, the next step was to extract the text so that it could be processed using standard NLP tools. This normally involves ignoring most formatting and special characters in the documents.

However formatting and special characters play a major role in scientific documents, in the form of mathematical expressions, which are interspersed through the text. It is impossible to ignore every non-alphanumeric character or or map them back to some standard token because too much information is lost. Existing tools such as DeTeX (Trinkle, 2002) remove LaTeX markup including the mathematics, rendering scientific text nonsensical. Keeping the LaTeX markup is also problematic since the tagger's morphological features are confused by the markup.

4.3 LaTeX to Unicode

Our solution was to render as much of the LaTeX as possible in text, using Unicode (Unicode Consortium, 2005) to represent the mathematics as faithfully as possible. Unicode has excellent support for mathematical symbols and characters, including the Greek letters, operators and various accents.

Mapping LaTeX back to the corresponding Unicode character is difficult. For example, `\acirc`, `\^{a}` and `\hat{a}` are all used to produce â, which in Unicode is 0x0174.

Several systems attempt to convert LaTeX to other formats e.g. XML (Grimm, 2003). No existing system rendered the mathematics faithfully enough or with high enough coverage for our purposes. Currently our coverage of mathematics is very good but there are still some expressions that cannot be translated, e.g. complex nested expressions, rare symbols and non-Latin/Greek/Hebrew alphabetic characters.

4.4 Sentences and Tokenisation

We used MXTerminator (Reynar and Ratnaparkhi, 1997) as the sentence boundary detector with an additional Python script to fix common errors, e.g. mistaken boundaries on Sect. and et al. We used the Penn Treebank (Marcus et al., 1994) sed script to tokenize the text, again with a Python script to fix common errors, e.g. splitting numbers like 1,000 on the

Our **FUSE**|TEL spectrum of **HD**|STA **73882**|STA is derived from time-tagged observations over the course of 8 orbits on **1999**|DAT **Oct**|DAT **30**|DAT . Several " burst " events occurred during the observation (**Sahnow**|PER et al. **2000**|DAT) . We excluded all **photon**|PART events that occurred during the bursts , reducing effective on-target integration time from **16.8**|DUR **ksec**|DUR to **16.1**|DUR **ksec**|DUR . Strong interstellar extinction and lack of co-alignment of the SiC channels with the LiF channels prevented the collection of useful data shortward of **1010**|WAV **Å**|WAV .

Figure 1: An extract from our final corpus, originally from astro-ph/0005090.

comma, and reattaching the LaTeX which the tokenizer split off incorrectly.

4.5 The Corpus

The articles for the corpus were selected randomly from the downloaded LaTeX documents and annotated by the second author. The annotation was performed using a custom Emacs mode which provided syntax highlighting for named entity types and mapped the keys to specific named entities to make the annotation as fast as possible. The average annotation speed was 165 tokens per minute. An extract from our corpus is shown in Figure 1. There are a total of 7840 sentences in our corpus, with an average of 26.1 tokens per sentence.

5 Named Entity Categories

Examples of the categories we used are listed in Table 2. We restricted ourselves to high level categories such as STAR and GALAXY rather than detailed ones typically used by astronomers to classify objects such as red giant and elliptical galaxy.

5.1 Areas of Ambiguity

There were some entities that did not clearly fit into one specific category or are used in a way that is ambiguous. This section outlines some of these cases.

Temperature and Energy Due to the high temperatures in X-ray astronomy, temperatures are conventionally referred to in units of energy (eV), for example:

> its 1 MeV temperature, the emission from...

Our annotation stays *consistent to the units*, so these cases are tagged as energies (EGY).

Angular distance Astronomers commonly refer to angular distances on the sky (in units of arc) because it is not possible to know the true distance between two objects without knowing their redshift. We annotate these according to the units, i.e. angles, although they are often used in place of distances.

Spectral lines and ions Absorption or emission of radiation by ions results in *spectral line features* in measured spectra. Common transitions have specific names (e.g. Hα) whereas others are referred to by the ion name (e.g. Si IV), introducing ambiguity.

5.2 Comparison with GENIA and MUC

The corpus has a named entity density of 5.4% of tokens. This is significantly higher than the density of the Astronomy Bootstrapping Corpus. The most frequency named entities types are: PER (1477 tags), DAT (1053 tags), TEL (867 tags), GAL (551 tags), and WAV (451 tags). The token 10 has the highest degree of ambiguity since it was tagged with every unit related tag: EGY, MASS, etc. and also as OBJ.

By comparison the GENIA corpus has a much higher density of 33.8% tokens on a sample the same size as our corpus. The highest frequency named entity types are: OTHER (16171 tags), PROTEIN (13197 tags), DNA DOMAIN (6848 tags) and PROTEIN FAMILY (6711 tags).

The density of tags in MUC is 11.8%, higher than our corpus but much lower than GENIA. The highest frequency named entities are ORGANISATION (6373 tags), followed by LOCATION (3828 tags) and DATE (3672 tags). Table 3 gives a statistical comparison of the three corpora. This data suggests that the astronomy data will be harder to automatically tag than MUC 7 because the density is lower and there are many more classes. However, if there were more classes or finer grained distinctions in MUC this would not necessarily be true. It also demonstrates how different biological text is to other scientific domains.

Class	Definition	Examples	Comments
GXY	galaxy	NGC 4625; Milky Way; Galaxy	inc. black holes
NEB	nebula	Crab Nebula; Trapezium	
STA	star	Mira A; PSR 0329+54; Sun	inc. pulsars
STAC	star cluster	M22; Palomar 13	
SUPA	supernova	SN1987A; SN1998bw	
PNT	planet	Earth; Mars ; HD 11768 b; tau Boo	inc. extra-solar planets
FRQ	frequency	10 Hz; 1.4 GHz	
DUR	duration	13 seconds; a few years	inc. ages
LUM	luminosity	10^{46} ergs^{-1}; 10^{10}L$_{\odot}$	inc. flux
POS	position	17:45.6; -18:35:31; $17^h12^m13^s$	
TEL	telescope	ATCA; Chandra X-ray observatory	inc. satellites
ION	ion	Si IV; HCO^+	inc. molecular ions
SUR	survey	SUMSS; 2 Micron All-Sky Survey	
DAT	date	2003; June 17; 31st of August	inc. epochs (e.g. 2002.7)

Table 2: Example entity categories.

Corpus	ASTRO	GENIA	MUC
# cats	43	36	8
# entities	10 744	40 548	11 568
# tagged	16,016	69 057	19 056
# avg len	1.49	1.70	1.64
tag density	5.4%	33.8%	11.8%

Table 3: Comparison with GENIA and MUC.

Condition	Contextual predicate
$f(w_i) < 5$	X is prefix of w_i, $\|X\| \leq 4$
	X is suffix of w_i, $\|X\| \leq 4$
	w_i contains a digit
	w_i contains uppercase character
	w_i contains a hyphen
$\forall w_i$	$w_i = X$
	$w_{i-1} = X$, $w_{i-2} = X$
	$w_{i+1} = X$, $w_{i+2} = X$
$\forall w_i$	$\text{POS}_i = X$
	$\text{POS}_{i-1} = X$, $\text{POS}_{i-2} = X$
	$\text{POS}_{i+1} = X$, $\text{POS}_{i+2} = X$
$\forall w_i$	$\text{NE}_{i-1} = X$
	$\text{NE}_{i-2}\text{NE}_{i-1} = XY$

Table 4: Baseline contextual predicates

6 Maximum Entropy Tagger

The purpose of creating this annotated corpus is to develop a named entity tagger for astronomy literature. In these experiments we adapt the C&C NE tagger (Curran and Clark, 2003) to astronomy literature by investigating which feature types improve the performance of the tagger. However, as we shall see below, the tagger can also be used to test and improve the quality of the annotation. It can also be used to speed up the annotation process by pre-annotating sentences with their most likely tag. We were also interested to see whether > 40 named-entity categories could be distinguished successfully with this quantity of data.

Condition	Contextual predicate
$f(w_i) < 5$	w_i contains period/punctuation
	w_i is only digits
	w_i is a number
	w_i is {upper,lower,title,mixed} case
	w_i is alphanumeric
	length of w_i
	w_i has only Roman numerals
	w_i is an initial (X.)
	w_i is an acronym (ABC, A.B.C.)
$\forall w_i$	memory NE tag for w_i
	unigram tag of w_{i+1}, w_{i+2}
$\forall w_i$	w_i, w_{i-1} or w_{i+1} in a gazetteer
$\forall w_i$	w_i not lowercase and $f_{\text{lc}} > f_{\text{uc}}$
$\forall w_i$	uni-, bi- and tri-grams of word type

Table 5: Contextual predicates in final system

The C&C NE tagger feature types are shown in Tables 4 and 5. The feature types in Table 4 are the same as used in MXPost (Ratnaparkhi, 1996) with the addition of the NE tag history features. We call this the *baseline* system. Note, this is not the baseline of the NE tagging task, only the baseline performance for a Maximum Entropy approach.

Table 5 includes extra feature types that were tested by Curran and Clark (2003). The *w_i is only digits* predicates apply to words consisting of all digits. Title-case applies to words with an initial uppercase letter followed by lowercase (e.g. `Mr`). Mixed-case applies to words with mixed lower- and uppercase (e.g. `CitiBank`). The length features encode the length of the word from 1 to 15 characters, with a single bin for lengths greater than 15.

The next set of contextual predicates encode extra information about NE tags in the current context. The memory NE tag predicate records the NE tag that was most recently assigned to the current word. This memory is reset at

N	Word	Correct	Tagged
23	OH	MOL	NONE
14	rays	PART	NONE
8	GC	GXYP	NONE
6	cosmic	PART	NONE
6	HII	ION	NONE
5	telescope	TEL	NONE
5	cluster	STAC	NONE
5	and	LUM	NONE
4	gamma	NONE	PART

Table 6: Detected Errors and Ambiguities

Experiment	P	R	F-score
BASELINE	93.0	82.5	87.5
EXTENDED	91.2	82.4	86.6
-MEMORY	92.1	84.3	88.0
-MEMORY/POS	92.3	83.9	87.9
COARSE BASE	92.6	86.7	89.5
COARSE EXTENDED	93.0	88.9	90.9

Table 7: Feature experiment results

the beginning of each document. The unigram predicates encode the most probable tag for the next words in the window. The unigram probabilities are relative frequencies obtained from the training data. This feature enables us to know something about the likely NE tag of the next word before reaching it.

Another feature type encodes whether the current word is more frequently seen in lowercase than title-case in a large external corpus. This is useful for disambiguating beginning of sentence capitalisation. Eventually the frequency information will come from the raw astronomy corpus itself.

Collins (2002) describes a mapping from words to *word types* which groups words with similar orthographic forms into classes. This involves mapping characters to classes and merging adjacent characters of the same type. For example, `Moody` becomes `Aa`, `A.B.C.` becomes `A.A.A.` and `1,345.05` becomes `0,0.0`. The classes are used to define unigram, bigram and trigram contextual predicates over the window. This is expected to be a very useful feature for scientific entities.

7 Detecting Errors and Ambiguities

We first trained the C&C tagger on the annotated corpus and then used this model to retag the corpus. We then compared this retagged corpus with the original annotations. The differences were manually checked and corrections made where necessary.

Table 6 shows the most frequent errors and ambiguities detected by this approach. Most of the differences found were either the result of genuine ambiguity or erroneous annotation.

GC, cosmic and HII are examples of genuine ambiguity that is difficult for the tagger to model correctly. GC means globular cluster which is not tagged, but less often refers to the Galactic Centre which *is* tagged (GXYP). cosmic occurs in two contexts: as part of cosmic ray(s) which is tagged as a particle; and in expressions such as cosmic microwave background radiation which is not tagged. HII is used most frequently in reference to HII ions and hence is tagged as an (ION). However, occasionally HII is used to refer to HII galaxies and not tagged.

OH and gamma rays are examples where there was some inconsistency or error in some of the annotated data. In both of these cases instances in the corpus were not tagged.

We also implemented the approach of Dickinson and Meurers (2003) for identifying annotation errors in part of speech (POS) tagging. Their approach finds the longest sequence of words that surround a tagging ambiguity. The longer the context, the more likely the ambiguity is in fact an annotation error. This approach identified a number of additional errors, particularly annotation errors within entities. However, many of the errors we may have found using this technique were already identified using the tagging described above.

8 Inter-annotator Agreement

To test the reliability of the annotations we performed two tests. Firstly, we asked an astronomy PhD student to take our annotation guidelines and annotate around 30,000 words (15% of the corpus). Secondly, the second author also reannotated a different 30,000 words about 2-months after the original annotation process to check for self consistency.

We used the kappa statistic (Cohen, 1960) to evaluate inter-annotator reliability. The kappa value for agreement with the PhD student annotation was 0.91 on all tags and 0.82 not including the NONE tags. Given that the annotation guidelines were not as complete as we would have liked, this agreement is very

good. The kappa value for agreement with the reannotated corpus was 0.96 on all tags and 0.92 not including the NONE tags.

When the differences between the 30,000 word sections and the original corpus were check manually (by the second author and the PhD student) practically all of them were found to be annotation errors rather than genuine ambiguity that they could not agree on.

9 Results

We split the corpus into 90% training and 10% testing sets. For our final results we performed 10-fold cross validation. For the experiments analysing the contribution of named entity feature types from the C&C tagger we used one of the 10 folds. The evaluation was performed using the CoNLL shared task evaluation script[1].

9.1 Feature Experiments

The results of the feature experiments are shown in Table 7. The Maximum Entropy baseline performance of 87.5% F-score is very high given the large number of categories. Clearly there is enough contextual information surrounding the entities that they can be fairly reliably tagged.

A surprising result is that using all of the additional features which helped significantly improve performance on newswire actually damages performance by $\sim$ 1%. Further experimental analysis with removing specific feature types found that the offending feature was the last tagged with tag x feature (the *memory* feature). Removing this feature improves performance a little bit more giving our best result of 88.0% F-score. We believe this feature performs particularly badly on numeric expressions which are part of many different named entity classes which may appear with the same word in a single article.

We experimented with removing the POS tag features since the POS tagger performed very badly on astronomy text, but this made little difference. We have experimented with removing the other feature types listed in Table 5 but this resulted in a small decrease in performance each time.

This demonstrates that with new training data it is fairly straightforward to achieve reasonable performance in identifying astronomy named entities.

[1] http://www.cnts.ua.ac.be/conll2003/ner/bin/

Category	Constituent categories
galaxy	GXY, GXYP, GXYC, NEBP, NEB
star	STA, STAP, STAC, SUPA
object	OBJ, OBJP, EVT
sso	PNT, PNTP, MOO, MOOP
units	FRQ, WAV, DIST, TEMP, DUR, MASS, ANG, LUM, VEL, PCT, EGY, UNIT, POS
inst.	TEL, INST
particle	PART, ELEM, MOL, ION, LN
person	PER, ORG, URL
location	LOC
obs.	SUR, CAT, DB
date	DAT, TIME
software	CODE

Table 8: Coarse-grained mapping

9.2 Coarse-grained categories

One interesting property of our named entity corpus is the very large number of categories relative to existing NE corpora such as MUC. To test what impact the number of classes has on performance we repeated the experiment described above, using coarser-grained named entity categories based on the mapping shown in Table 8.

The course grained classifier achieves an F-score of 89.5% using the baseline feature set and an F-score of 90.9% using the extended feature set without the memory feature. The key difference between the fine and coarse grained results is the significantly better recall on coarse grained classes.

10 Conclusion

This is a pilot annotation of astronomy texts with named entity information. Now that we have created the initial corpus we intend to reevaluate the categories, aiming for greater consistency and coverage of the entities of interest in the corpus.

We have performed preliminary experiments in training taggers using our corpus. These experiments have produced very promising results so far (87.8% F-score on 10-fold cross validation). We intend to extend our evaluation of individual features for scientific text and add features that exploit online astronomy resources.

This paper has described in detail the process of creating a named entity annotated corpus of astronomical journal articles and conference papers. This includes translating the

LaTeX typesetting information into a useable format. Unlike existing work we have rendered the mathematics in Unicode text rather than just removing it, which is important for further analysis of the data. The resulting corpus is larger than existing resources, such as MUC, but has been annotated with a much more detailed set of over 40 named entity classes.

Finally, we have demonstrated that high accuracy named entity recognisers can be trained using the initial release of this corpus, and shown how the tagger can be used to iteratively identify potential tagging errors. The quality of the results should only improve as the corpus size and quality is increased.

11 Acknowledgments

We would like to thank the arXiv administrators for giving us access to the astrop-ph archive. This research has made use of NASA's Astrophysics Data System Bibliographic Services. This research has made use of the NASA/IPAC Extragalactic Database (NED) operated by the Jet Propulsion Laboratory, California Institute of Technology.

This research was funded under a University of Sydney Research and Development Grant and ARC Discovery grants DP0453131 and DP0665973.

References

ADS. 2005. Astronomical Data Service. `http://www.adsabs.harvard.edu/`.

arXiv. 2005. arXiv.org archive. `http://arxiv.org`.

M. Becker, B. Hachey, B. Alex, and C. Grover. 2005. Optimising selective sampling for bootstrapping named entity recognition. In *Proceedings of the ICML Workshop on Learning with Multiple Views*, pages 5–11, Bonn, Germany.

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

M. Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 489–496, Philadephia, PA USA.

J.R. Curran and S. Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proceedings of the 7th Conference on Natural Language Learning (CoNLL)*, pages 164–167, Edmonton, Canada.

M. Dickinson and W.D. Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 107–114, Budapest, Hungary.

P. Ginsparg. 2001. Creating a global knowledge network. In *UNESCO Expert Conference on Electronic Publishing in Science*, Paris, France.

J. Grimm. 2003. Tralics, a LaTeX to XML translator. *TUGboat*, 24(3):377 – 388.

B. Hachey, B. Alex, and M. Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the 9th Conference on Natural Language Learning (CoNLL)*, pages 144–151, Ann Arbor, MI USA.

R. J. Hanisch and P. J. Quinn. 2005. The IVOA. `http://www.ivoa.net/pub/info/`.

S. Harabagiu, D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morărescu. 2000. Falcon: Boosting knowledge for answer engines. In *Proceedings of TREC-9*.

L. Hirschman and R. Gaizauskas. 2001. Natural language question answering: The view from here. *Journal of Natural Language Engineering*, 7(4):275–300.

L. Hirschman, J.C. Park, J. Tsujii, L. Wong, and C.H. Wu. 2002. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18(12):1553–1561.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(s1):i180–i182.

M.-C. Lortet, S. Borde, and F. Ochsenbein. 1994. Second Reference Dictionary of the Nomenclature of Celestial Objects. *A&AS*, 107:193–218, October.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

A. P. Martinez, S. Derriere, N. Gray, R. Mann, J. McDowell, T. McGlynn, Ochsenbein F., P. Osuna, G. Rixon, and R. Williams. 2005. The UCD1+ controlled vocabulary Version 1.02.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA USA.

J.C. Reynar and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 16–19, Washington DC, USA.

Daniel Trinkle. 2002. Detex. `http://www.cs.purdue.edu/homes/trinkle/detex/`.

Unicode Consortium. 2005. *The Unicode Standard*. Addison-Wesley, 4th edition.

Die Morphologie (f) : Targeted Lexical Acquisition for Languages other than English

Jeremy Nicholson†, Timothy Baldwin†, and Phil Blunsom‡
†‡ Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
and
† NICTA Victoria Research Laboratories
University of Melbourne, VIC 3010, Australia
{jeremymn,tim,pcbl}@csse.unimelb.edu.au

Abstract

We examine standard deep lexical acquisition features in automatically predicting the gender of noun types and tokens by bootstrapping from a small annotated corpus. Using a knowledge-poor approach to simulate prediction in unseen languages, we observe results comparable to morphological analysers trained specifically on our target languages of German and French. These results describe further scope in analysing other properties in languages displaying a more challenging morphosyntax, in order to create language resources in a language-independent manner.

1 Introduction

As a result of incremental annotation efforts and advances in algorithm design and statistical modelling, deep language resources (DLRs, i.e. language resources with a high level of linguistic sophistication) are increasingly being applied to mainstream NLP applications. Examples include analysis of semantic similarity through ontologies such as WordNet (Fellbaum, 1998) or VerbOcean (Chklovski and Pantel, 2004), parsing with precision grammars such as the DELPH-IN (Oepen et al., 2002) or PARGRAM (Butt et al., 2002) grammars, and modelling with richly annotated treebanks such as PropBank (Palmer et al., 2005) or CCGbank (Hockenmaier and Steedman, 2002).

Unfortunately, the increasing complexity of these language resources and the desire for broad coverage has meant that their traditionally manual mode of creation, development and maintenance has become infeasibly labour-intensive. As a consequence, deep lexical acquisition (DLA) has been proposed as a means of automatically learning deep linguistic representations to expand the coverage of DLRs (Baldwin, 2005). DLA research can be divided into two main categories: targeted DLA, in which lexemes are classified according to a given lexical property (e.g. noun countability, or subcategorisation properties); and generalised DLA, in which lexemes are classified according to the full range of lexical properties captured in a given DLR (e.g. the full range of lexical relations in a lexical ontology, or the system of lexical types in an HPSG).

As we attest in Section 2, most work in deep lexical acquisition has focussed on the English language. This can be explained in part by the ready availability of targeted language resources like the ones mentioned above, as well as secondary resources — such as corpora, part-of-speech taggers, chunkers, and so on — with which to aid prediction of the lexical property in question. One obvious question, therefore, is whether the techniques used to perform lexical acquisition in English generalise readily to other languages, where subtle but important differences in morphosyntax might obfuscate the surface cues used for prediction.

In this work, we will examine the targeted prediction of the gender of noun types and tokens in context, for both German and French. As an example, the following phrases display adjectival gender inflection in two of the three languages below.

> the open window
> das offene Fenster
> la fenêtre ouverte

window has no gender in English, *Fenster* is neuter in German and *fenêtre* is feminine in French. So

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 67–74.

in English, neither the determiner *the* or the adjective *open* inflect, whereas *das* and *la* are the neuter and feminine forms respectively of the determiner, and *offen* and *ouvert* take the neuter and feminine respective suffixes of *-e*.

On the face of it, the task is remarkably simple: native speakers can achieve near-perfection, and even the accuracy of simplistic morphological analysers is taken for granted. However, both of these rely on significant knowledge of the inflectional morphosyntax of the language, whereas we will take a knowledge-poor approach, and bootstrap from an annotated corpus. An additional motivation for automating gender learning is that we are interested in semi-automated precision grammar development over the full spectrum of languages, from the highest to the lowest density. Given that there is no guarantee we will be able to access a native speaker for a low–density language, automation is the natural course to take. Even if we do have access to a native speaker, we would like to maximise use of their time, and free them up from annotation tasks which we can hope to perform reliably through automatic means.

The knowledge-poor approach is an interesting one — although the features used for prediction are linguistically motivated, we remain agnostic toward a specific target language. Since no language-specific features are being used, the knowledge-poor approach is presumed to generalise over unseen languages, as long as there is consistent, well-defined morphosyntax within it.

Despite its seeming ease, and our examination of gender as a "black-box" learning feature, having gender information is extrinsically valuable in many contexts. For example, natural language generation and machine translation both rely heavily on knowing the gender of a word for accurate inflectional agreement.

The structure of the remainder of this paper is as follows. Section 2 provides a background for deep lexical acquisition and gender prediction. Section 3 describes the language resources of which we made use, and Section 4 details the feature set. Finally, we evaluate our method in Section 5, and supply a discussion and brief conclusion in Sections 6 and 7.

2 Background

2.1 Deep Lexical Acquisition

As mentioned above, DLA traditionally takes two forms: targeted toward a specific lexical property, or generalised to map a term to an amalgam of properties defined for a given resource.

The latter technique is often construed as a classification task where the classes are the lexical categories from the target resource. One example is extending an ontology such as WordNet (e.g. Pantel (2005), Daudé et al. (2000)). Another is learning the categories for the lexicon of a precision grammar, such as the English Resource Grammar (ERG; Flickinger (2002);Copestake and Flickinger (2000)), as seen in Baldwin (2005). A common tool for this is supertagging, where the classes are predicted in a task analogous to part-of-speech (POS) tagging (Clark, 2002; Blunsom and Baldwin, 2006).

The former technique is exemplified by expert systems, where the target language is occasionally not English. These learn properties such as verb subcategorisation frames (e.g. Korhonen (2002) for English or Schulte im Walde (2003) for German) or countability (e.g. Baldwin and Bond (2003) for English, or van der Beek and Baldwin (2004) for Dutch, with a crosslingual component).

Common methodologies vary from mining lexical items from a lexical resource directly (e.g. Sanfilippo and Poznanski (1992) for a machine–readable dictionary), learning a particular property from a resource to apply it to a lexical type system (e.g. Carroll and Fang (2004) for verb subcategorisation frames), restricting possible target types according to evidence, and unifying to a consolidated entry (e.g. Fouvry (2003) for precision grammar lexical types), or applying the lexical category of similar instances, based on some notion of similarity (e.g. Baldwin (2005), also for lexical types). It is this last approach that we use in this work.

Implicit in all of these methods is a notion of the secondary language resource (LR). While the primary LR is the (actual or presumed) resource whose types are targeted by the DLA, a secondary LR is an available resource that can be used to aid acquisition. Common examples, as mentioned above, are corpora, POS taggers, and chunkers. Secondary LRs of varying degrees of complexity are available for some languages; however we examine primarily simple LRs in order to remain

faithful to lower–density languages.

2.2 Gender Prediction

Gender is a morphosemantic phenomenon observed in many Indo-European languages. It is observed generally in three classes: masculine, feminine, or neuter (French has masculine and feminine only). Whereas the only English words that inflect for gender are pronouns, in most Indo-European languages at least nouns, adjectives, and determiners also inflect. This normally occurs by way of suffixation, but some languages, such as Swahili, use prefixation.

Gender appears to be a purely semantic property which is determined based on the underlying shape, manner, or sex of the referent. However, morphology can play a strong role, by way of gender selection according to the morphemes of the wordform. A classic example is *Mädchen* "girl" in German, which is neuter because words with the *-chen* suffix are neuter, despite the obvious feminine semantic connotations of this instance.

The contextual and morphological effects shown in the "open window" example above have been theorised as priming the gender predictions of language users when confronted with unseen words.[1] When contextual or lexicographic information is available for a language, this is usually a reliable method for the prediction of gender. Consequently, automatic prediction of gender in languages which have inflectional morphology is usually seen as the domain of the POS tagger (such as Hajič and Hladká (1998)), or morphological analyser (e.g. GERTWOL (Haapalainen and Majorin, 1995) for German and FLEMM (Namer, 2000) for French).

One work in automatic gender prediction that is similar to this one is the bootstrapping approach of Cucerzan and Yarowsky (2003). Starting with a seed set of nouns whose gender is presumably language–invariant, they mine contextual features to hypothesise the gender of novel instances. They then extract simple morphological features of their larger predicted set, and use these to predict the gender of all nouns in their corpora.

The major differences between this work and our own are in the approach Cucerzan and Yarowsky use, and the classes that they can handle. First, their semi-automatic approach relies on a bilingual dictionary from which to extract the seeds — if a machine readable one does not exist, they annotate the seeds by hand. Our approach is fully automatic and can act with an arbitrary set of seeds (although an arbitrarily pathological set of seeds would perform arbitrarily poorly). Second, their method is only well-defined for predicting gender in languages with only masculine and feminine, as they do not propose canonical neuter noun candidates. Our approach makes no claims on the number or underlying semantics of genders in a language, and can equally be extended to predict other morphosyntactic properties such as case and number, where canonical forms are poorly defined.

3 Secondary Language Resources

We used a number of secondary language resources: most notably annotated and unannotated corpora, as well as inflectional lexicons and a POS tagger.

3.1 Corpora

Our primary data sources were two corpora: the TIGER treebank[2] (Brants et al., 2002) for German and the BAF corpus[3] (Simard, 1998) for French.

TIGER is a corpus of about 900K tokens of German newspaper text from the Frankfurt Rundschau, semi-automatically annotated for lemma, morphology, POS and syntactic structure.

The BAF corpus is a bilingual French–English collection of eleven documents comprising Canadian government proceedings, machine translation technical documents, and a Jules Verne novella. There are about 450K sentence-aligned French tokens, with no annotation of morphology or syntax. This corpus is heavily domain–specific, and the lack of annotation provides particular problems, which we explain below. Note that we make no use of the English component of BAF in this paper.

3.2 Inflectional Lexicons

Whereas our German corpus has gold-standard judgements of gender for each token, the French corpus has no such information. Consequently, we use a semi-automatic method to match genders to nouns. Using the Le*fff* syntactic lexi-

[1] See Tucker et al. (1977), among others, for detailed studies of L1 and L2 gender acquisition.

[2] http://www.ims.uni-stuttgart.de/projekte/TIGER

[3] http://rali.iro.umontreal.ca/Ressources/BAF

con[4] (Sagot et al., 2006) and Morphalou[5] (Romary et al., 2004), a lexicon of inflected forms, we automatically annotate tokens for which the sources predict an unambiguous gender, and hand-annotate ambiguous tokens using contextual information. These ambiguous tokens are generally animate nouns like *collègue*, which are masculine or feminine according to their referent, or polysemous nouns like *aide*, whose gender depends on the applicable sense.

3.3 POS Taggers

Again, TIGER comes annotated with hand-corrected part-of-speech tags, while the BAF does not. For consistency, we tag both corpora with TreeTagger[6] (Schmid, 1994), a decision tree–based probabilistic tagger trained on both German and French text. We were interested in the impact of the accuracy of the tagger compared to the corrected judgements in the corpus as an extrinsic evaluation of tagger performance. The tagger token accuracy with respect to the TIGER judgements was about 96%, with many of the confusion pairs being common nouns for proper nouns (as the uniform capitalisation makes it less predictable).

4 Deep Lexical Acquisition

We use a deep lexical acquisitional approach similar to Baldwin (2005) to predict a lexical property. In our case, we predict gender and restrict ourselves to languages other than English.

Baldwin examines the relative performance on predicting the lexical types in the ERG, over various linguistically-motivated features based on morphology, syntax, and an ontology: the so-called "bang for the buck" of language resources.

To take a similar approach, we extract all of the common nouns (labelled NN), from each of the corpora to form both a token and a type data set. We generate our feature set independently over the token data set and the type data set for both morphological and syntactic features (explained below) without feature selection, then perform 10-fold stratified cross-validation using a nearest-neighbour classifier (TiMBL 5.0: Daelemans et al. (2003)) with the default $k = 1$ for evaluation.

A summary of the corpora appears in Table 1.

Corpus	Tokens	NN Tokens	NN Types
TIGER	900K	180K	46K
BAF	450K	110K	8K

Table 1: A summary of the two corpora: TIGER for German and BAF for French. The comparatively low number of noun types in BAF is caused by domain specificity.

4.1 Morphological DLA

Morphology-based deep lexical acquisition is based on the hypothesis that words with a similar morphology (affixes) have similar lexical properties. Using character n-grams is a simple approach that does not require any language resources other than a set of pre-classified words from which to bootstrap.

For each (token or type) instance, we generate all of the uni-, bi-, and tri-grams from the word-form, taken from the left (prefixes and infixes) and the right (suffixes and infixes), padded to the length of the longest word in the data. For example, the 1-grams for *fenêtre* above would be *f, e, n, ê, t, r, e, #, #, ...* from the left (L) and *e, r, t, ê, n, e, f, #, #, ...* from the right (R).

We evaluate using each of 1-, 2-, and 3-grams, as well as the combination of 1- and 2-grams, and 1-, 2-, and 3-grams from L and R and both (LR) — to make $5 \times 3 = 15$ experiments.

Other resources for morphological analysis exist, such as derivational morphological analysers, lemmatisers, and stemmers. We do not include them, or their information where it is available in our corpora, taking the stance that such tools will not be readily available for most languages.

4.2 Syntactic DLA

Syntax-based deep lexical acquisition purports that a lexical property can be predicted from the words which surround it. Most languages have at least local morphosyntax, meaning that morphosyntactic and syntactico-semantic properties are attested in the surrounding words.

For each token, we examine the four word forms to the left and right, the POS tags of these word-forms, and corresponding bi-word and bi-tag features according to (Baldwin, 2005). For each type, we take the best N of each feature across all of the relevant tokens.

We examine both left (preceding, in languages written left–to–right) and right (following) context

[4] http://www.lefff.net

[5] http://www.cnrtl.fr/morphalou

[6] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

to maintain a language agnostic approach. While in general, contextual gender information is encoded in the noun modifiers, it is unclear whether these modifiers precede the noun (as in head–final languages like English or German), follow the noun, or occur in some combination (as in French).

While context in freer–word–order languages such as German is circumspect, limiting our feature set to four words on each side takes into account at least local agreement (which is a fixture in most languages). Other unrelated information can presumably be treated as noise.

A summary of the feature set appears in Table 2.

4.3 Ontological DLA

We do not make use of an ontology in the way that Baldwin does; although a candidate ontology does exist for these particular languages (EuroWordNet; Vossen (1998)), the likelihood of such as resource existing for an arbitrary language is low.

5 Evaluation

We evaluate each of the feature sets across the four data sets collected from the two corpora: the German NN-tagged tokens in TIGER, the German NN-tagged types in TIGER, the French NN-tagged tokens in BAF and the French NN-tagged types in BAF. There were four lexical types for German: MASC, FEM, NEUT, and * and three lexical types for French: MASC, FEM, *. The instances labelled * were those to which a gender could not be sensibly ascribed (e.g. abbreviations such as *PGs*), or uniquely defined (e.g. gender-underspecified nouns such as *relâche*).

Baseline accuracy for each set corresponds to the majority–class: for German tokens and types, this was FEM (41.4 and 38.4% respectively); for French tokens and types, this was MASC (51.8 and 52.7%).

5.1 Morphology

The results for the morphological features are shown in Table 3, for 1, 2, and 3-grams taken from the left and right.

The performance over tokens is excellent, comparable to that of language-specific morphological analysers. Taking characters from the right unsurprisingly performs best, as both German and French inflect for gender using suffixes.

	German		***French***	
	token	***type***	***token***	***type***
L1	93.8	77.0	99.4	85.5
L2	93.6	77.0	99.4	88.0
L3	93.4	73.5	99.4	86.0
L1+2	93.8	77.3	**99.5**	87.5
L1+2+3	93.6	75.1	99.4	87.3
R1	97.1	85.3	**99.5**	87.9
R2	**97.4**	**86.6**	**99.5**	**88.4**
R3	96.9	84.2	99.4	85.3
R1+2	**97.4**	86.5	**99.5**	**88.4**
R1+2+3	97.3	85.9	**99.5**	87.5
LR1	95.6	78.5	99.4	85.5
LR2	96.2	82.0	99.4	86.1
LR3	95.7	78.9	99.4	85.7
LR1+2	96.1	81.6	99.4	86.1
LR1+2+3	96.0	80.9	99.4	85.5

Table 3: Morphological results using TiMBL

	German		***French***	
	token	***type***	***token***	***type***
All	82.2	52.5	—	—
TT POS	81.7	52.1	95.5	66.6
WF only	**84.9**	**53.9**	**96.6**	**67.6**

Table 4: Syntactic results using TiMBL

The best results invariably occurred when using bigrams, or unigrams and bigrams together, suggesting that gender is encoded using more than just the final character in a word. This is intuitive, in that a 1-letter suffix is usually insufficient evidence; for example, *-n* in French could imply a feminine gender for words like *maison, information* and a masculine gender for words like *bâton, écrivain*.

5.2 Syntax

The syntax results shown in Table 4 show the gold-standard POS tags (ALL) against those estimated by TreeTagger (TT POS), when combined with wordform context. We also contrast these with using the wordforms without the part-of-speech features (WF only). For type results, we took the best N features across corresponding tokens — for consistency, we let $N = 1$, i.e. we considered the best contextual feature from all of the tokens.[7]

[7]Experimentally, a value of 2 gave the best results, with a constant decrease for larger N representing the addition of ir-

Feature type	*Positions/description*
MORPHOLOGY	
Left	1-, 2-, 3-, 1+2-, 1+2+3-grams
Right	1-, 2-, 3-, 1+2-, 1+2+3-grams
Left/Right	1-, 2-, 3-, 1+2-, 1+2+3-grams
SYNTAX	
Wordform	$-4, -3, -2, -1, +1, +2, +3, +4$
POS tag	$-4, -3, -2, -1, +1, +2, +3, +4$
Bi-Word	$(-3, -2), (-3, -1), (-2, -1), (+1, +2), (+1, +3), (+2, +3)$
Bi-Tag	$(-4, -1), (-3, -2), (-3, -1), (-2, -1), (+1, +2), (+1, +3), (+1, +4), (+2, +3)$

Table 2: Morphological (n-gram) and syntactic (contextual) features.

Both token-wise and type-wise results are much poorer than the ones observed using morphological features. This is unsurprising, firstly because gender is primarily a morphological feature, and is encoded in syntax only through inflection of contextual wordforms. Also, often contextual evidence for gender is weak — for example, nouns beginning with a vowel in French do not take the canonical *le, la*, only *l'* (e.g. *l'ami* (m)); similarly, plural words in German do not inflect for gender: i.e. instead of taking *der, die, das*, plural nouns only take *die* (e.g. *die Freunde* (m)).

In fact, gender is so much a morphological feature that removing the part-of-speech features uniformly improves results. Again, this is unsurprising, seeing as the contextual parts of speech impact only weakly on gender preferences.[8] We return to discuss POS features below.

6 Discussion

The application of language–inspecific features to the task of gender prediction was quite successful, with both morphological and syntactic features comfortably outperforming both the type and token baselines in both German and French. This, however, is not a stunning achievement, as a rule–based system built by hand in a few minutes by a native speaker of the target language could also boast such claims.

The morphological features, based on character n-grams, performed much better than the syntactic features, based on contextual wordform and part-of-speech features. This is validated by the natural claim that the morphemes and semantic gender of a word are somehow linked more strongly than gender to its syntactic properties. Capturing the clues in adjectival and determiner inflection, shown in various examples above, is more challenging for an automatic system.[9]

We attest that the observed improvement in performance between gender prediction in German and French, especially at the token level, is not an indication of a simpler task in French, only a domain–specific corpus. While TIGER is a treebank of newspaper data, the BAF is a small number of topical texts, with little variability.

This specificity perhaps is best evinced through the ratio of tokens to types: for German there are approximately 4 tokens/type, for French, this number balloons to almost 14, despite the two corpora being of the same relative size. Having a large number of exemplars in the training split will almost certainly bias prediction, as the gold-standard tag is usually known. There is minimal false evidence: the proportion of multi-gendered types is only about 3%.

Consequently, the results over noun types are more interesting than those over noun tokens, as they smooth to some extent the multiple corpus instances and domain effects. For types, morphological features taken from the left are still much better in French than German, but those taken from the right give similar results. Syntactic features are consistently better as well. It would be interesting to contrast these with results taken from a French treebank, to parallelise the results for German.

As mentioned above, using bigrams or a combination of unigrams and bigrams generally gives

relevant features. A more generous match over any of the corresponding features might alleviate this problem somewhat.

[8] Contextual parts-of-speech generally are uninformative for gender: consider whether masculine, feminine, or neuter nouns are more likely to be followed by a finite verb. This is not exclusively the case, however, as subcategorisation frames are occasionally influenced by gender (e.g. the propensity of deverbal nouns ending in *-tion* (f) to take a *de*-complement); we saw no evidence of this in our data, however.

[9] Approaches like pertinence (Turney, 2006) using a very large corpus could help to mine contextual features for an unknown language. Of course, the probability of having a very large corpus for an unknown language is low, so the problem is somewhat circular.

the best performance for morphological features. This contrasts with the approach taken by Cucerzan and Yarowsky (2003), who extract unigram suffix morphological features. We hypothesise that having longer features may give this technique better discrimination, although this remains to be seen for other languages.

It is not surprising that suffix morphological features perform better than those based on prefixes for these languages, what is surprising is that the morphological features taken from the left work at all. On a token level, this can be partially explained by instances of exact matches occurring in the training data. On the type level, we surmise that there is enough training data for the classifier to accurately predict gender according to instances of uniform length. This hypothesis is supported by reducing the cross-validation split to 10%-90% (effectively simulating a low-density language); for German, unigrams from the left drop to 56% accuracy, while unigrams from the right only fall to 75%.

While the poor performance of the syntactic features leads us to conclude that they are unreliable for this particular task, they may still be fruitful in extending this approach to other lexical properties. A similar morphosyntactic property to gender is case, but this is more heavily syntactic and a method based on morphology only is likely to struggle.

In retaining our syntactic features, we analyse the performance particularity of the POS tagger. While it has a 4% error rate over tokens, a drop of only a few tenths of a percentage is observed in place of gold-standard tags. With wordform context by itself performing better, having an accurate POS tagger seems an inefficient use of our resources, as it is only moderately available across target languages. However, there are syntactico-semantic properties such as countability and subcategorisation frames which rely on syntactic distinctions that are almost irrelevant for morphosyntactic phenomena (e.g. particle vs. preposition confusion). The downstream application of simplistic POS taggers remains to be seen for these tasks.

Obvious extensions to this work, as mentioned above, are making use of a French treebank, and examining other morphosyntactic lexical properties such as number and case, or syntactico-semantic properties such as countability and subcategorisation frames. Taking several simple morphosyntactic properties into account could lead to a language–independent morphological analyser.

Just as important, however, is an analysis of these types of properties (where they exist) for languages with a markedly different morphosyntax. Examples are complex case systems seen in Eastern European languages, agglutinative morphology such as in Turkish, or infixing as in several Austronesian languages.

Finally, the preponderance of data available in English (among other languages) makes cross–lingual deep lexical acquisition tempting. Similarly to Cucerzan and Yarowsky (2003), where a small bilingual dictionary exists, it seems possible to bootstrap from a high–volume data set to that of a smaller language, presumably by learning the underlying lexical semantics (e.g. the countability learning in van der Beek and Baldwin (2004)). One telling question, however, is the necessary "closeness" of the source and target language for this to be feasible.

7 Conclusion

We presented an analysis of standard deep lexical acquisition features in naively predicting gender automatically for German and French noun tokens and types. The morphological features performed comparably to analysers trained on the target languages, while the syntactic features provide scope for other morphosyntactic lexical features. This methodology could aid in construction of language resources in a language–independent manner.

References

Timothy Baldwin and Francis Bond. 2003. Learning the countability of English nouns from corpus data. In *Proc. of the 41st Annual Meeting of the ACL*, pages 463–470, Sapporo, Japan.

Timothy Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proc. of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, USA.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 164–171, Sydney, Australia.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proc. of the Workshop on Treebanks and Linguistic Theories (TLT02)*, Sozopol, Bulgaria.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proc. of the 2002 COLING Workshop on Grammar Engineering and Evaluation*, Taipei, Taiwan.

John Carroll and Alex Fang. 2004. The automatic acquisition of verb subcategorisations and their impact on the performance of an HPSG parser. In *Proc. of the 1st International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 107–114, Sanya City, China.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 33–40, Barcelona, Spain.

Stephen Clark. 2002. Supertagging for combinatorial categorial grammar. In *Proc. of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pages 19–24, Venice, Italy.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage english grammar using HPSG. In *Proc. of the Second conference on Language Resources and Evaluation*, Athens, Greece.

Silviu Cucerzan and David Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *Proc. of the 3rd International Conference on Human Language Technology Research and 4th Annual Meeting of the NAACL (HLT-NAACL 2003)*, pages 40–47, Edmonton, Canada.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2003. *TiMBL: Tilburg Memory Based Learner, version 5.0, Reference Guide*. ILK Technical Report 03-10.

Jordi Daudé, Lluis Padró, and German Rigau. 2000. Inducing ontological co-occurrence vectors. In *Proc. of the 38th Annual Meeting of the ACL*, pages 125–132, Ann Arbor, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*. CSLI Publications, Stanford, USA.

Frederik Fouvry. 2003. *Robust Processing for Constraint–Based Grammar Formalisms*. Ph.D. thesis, University of Essex, Colchester, UK.

Mariikka Haapalainen and Ari Majorin. 1995. GERTWOL und morphologische disambiguierung für das Deutsche. In *Proc. of the 10th Nordic Conference on Computational Linguistics*, Helsinki, Finland.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proc. of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics (COLING/ACL-98)*, pages 483–490, Montréal, Canada.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, Las Palmas, Spain.

Anna Korhonen. 2002. *Subcategorization Acquisition*. Ph.D. thesis, University of Cambridge, Cambridge, UK.

Fiametta Namer. 2000. FLEMM : un analyseur flexionnel du Français à base de règles. *Traitement Automatique des Langues*, 41:523–548.

Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors. 2002. *Collaborative Language Engineering. A Case Study in Efficient Grammar-Based Processing*. CSLI Publications, Stanford, USA.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Patrick Pantel. 2005. Inducing ontological co-occurrence vectors. In *Proc. of the 43rd Annual Meeting of the ACL*, pages 125–132, Ann Arbor, USA.

Laurent Romary, Susanne Salmon-Alt, and Gil Francopoulo. 2004. Standards going concrete: from LMF to Morphalou. In *Proc. of the COLING-2004 Workshop on Enhancing and Using Electronic Dictionaries*, pages 22–28, Geneva, Switzerland.

Benoît Sagot, Lionel Clément, Éric Villemonte de La Clergerie, and Pierre Boullier. 2006. The l*efff* syntactic lexicon for French: Architecture, acquisition, use. In *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 1348–1351, Genoa, Italy.

Antonio Sanfilippo and Victor Poznanski. 1992. The acquisition of lexical knowledge from combined machine–readable dictionary sources. In *Proc. of the 3rd Conference on Applied Natural Language Processing (ANLP 1992)*, pages 80–87, Trento, Italy.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proc. of the 1st International Conference on New Methods in Language Processing*, Manchester, UK.

Sabine Schulte im Walde. 2003. *Experiments on the Automatic Induction of German Semantic Verb Classes*. Ph.D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Sttutgart, Stuttgart, Germany.

Michel Simard. 1998. The BAF: A corpus of English and French bitext. In *Proc. of the 1st International Conference on Language Resources and Evaluation (LREC'98)*, Granada, Spain.

G. Richard Tucker, Wallace E. Lambert, and André Rigault. 1977. *The French speaker's skill with grammatical gender: an example of rule-governed behavior*. Mouton, The Hague, Netherlands.

Peter D. Turney. 2006. Expressing implicit semantic relations without supervision. In *Proc. of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 313–320, Sydney, Australia.

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.

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

Automatic Mapping Clinical Notes to Medical Terminologies

Jon Patrick, Yefeng Wang and Peter Budd
School of Information Technologies
University of Sydney
NSW 2006, Australia
{jonpat,ywang1,pbud3427}@it.usyd.edu.au

Abstract

Automatic mapping of key concepts from clinical notes to a terminology is an important task to achieve for extraction of the clinical information locked in clinical notes and patient reports. The present paper describes a system that automatically maps free text into a medical reference terminology. The algorithm utilises Natural Language Processing (NLP) techniques to enhance a lexical token matcher. In addition, this algorithm is able to identify negative concepts as well as performing term qualification. The algorithm has been implemented as a web based service running at a hospital to process real-time data and demonstrated that it worked within acceptable time limits and accuracy limits for them. However broader acceptability of the algorithm will require comprehensive evaluations.

1 Introduction

Medical notes and patient reports provide a wealth of medical information about disease and medication effects. However a substantial amount of clinical data is locked away in non-standardised forms of clinical language which could be usefully mined to gain greater understanding of patient care and the progression of diseases if standardised. Unlike well written texts, such as scientific papers and formal medical reports, which generally conform to conventions of structure and readability, the clinical notes about patients written by a general practitioners, are in a less structured and often minimal grammatical form. As these notes often have little if any formal organisation, it is difficult to extract information systematically. Nowadays there is an increased interest in the automated processing of clinical notes by Natural Language Processing (NLP) methods which can exploit the underlying structure inherent in language itself to derive meaningful information (Friedman et al., 1994).

In principle, clinical notes could be recorded in a coded form such as SNOMED CT (SNOMED International, 2006) or UMLS (Lindberg et al., 1993), however, in practice notes are written and stored in a free text representation. It is believed that the encoding of notes will provide better information for document retrieval and research into clinical practice (Brown and Sönksen, 2000). The use of standard terminologies for clinical data representation is critical. Many clinical information systems enforce standard semantics by mandating structured data entry. Transforming findings, diseases, medication procedures in clinical notes into structured, coded form is essential for clinician research and decision support system. Using concepts in domain specific terminology can enhance retrieval. Therefore, converting free text in clinical notes to terminology is a fundamental problem in many advanced medical information systems.

SNOMED CT is the most comprehensive medical terminology in the world and it has been adopted by the Australia government to encode clinical disease and patient reports. The doctors want a system to develop a standard terminology on SNOMED CT for reporting medical complaints so that their information is exchangeable and semantically consistent for other practitioners, and permit automatic extraction of the contents of clinical notes to compile statistics about diseases and their treatment. Translate medical concepts in free text into standard medical terminology in coded form is a hard problem, and cur-

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 75–82.

rently mostly solved by employing human coders trained both in medicine and in the details of the classification system. To increase the efficiency and reduce human cost, we are interested to develop a system that can automate this process. There are many researchers who have been working on mapping text to UMLS (The Unified Medical Language System), however, there is only a little work done on this topic for the SNOMED CT terminology. The present work proposes a system that automatically recognises medical terms in free text clinical notes and maps them into SNOMED CT terminology. The algorithm is able to identify core medical terms in clinical notes in real-time as well as negation terms and qualifiers. In some circles SNOMED CT is termed an ontology, however this paper only covers its role as a terminology so we will use that descriptor only.

2 Related Work

2.1 Concept Mapping in Medical Reports

There has been a large effort spent on automatic recognition of medical and biomedical concepts and mapping them to medical terminology. The Unified Medical Language System Metathesaurus (UMLS) is the world's largest medical knowledge source and it has been the focus of much research. Some prominent systems to map free text to UMLS include SAPHIRE (Hersh et al., 1995), MetaMap (Aronson, 2001), IndexFinder (Zou et al., 2003), and NIP (Huang et al., 2005). The SAPHIRE system automatically maps text to UMLS terms using a simple lexical approach. IndexFinder added syntactic and semantic filtering to improve performance on top of lexical mapping. These two systems are computationally fast and suitable for real-time processing. Most of the other researchers used advanced Natural Language Processing Techniques combined with lexical techniques. For example, NIP used sentence boundary detection, noun phrase identification and parsing. However, such sophisticated systems are computationally expensive and not suitable for mapping concepts in real time.

MetaMap has the capacity to code free text to a controlled terminology of UMLS. The MetaMap program uses a three step process started by parsing free-text into simple noun phrases using the Specialist minimal commitment parser. Then the phrase variants are generated and mapping candidates are generated by looking at the UMLS source vocabulary. Then a scoring mechanism is used to evaluate the fit of each term from the source vocabulary, to reduce the potential matches. The MetaMap program is used to detect UMLS concepts in e-mails to improve consumer health information retrieval (Brennan and Aronson, 2003).

The work done by (Hazelhurst et al., 2005) is on taking free text and mapping it into the classification system UMLS (Unified Medical Language System). The basic structure of the algorithm is to take each word in the input, generate all synonyms for those words and find the best combination of those words which matches a concept from the classification system. This research is not directly applicable to our work as it does not run in real time, averaging 1 concept matched every 20 seconds or longer.

2.2 Negation and Term Composition

Negation in medical domains is important, however, in most information retrieval systems negation terms are treated as stop words and are removed before any processing. UMLS is able to identify propositions or concepts but it does not incorporate explicit distinctions between positive and negative terms. Only a few works have reported negation identification (Mutalik et al., 2001; Chapman et al., 2001; Elkin et al., 2005).

Negation identification in natural languages is complex and has a long history. However, the language used in medical domains is more restricted and so negation is believed to be much more direct and straightforward. Mutalik et al (2001) demonstrated that negations in medical reports are simple in structure and syntactic methods are able to identify most occurrences. In their work, they used a lexical scanner with regular expressions and a parser that uses a restricted context-free grammar to identify pertinent negatives in discharge summaries. They identify the negation phrase first then identify the term being negated.

In the work of (Chapman et al., 2001), they used a list of negation phrases derived from 1,000 sentences of discharge summaries. The text is first indexed by UMLS concepts and a rule base is then applied on the negation phrases to identify the scope of the negation. They concluded that medical text negation of clinical concepts is more restricted than in non-medical text and medical narrative is a sublanguage limited in its purpose, so therefore may not require full natural language understanding.

3 SNOMED CT Terminology

The Systematized Nomenclature of Medicine Clinical Terminology (SNOMED CT) is developed and maintained by College of American Pathologists. It is a comprehensive clinical reference terminology which contains more than 360,000 concepts and over 1 million relationships. The concepts in SNOMED CT are organised into a hierarchy and classified into 18 top categories, such as *Clinical Finding*, *Procedure*, *Body Part*, *Qualifier* etc. Each concept in SNOMED CT has at least three descriptions including 1 preferred term, 1 fully specified name and 1 or more synonyms. The synonyms provide rich information about the spelling variations of a term, and naming variants used in different countries. The concepts are connected by complex relationship networks that provide generalisation, specialisation and attribute relationships, for example, "*focal pneumonia*" is a specialisation of "*pneumonia*". It has been proposed for coding patient information in many countries.

4 Methods

4.1 Pre-processing

Term Normalisation

The clinical notes were processed at sentence level, because it is believed that the medical terms and negations do not often cross sentence boundaries. A maximum entropy model based sentence boundary detection algorithm (Reynar, and Ratnaparkhi, 1996) was implemented and trained on medical case report sentences. The sentence boundary detector reports an accuracy of 99.1% on test data. Since there is a large variation in vocabulary written in clinical notes compared to the vocabulary in terminology, normalisation of each term is necessary. The normalisation process includes stemming, converting the term to lower case, tokenising the text into tokens and spelling variation generation (haemocyte vs. hemocyte). After normalisation, the sentence then is tagged with POS tag and chunked into chunks using the GENIA tagger (Tsuruoka et al., 2005). We did not remove stop words because some stop words are important for negation identification.

Administration Entity Identification

Entities such as *Date*, *Dosage* and *Duration* are useful in clinical notes, which are called administration entities. A regular expression based named entity recognizer was built to identify administration units in the text, as well as quantities such as *5 kilogram*. SNOMED CT defined a set of standard units used in clinical terminology in the subcategory of *unit* (*258666001*). We extracted all such units and integrated them into the recognizer. The identified quantities are then assigned the SNOMED CT codes according to their units. Table 1 shows the administration entity classes and examples.

Entity Class	Examples
Dosage	40 to 45 mg/kg/day
Blood Pressure	105mm of Hg
Demography	69 year-old man
Duration	3 weeks
Quantity	55x20 mm

Table 1: Administration Entities and Examples.

4.2 SNOMED CT Concept Matcher

Augmented SNOMED CT Lexicon

The Augmented Lexicon is a data structure developed by the researchers to keep track of the words that appear and which concepts contain them in the SNOMED CT terminology. The Augmented Lexicon is built from the Description table of SNOMED CT. In SNOMED CT each concept has at least three descriptions, preferred term, synonym term and fully specified name. The fully specified name has the top level hierarchy element appended which is removed. The description is then broken up into its atomic terms, i.e. the words that make up the description. For example, *myocardial infarction (37436014)* has the atomic word *myocardial* and *infarction*. The UMLS Specialist Lexicon was used to normalise the term. The normalisation process includes removal of stop words, stemming, and spelling variation generation. For each atomic word, a list of the Description IDs that contain that word is stored as a linked list in the Augmented Lexicon. An additional field is stored alongside the augmented lexicon, called the "Atomic term count" to record the number of atomic terms that comprise each description. The table is used in determining the accuracy of a match by informing the number of tokens needed for a match. For example, the atomic term count for *myocardial infarction (37436014)* is *2*, and the accuracy ratio is *1.0*. Figure 1 contains a graphical representation of the Augmented SNOMED CT Lexicon.

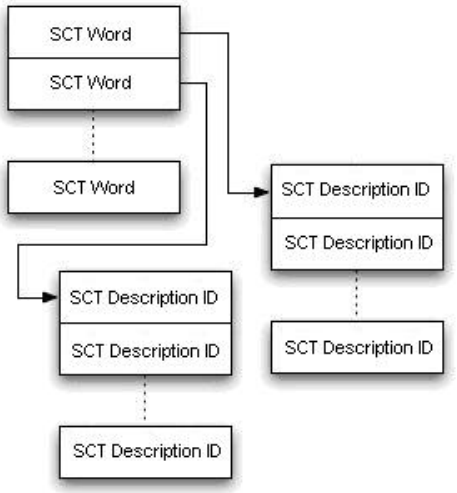

Figure 1: Augmented SCT Lexicon

Token Matching Algorithm

The token matching algorithm takes unstructured text and pre-processes it using the same techniques as are applied to the concepts when generating the augmented lexicon. It then attempts to find each SNOMED CT Description which is contained in the input sentence. For each word, the algorithm looks up the Augmented Lexicon, retrieving a list of the descriptions which contain the word. Figure 2 gives a graphical representation of the data structure used in the algorithm. The elements of the matrix are n-grams from the input sentence with the diagonal line sequence runs of two words. The remainder of the matrix is the cell to the left of it with the next word appended onto it. In this way the algorithm covers every possible sequence of sequential tokens

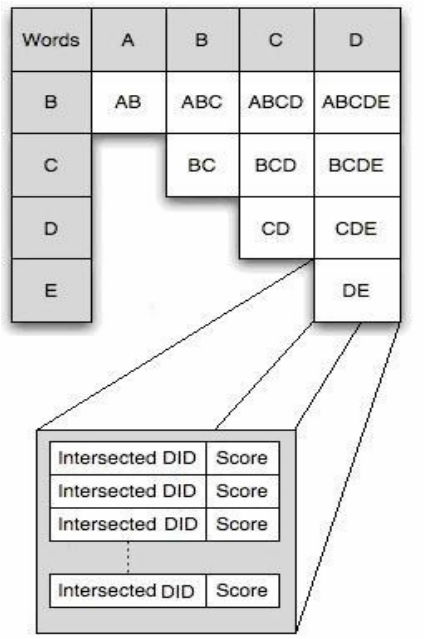

Figure 2: Matching Matrix example

The data stored in each cell is a list of Description IDs (DID) that are in all the tokens that comprise the cell, i.e. the intersection of each set of DID of each word. The score is then calculated using the "atomic term count", which stores the number of tokens that make up that description. The score is the number of tokens in the current cell that have the DID in common divided by the number of tokens in the full description, i.e.:

$$\text{Score} = \frac{\#\text{of Tokens in Sequence}}{\#\text{of Tokens in Full Description}}$$

The algorithm itself is shown here in Figure 3 as pseudo-code. Step 1 is building the Matching Matrix. Step 2 is using the Matching Matrix to find the best combination of sequences that gives the highest score. This final score is dependant on the number of tokens used to make the match divided by the total number of tokens in the input stream, i.e.:

$$\text{Score} = \frac{\#\text{of Tokens used in all matches}}{\#\text{of Tokens in total input stream}}$$

```
STEP 1
for each word in list:
        add entry to the Matching Matrix
        for new column:
                Intersect new word with
                cell from matching table

Sort the matching array in descending order based off
        the scores
for each row in the matrix:
        start at the right most cell

STEP 2
if the top score for the cell is 1.0
        add cell details to current best match list,
        update current match score.
        recursively call STEP 2 on cell
        (row=column+2, column=right)
else:
        move one column left to the next cell
        or
        the right-most cell of the next row if left cell
        empty

repeat STEP 2 until visited all cells
```

Figure 3: Matching Algorithm Pseudo-code

Adding Abbreviations

Different sub-domains have different definitions of abbreviations. In medical domain, the abbreviations are highly ambiguous, as (Liu et al., 2002) show that 33% of abbreviations in UMLS are ambiguous. In different hospitals, they have their own convention of abbreviations, and the abbreviations used are not the same cross the sections in the same sub-domain. This creates difficulties for resolving the abbreviation problem. As we are processing clinical data in the RPAH (Royal Prince Alfred Hospital) ICU (Intensive Care Unit), we believe that the abbreviations used in their reports are restricted to a sub-domain and not that ambiguous. We use a list of abbreviations provided by the ICU department, and integrated them into the Augmented Lexicon. The abbreviations are manually mapped to SNOMED CT concepts by two experts in RPAH. The list consists of 1,254 abbreviations, 57 of them are ambiguous (4.5%). We decided not to disambiguate the abbreviations in the token matching, but return a list of all possible candidates and leave it for later stage to resolve the ambiguity.

4.3 Negation Identification

In our system, we aim to identify the negation phrases and the scope of the negation. Two kinds of negations are identified, the pre-coordianted SNOMED CT concepts and concepts that are explicitly asserted as negative by negation phrases. A pre-coordinated phrase is a term that exists in SNOMED CT terminology that represents a negative term, for example *no headache*.

SNOMED CT contains a set of per-coordinated negative terms under the *Clinical Finding Absent (373572006)* category that indicate the absence of findings and diseases. However, SNOMED CT is not an exhaustive terminology, it is not able to capture all negated terms. Moreover clinical notes have many negation forms other than "absence", such as "denial of procedures". For a negative term that has a pre-coordinated mapping in SNOMED CT, we mark up this term using the SNOMED CT concept id (CID), for other negations, we identify the negation phrases and the SNOMED CT concepts that the negation applies on. The following examples show the two different negations:

no headache (Pre-coordinated negation term)
"absent of"
CID: 162298006
no headache (context-dependent category)

no evidence of neoplasm malignant
(Explicitly asserted negation)
negation phrase: "no evidence of"
CID: 363346000
malignant neoplastic disease (disorder)

Figure 4: Examples of Negations

To identify explicitly asserted negation, we implemented a simple-rule based negation identifier similar to (Chapman et al, 2001; Elkin et al, 2005). At first the SNOMED CT concept id is assigned to each medical term, the negation phrases then are identified using a list of negation phrases in (Chapman et al, 2001). Then a rule base is applied on the negation phrase to check at its left and right contexts to see if any surrounding concepts have been negated. The algorithm is able to identify the negation of the form:

negation phrase … (SNOMED CT phrase)*
(SNOMED CT phrase)* … negation phrase

The contexts can up to 5 non-stopwords long, which allow identification of negation of coordination structure, for example in the following sentence segment:

… and pelvis ***did not*** reveal **retroperitoneal lymphadenopathy** or **mediastinal lymphadenopathy** …

In this sentence segment, the terms, *retroperitoneal lymphadenopathy* and *mediastinal lymphadenopathy* are negated.

Whenever there is a overlapping between Pre-coordinated negation and explicitly asserted negation, we identify the term as pre-coordinated negation. For example, the term *no headache (162298006)* will not be identified as the negation of *headache (25064002)*.

4.4 Qualification and Term Composition

In medical terminology a term may contain an atomic concept or composition of multiple concepts, for example the term pain is an atomic concept and *back pain* represents composition two atomic concepts back and pain. Some composite concepts appear as single concepts in medical terminology, for example *back pain* is a single concept in SNOMED CT. Such concept is called pre-coordinated concept. However, the

medical terms can be composed by adding adjective modifiers to form new terms, for example, the add qualifiers to the concept *pain* can have *back pain*, *chronic back pain*, *chronic low back pain* etc. It is impossible to pre-coordinate combinations of all qualifiers into a terminology, because it will lead to term explosion. Term composition allows user to create new composite concepts using two or more single or composite concept. It is a solution to so called content completeness problem.

The SNOMED CT terminology has a subclass of terms called qualifier values. The qualifier values are used to qualify core concepts. The SNOMED CT defined qualifying relationship adds additional information about a concept without changing its meaning. In most cases, the qualifier is an adjective. There are also some nouns classified as qualifiers, such as *fractions (278277004)*.

The purpose of the qualifier matching is to perform term composition. We separate the qualifiers apart from the Augmented Lexicon when performing concept matching, and build another lexicon that contains only qualifiers. Another reason for treating the qualifier differently is that the qualifier values always conflict with commonly used English words, for example, the unit qualifier *day (258703001)*, side qualifier *left (7771000)*, technique qualifier *test (272394005)*. Such qualifiers cause noise when mapping text to concepts, and they should be refined by looking at their context.

The Concept Matchers runs at first to identify any SNOMED CT concepts and qualifiers. A search then is run to look at the qualifiers' surroundings using the following rules to identify the scope of qualification. A concept can have multiple qualifiers to modify it.

(Qualifier / JJ|NN)* … (Concept / NN)*
(Concept / NN)* … (Qualifier / JJ|NN)*

The first rule aims identify left hand side qualifications, for example in the following sentence segment:

… She had severe **lethargy** and ***intermittent right upper*** **abdominal discomfort** …

The second rule aims to identify right hand side qualification, for example:

... **autoimmune screening** were ***normal*** …

If no concepts are found with in a context window, the qualifier then is not considered as a modifier to any medical concepts, thus removed to reduce noise.

5 Results and Discussion

The token matching algorithm has been implemented as a web-based service named TTSCT (Text to SNOMED CT) that provides web interfaces for users to submit clinical notes and respond with SNOMED CT codes in real-time. The system is able to encode SNOMED CT concepts, qualifiers, negations, abbreviations as well as administration entities. It has been developed as the first step to the analysis and deep understanding of clinical notes and patient data. The system has been installed in RPAH (Royal Prince Alfred Hospital) ICU (Intensive Care Unit) aiming to collect bedside patient data. The web interface has been implemented in several clinical form templates the RPAH, allowing data to be captured as the doctors fill in these forms. A feedback form has been implemented allowing clinicians to submit comments, identify terms that are missed by the system and submit corrections to incorrectly labelled terms. Figure 5 shows the concepts that have been identified by the TTSCT system and Figure 6 shows the responding SNOMED CT codes.

No neoplasm malignant negation seen.

Sections confirm **CRANIOPHARYNGIOMA** concept with **small** qualifier **fragments** qualifier of **adjacent** qualifier **brain tissue** concept.

The **slides** concept show degenerate **atypical** qualifier urothelial **cells** concept occurring in **sheets** qualifier and singly with **hyperchromatic** qualifier **enlarged** qualifier **irregular** qualifier **nuclei** concept.

Figure 5: A Sample Clinical Note

SNOMED CT Concept	SCT Concept ID	SCT Fully Specified Name
CRANIOPHARYNGIOMA	40009002	Craniopharyngioma (morphologic abnormality)
	189179009	Craniopharyngioma (disorder)
Brain tissue	256865009	Brain tissue (substance)
Cells	4421005	Cell structure (cell structure)
	362837007	Entire cell (cell)
hyperchromatic	9767008	Hyperchromatism (morphologic abnormality)

Qualifiers	SCT Concept ID	SCT Fully Specified Name	Scope of Qualification
Small	255507004	Small (qualifier value)	
	263796003	Lesser (qualifier value)	
Fragments	29140007	Fragment of (qualifier value)	
Adjacent	18769003	Juxta-posed (qualifier value)	brain tissue
Atypical	112231000	Atypical (qualifier value)	Cells
Sheets	255292000	Sheets (qualifier value)	
Enlarged	260376009	Enlarged (qualifier value)	
Irregular	49608001	Irregular (qualifier value)	
Fragments	29140007	Fragment of (qualifier value)	Tissue

Negation	Negation Phrase	Negative Term
no neoplasm malignant	No	neoplasm malignant (86049000)

Figure 6: Concepts, Qualifiers and Negations Identified From the Sample Note

We are currently collecting test data and evaluating the accuracy of our method. We plan to collect patient reports and cooperate with the clinicians in the RPAH to identify correct mappings, missing mappings and incorrect mappings. Although the algorithm hasn't been comprehensively evaluated on real data, we have collected some sample patient reports and a few feedback from some clinicians. Preliminary results demonstrate that the algorithm is able to capture most of the terms within acceptable accuracy and response time.

By observation, missing terms and partially identified terms are mainly due to the incompleteness in SNOMED CT. In the above example, the *atypical urothelial cells* is only partially matched, because neither *atypical urothelial cell* is present in SNOMED CT as a single term nor *urothelial* can be found as a qualifier in SNOMED CT. However the qualified term *moderate urothelial cell atypia* can be found in SNOMED CT. This raises the question of term composition and decomposition because the terms in the terminology have different levels of composition and the qualification can be written in a different order with morphological transformation (*urothelia cell atypia* vs. *atypical urothelial cell*). The qualifier ontology and term relationships must be addressed to make sure term composition is done in a reliable manner.

Restricting the concept mapping to noun phrase chunkers can rule out many false positives and also increase the speed of processing, however many pre-coordinated terms and qualifications cross noun phrase boundaries, for example the term "*Third degree burn of elbow (87559001)*" will be broken into two terms "*Third degree burn (403192003)*" and "*elbow (76248009)*" and their relationship not preserved.

6 Conclusions

In conclusion, we propose an algorithm to code free text clinical notes to medical terminology and implemented it as a web-service system. The algorithm utilised NLP techniques to enhance lexical concept mappings. A qualifier identifier and negation identifier have been implemented for recognising composite terms and negative concepts, which can then create more effective information retrieval and information extraction. The system is yet to be fully evaluated, nevertheless the test on sample data shows it is already meeting expectations. In the future, we will perform comprehensive evaluation for the algorithm on real clinical data, and compare the system with some well known term indexing algorithms.

References

Aronson, A. R. (2001). *Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program*. Proc AMIA Symp 17: 21.

Brennan, P. F. and A. R. Aronson (2003). *Towards linking patients and clinical information: detecting UMLS concepts in e-mail*. Journal of Biomedical Informatics 36(4/5): 334-341.

Brown, P. J. B. and P. Sönksen (2000). *Evaluation of the Quality of Information Retrieval of Clinical Findings from a Computerized Patient Database Using a Semantic Terminological Model.* Journal of the American Medical Informatics Association 7: 392-403.

Chapman, W. W., W. Bridewell, et al. (2001). *A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries.* Journal of Biomedical Informatics 34(5): 301-310.

Chute, C. G. and P. L. Elkin (1997). *A clinically derived terminology: qualification to reduction*. Proc AMIA Annu Fall Symp 570: 4.

Elkin, P. L., S. H. Brown, et al. (2005). *A controlled trial of automated classification of negation from clinical notes*. BMC Medical Informatics and Decision Making 2005(5): 13.

Friedman, C., P. O. Alderson, et al. (1994). *A general natural-language text processor for clinical radiology.* Journal of the American Medical Informatics Association 1(2): 161-174.

Friedman, C., L. Shagina, et al. (2004). *Automated Encoding of Clinical Documents Based on Natural Language Processing*. J Am Med Inform Assoc 11: 392-402.

Hazlehurst, B., H. R. Frost, et al. (2005). *MediClass: A System for Detecting and Classifying Encounter-based Clinical Events in Any Electronic Medical Record*, American Medical Informatics Association.

Hersh, W. R. and D. Hickam (1995). *Information retrieval in medicine: The SAPHIRE experience.* Journal of the American Society for Information Science 46(10): 743-747.

Huang, Y., H. J. Lowe, et al. (2005). *Improved Identification of Noun Phrases in Clinical Radiology Reports Using a High-Performance Statistical Natural Language Parser Augmented with the UMLS Specialist Lexicon*, American Medical Informatics Association.

Lindberg, D. A., B. L. Humphreys, et al. (1993). *The Unified Medical Language System.* Methods Inf Med 32(4): 281-91.

Liu H., Johnson S. B. and Friedman C. (2002), *Automatic Resolution of Ambiguous Terms Based on Machine Learning and Conceptual Relations in the UMLS.* Journal of the American Medical Informatics Association, Vol. 9, No. 6, Pages 621-636.

Mutalik, P. G., A. Deshpande, et al. (2001). *Use of General-purpose Negation Detection to Augment Concept Indexing of Medical DocumentsA Quantitative Study Using the UMLS*, Journal of the American Medical Informatics Association 2001.

Nadkarni, P., R. Chen, et al. (2001). *UMLS Concept Indexing for Production Databases*. Journal of the American Medical Informatics Association 8: 80-91.

Reynar, J. C. and A. Ratnaparkhi (1997). *A maximum entropy approach to identifying sentence boundaries.* Proceedings of the Fifth Conference on Applied Natural Language Processing: 16–19.

SNOMED International. (2006). *SNOMED International: The Systematized Nomenclature of Medicine*. http://www.snomed.org. Last accessed 08-09-2006.

Stearns, M. Q., C. Price, et al. (2001). *SNOMED clinical terms: overview of the development process and project status.* Proc AMIA Symp 662(6). Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503-512.

Tsuruoka, Y., Y. Tateishi, et al. *Developing a robust part-of-speech tagger for biomedical text*. Lecture notes in computer science: 382-392.

Zou, Q., W. W. Chu, et al. (2003). *IndexFinder: A Method of Extracting Key Concepts from Clinical Texts for Indexing.* Proc AMIA Symp 763: 7.

Pseudo Relevance Feedback Using Named Entities for Question Answering

Luiz Augusto Pizzato and **Diego Mollá**
CLT - Macquarie University
Sydney, Australia
{pizzato, diego}@ics.mq.edu.au

Cécile Paris
ICT - CSIRO
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

Relevance feedback has already proven its usefulness in probabilistic information retrieval (IR). In this research we explore whether a pseudo relevance feedback technique on IR can improve the Question Answering task (QA). The basis of our exploration is the use of relevant named entities from the top retrieved documents as clues of relevance. We discuss two interesting findings from these experiments: the reasons the results were not improved, and the fact that today's metrics of IR evaluation on QA do not reflect the results obtained by a QA system.

1 Introduction

Probabilistic Information Retrieval estimates the documents' probability of relevance using a small set of keywords provided by a user. The estimation of these probabilities is often assisted by the information contained in documents that are known to be relevant for every specific query. The technique of informing the IR system which documents or information are relevant to a specific query is known as relevance feedback. As reported by Ruthven and Lalmas (2003), relevance feedback techniques have been used for many years, and they have been shown to improve most probabilistic models of Information Retrieval (IR).

Relevance feedback is considered as pseudo (or blind) relevance feedback when there is an assumption that the top documents retrieved have a higher precision and that their terms represent the subject expected to be retrieved. In other words, it is assumed that the documents on the top of the retrieval list are relevant to the query, and information from these documents is extracted to generate a new retrieval set.

In this paper we explore the use of a pseudo relevance feedback technique for the IR stage of a Question Answering (QA) system. It is our understanding that most questions can be answered using an arbitrary number of documents when querying an IR system using the words from the topic and the question. Because QA is normally a computationally demanding task, mostly due to the online natural language processing tools used, we believe that IR can help QA by providing a small list of high-quality documents, i.e. documents from where the QA system would be able to find the answer. In this sense, documents containing answers for a question in a sentence structure that can be easily processed by a QA system will be highly relevant.

In this work, we describe an experiment using a pseudo relevance feedback technique applied over a probabilistic IR system to try to improve the performance of QA systems. Since most types of factoid questions are answered by named entities, we assume that documents addressing the correct topic but not containing any named entity of the expected answer class would have a low probability of relevance regarding QA. Therefore, we hypothesise that documents containing named entities of the correct class have higher probability of relevance than those not containing them.

The relevance feedback applied to QA differs from the one applied to general IR in the sense that QA deals more with the presence of a passage that can answer a certain question than with the presence of its topic. In this sense, our technique focuses on feeding terms into the IR engine that could represent an answer for the questions.

Despite the fact that it is possible to apply the

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 83–90.

technique to most question types, in this work we only report the results of questions regarding people's names. We understand that the other types of questions are as important as this and may generate different results due to the different frequency of their appearance in the documents; however people's names can provide us with concrete results since it is a type of named entity that has been widely experimented on recognisers and is likely to be present in most types of newswire texts, as those in Aquaint corpus (Graff, 2002). We performed our experiments using the Aquaint corpus and the set of question from the QA track of TREC'2004 (Voorhees, 2005).

The next section provides some background information on IR techniques applied to QA. Section 3 explains the principles behind the named-entity relevancy feedback technique and how we implemented it. Section 4 focuses on the evaluation of the technique regarding its use as an IR tool and as a module of a QA system. Section 5 presents the concluding remarks and future work.

2 Document Retrieval for Question Answering

Question Answering is the field of research that focuses on finding answers for natural language questions. Today, QA focuses on finding answers using textual documents, as in the TREC QA Tracks (Voorhees and Tice, 2000). Although finding answers by first populating a database with likely answers to common questions and then consulting the database at question time is still an interesting task from an information extraction point of view, most research in this area is focusing on online open domain question answering using large collections of documents.

Research on offline/database and online/textual QA styles has shown that using offline/database information is possible to achieve a higher precision with the cost of a lower recall comparing with online/textual information (Mur, 2004). Even though methods using textual corpora have not yet obtained a precision high enough for practical applications, a large amount of question types can hypothetically be answered.

Most QA systems follow a framework that involves processing the question, finding relevant documents and extracting the required answer. The majority of QA systems tend to apply their complex methodologies on both ends of this framework (on question analysis and on answer extraction), but in order to extract the correct answer for a question, a QA system needs to find a document that contains the answer and some supporting evidence for it. Therefore, one of the fundamental stages of a QA system is the document retrieval phase. It does not matter how advanced the techniques used by the QA are if the retrieved set of documents does not include the answer it requires.

In Section 4 we show that, using just the question topic as the IR query, it is possible to obtain reasonable results on a QA system. However, depending on the complexity of the techniques used by the QA system, it is necessary to reduce the retrieval set to a minimal and optimal number of documents in order to allow the completion of the task in a reasonable time.

Some work has been done on specific IR models for aiding the QA task. The work of Monz (2004) defines a weighting scheme that takes into consideration the distance of the query terms. Murdock and Croft (2004) propose a translation language model that defines the likelihood of the question being the translation of a certain document. Tiedemann (2005) uses a multi-layer index containing more linguistic oriented information and a genetic learning algorithm to determine the best parameters for querying those indexes when applied for the QA task. In other words, Tiedemann argues that since question answering is an all-natural language task, linguistic oriented IR will help finding better documents for QA. However, just the use of extra information may not necessarily improve QA when it is important to know the right configuration of your modules for the task.

Another way of limiting the amount of information sent to the QA system is by selecting the best passages or sentences that a QA system will analyse. Some IR work focuses on improving QA by passage retrieval re-ranking using word overlap measures. For instance, Tellex et al. (2003) compared a group of passage retrieval techniques and concluded that those that apply density-based metrics are the most suitable to be used on QA.

Since most IR is treated as a blackbox by the QA community, manipulation of the IR results is normally performed by query modification. The most common query modifications are lexical substitution and query expansion. These techniques seem to be obligatory for most QA systems when

the original retrieval set has a small recall. However, it tends to reduce the precision in a way that harms QA by introducing documents of unrelated subjects. On the other hand, White and Sutcliffe (2004) have shown that since only a small amount of terms from questions match the supporting answer sentence, it is important for QA systems that rely on word overlap to apply some semantic or morphological expansion.

Another way of modifying the IR phrase is by performing passive to active voice transformation of the question, as in Dumais et al. (2002). This has been shown to work well since some IR systems give preference to the distance and order of terms in the query by making the affirmative voice of the answers preferable over the passive one of the questions.

Most IR research applied to QA use similar metrics to Roberts and Gaizauskas (2004) to evaluate their systems. These metrics, defined by the authors as coverage and redundancy, evaluate respectively the percentage of a set of questions that could be answered using the top-N documents of a retrieval set, and how many answers on average it finds.

It is understandable that this metrics are closely related to the needs of QA systems, and we will show that even though they provide us with the information of how likely we are of finding the answer in the retrieval set, they do not guarantee a better QA performance. This and other issues are addressed in the following sections.

3 Relevance Feedback Using Named Entities

Because named entities are required as answers for most fact-based questions, we are hypothesising that a relevance feedback mechanism that focuses on this kind of information will be useful. Therefore, we are focusing on the QA concept of relevance by trying to reduce the number of documents that would not be able to answer a factoid question. By doing this, not only the process will guide the document retrieval towards documents relevant to the question topic (general IR relevancy) but also towards those containing entities that could answer the question (QA relevancy).

Let us say that we have a question *Q* of a topic T[1] and a probabilistic IR engine using the combination *Q+T* to obtain *R1* as a set of documents. Our process applies a named entity recogniser over the top-N ranked documents of *R1*, thus obtaining a set of named entities *E*. The feedback process consists of enriching the previous query as *Q+T+E* in order to obtain a new set of documents *R2*.

Our expectation on this technique is that not only documents containing the correct answer in *R1* will be boosted in ranking on *R2*, but also that documents that have a high ranking in *R1* and do not contain any name entity of the expected answer type will then be demoted in *R2*. Therefore, documents that theoretically would not contribute to the QA performance will not take part on the answer extraction phase, allowing their slots of processing to be occupied by other more relevant documents.

In order to exemplify this process, consider the TREC 2005 QA Track question 95.3 regarding *the return of Hong Kong to Chinese sovereignty*: "*Who was the Chinese President at the time of the return?*"

The first phase of the process is the question analysis that defines what the expected answer type is and what the question main words are. Then the question and its topic define an IR query that generates the retrieval set R1.

The next process extracts the named entities of the expected answer type of the first *N* documents in the R1 set of documents. For the example, fifteen names of people were extracted, mostly Chinese and all of them related with politics. A new IR query is built using these fifteen names and the final set *R2* of documents is retrieved.

The list of names found for this query is listed on Table 1. We can observe that, among those names there is the correct answer for the question (*President Jiang Zemin*), which helped generating a better retrieval for this question with the pseudo relevance feedback mechanism.

Table 1: Extracted Named Entities

President Mario Alberto N. L. Soares	President Jiang Zemin
General Secretary Aleksandr Zharikov	Minister Qian Qichen
Minister Sabah Al- Ahmad Al-Jaber	Minister Zhou Nan
Prime Minister Mahmoud Zouebi	Mr. Deng Xiaoping
President Maumoon Abdul Gayoom	Premier Li Peng
President Ugo Mifsud Bonnici	Liu Huaqiu
President Meets Chinese	laws Will
President Leonid Kuchma	

[1]The distinction between topic (or target) and question is made clear on recent TREC QA Tracks (Voorhees, 2005). However, most cases of question answering systems would have the topic extracted from the question itself.

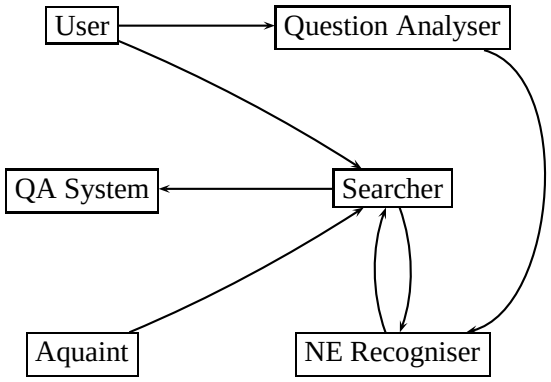

Figure 1: System overview for the the relevance feedback process.

Our hypothesis is that the named-entity feedback technique improves the overall document retrieval for QA by providing a retrieval set of documents that facilitates the extraction of the correct answer by a QA system. The technique should theoretically improve good questions (where a correct feedback is obtained) and not deteriorate bad ones[2]. Next section describes the experiments we performed and the results.

3.1 Implementation

The technique consists of posting the original question to a probabilistic IR engine, extracting the named entities of the expected answer type from the top-N results, and re-feeding the IR engine with an expanded query. By doing this, we are telling the IR system that documents containing those named entities are relevant to the question. Several implementations and set-ups can be tested using this approach, but the basic framework we implemented is shown on Figure 1.

We developed our IR system using C++ and the XAPIAN[3] Toolkit for Probabilistic IR. The Aquaint Corpus (Graff, 2002) was indexed using full text but stopwords, and it was searched using Xapian Probabilistic Methods (it uses Robertson's BM25 (Robertson et al., 1992) for ranking).

As can be seen in Figure 1, the user poses a question to the system. It is simultaneously processed by the question analyser and the searcher. The question analyser returns the expected answer type (a named-entity class for factoid questions), while the searcher returns a list of documents or snippets of text from the Aquaint corpus ranked by Xapian BM25 implementation. The named-entity recogniser receives the output of these two processes and extracts the corresponding named entities from the received files. Once this is done, it re-feeds the query to the searcher with the additional named entities. The searcher then feeds the results into the QA system.

4 Experiments and Evaluation

We use in our experiments the data collection made available by NIST on the TREC QA Tracks[4]. All the questions and judgement files of TREC 2003 QA Track were used on a preliminary evaluation of this process. Because this experiment required that all components shown on Figure 1 be fully functional, several setups were implemented, including a manual question classification (to ensure 100% correctness) and the implementation of a simple passage retrieval algorithm.

In our evaluation, we labelled documents as relevant or not relevant by assuming that relevant documents are those containing the required answer string. These early tests showed us that using the set of 500 TREC 2003 questions with our pseudo-relevance feedback technique improved the results over the initial retrieval. The improvement, however, was small and not statistically relevant.

On our system architecture, the question classification was performed using the Trie-based technique (Zaanen et al., 2005) which has a performance of around 85% accuracy when trained with the set of questions made available by Li and Roth (2002). This means that in 15% of the cases, we might have an immediate degradation of the results (by adding the wrong named-entities to the query). Because of this, we trained the classification with the same questions as the verification set. This was done to ensure complete correctness on this part of the module. However, because of the large amount of expected answer types present in the classification we used, named entity recognition proved to be a particularly complex task.

Since many questions required numbers as their answers and most documents contain some kind of number, defining a document relevant and using numbers as indication of relevancy does not work well. This demonstrated that even though we obtained better overall results using all categories

[2] A question is bad when used as a query on an IR system, it is unable to retrieve any document containing an answer.

[3] `http://www.xapian.org/`

[4] `http://trec.nist.gov/data/qa.html`

available, some of them were a real challenge for the evaluation.

We also observed that some named-entity classes could not be properly identified by our named-entity recogniser. Therefore we shifted our attention to only people's names, as we understood them to be less likely to suffer from the issues above reported. We also started to use two well known named entity recognisers: Lingpipe[5] and ANNIE[6] on Gate.

The evaluation was performed intrinsically and extrinsically in the same sense as Spärck Jones and Galliers (1996). Intrinsic and extrinsic evaluations differ because the former evaluates a system according to its primary function, while the latter evaluates a system according to its function or its setup purpose. In our study, the evaluation was performed using the combined set of questions and topics of the TREC 2004 and 2005 along with their respective judgement sets. Different setups were experimented, but mainly variations of passage window, the number of top documents used and the weights assigned to the different components (*T*, *Q* and *E*) of the query. We extrinsically evaluated the effectiveness of the retrieval sets by the percentage of correct answers the AnswerFinder(Molla and van Zaanen, 2006) system generated, and intrinsically evaluated the same sets of documents using the standard precision metric for IR and other metrics defined by Roberts and Gaizauskas (2004) for IR on QA:

- **Precision**: percentage of related documents over all questions;
- **Coverage**: percentage of questions that potentially could be answered using the top-*N* documents; this means that at least one of the top-*N* documents potentially answers the question; and
- **Redundancy**: average of how many answers can be found using the top-*N* documents;

We applied the retrieved document set on AnswerFinder and measured the exact results using the patterns made available by Litkowski on the TREC QA Data Webpage.

[5] `http://www.alias-i.com/lingpipe/`
[6] `http://gate.ac.uk/ie/annie.html`

4.1 Results

Our evaluation focused on using pseudo relevance feedback to enrich the IR query used by QA systems to find some documents that could answer natural language questions. We performed an intrinsic evaluation using some standard metrics for IR on QA, and, at the same time, we also performed an extrinsic evaluation by using the retrieval set on the QA system.

Sets of documents were retrieved using a combination of Topics (*T*), Questions (*Q*), Entities (*E*) and Answers (*A*). The following combinations were tested:

- *T*: Only the topic is sent as a query. This set of queries evaluates the potentiality of improving the retrieval set that NIST provides for every topic.
- *TQ*: The queries are made of Topic and Question. This is the current retrieval set used by the AnswerFinder system.
- *TQE*: This is the feedback technique, where Topic, Question and the Named Entities extracted from top-N documents are combined;
- *TQA*: This is the optimal feedback technique, where Topic, Question and Answers are combined. This set evaluated how far from the optimal retrieval we are;
- *TQEA*: These queries combine the feedback technique with the answers, so we can measure the amount of noise introduced by adding bad named entities. We made sure that a named entity that was also the answer was not introduced twice so its score would not be erroneously duplicated on the query.

Different combinations could also be tested, for instance *TA*, *TE* or just *A*, *E* and *Q*. We understand that those and other combinations could provide some insight on certain matters, but we believe that they would not represent a realistic retrieval set. It is a fact that the terms from *T* must be present in the retrieval set, since all documents must address the correct topic. For instance, including *Q* without having *T* will not generate a relevant retrieval because the subject of the question is not present. Also, including *A* or *E* without *Q* and *T* may represent a totally different retrieval that is not desired in this study.

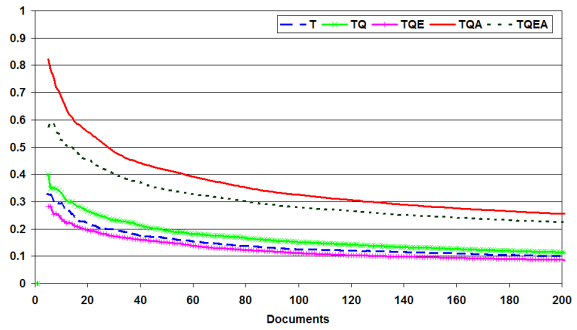

Figure 2: Precision

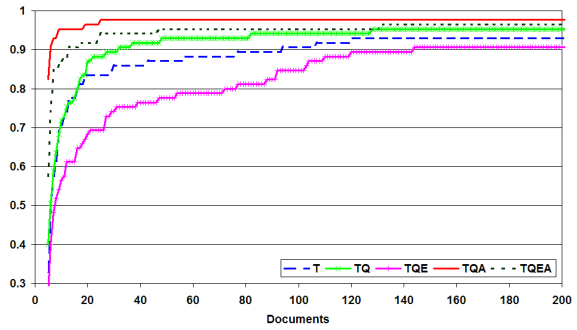

Figure 3: Coverage

The precision, coverage and redundancy obtained for the TREC 2004 and 2005 questions regarding people's name are respectively shown in Figures 2, 3 and 4. We note that the results for the feedback technique do not improve the results on neither *T* nor *TQ* on any of the measures we obtained. As expected, the addition of the answer on *TQA* represents the optimal retrieval set, obtaining the coverage of 86% on the first document per question and over 90% on the second.

The noise introduced on TQEA is not a major concern when the answers are involved in the

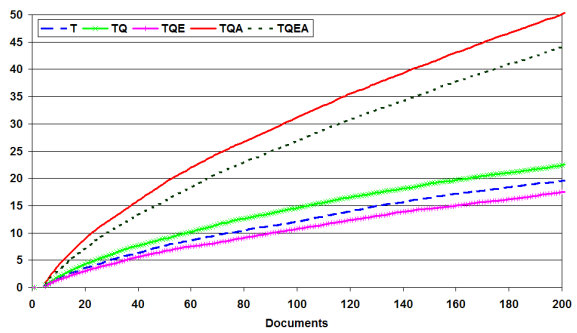

Figure 4: Redundancy

query. This is an indication that most entities found by the feedback mechanism do not represent an answer. This raises two issues: how to improve the technique so that the answers are included in the feedback; and how to minimise the noise so that a potential good feedback is not worsened.

To address the first problem we can foresee two solutions: one is improving the accuracy of the named-entity recogniser, something we cannot address in this study. The other is increasing the search space without adding more noise in the query. This is a difficult task and it could be achieved by finding the smallest possible windows of text containing the answer on several documents. We performed some experiments using different numbers of documents and variable passage size, at the moment fewer documents and smaller passages provide our best results.

We understand that documents in the first retrieval set *R1* will contain named-entities of the same type, but not necessarily the correct one (the answer), thus creating some noise in the query. We believed that a certain degree of noise would not hurt the retrieval performance. However, our experiments, as shown, demonstrate otherwise. The noise created by erroneous entities affects the performance once the elements in *E* become more important than the elements in *Q*. Because we cannot guarantee the correctness of any of the named-entities included in *E*, the resulting retrieval set *R2* might represent a worse retrieval set than *R1*. However, these cases may not influence the results in a QA system since *R1* would also not lead to the correct result.

This shows that our feedback technique suffers from the same flaws most pseudo-feedback techniques have. For instance Ruthven and Lalmas (2003) show that when the initial retrieval set is not good, the pseudo-feedback techniques is likely to worsen the results because, instead of bringing the query closer to the topic at hand, it will take it further away (a phenomenon called query drift). We hypothesise that since our technique is meant to be applicable over a QA system, if the initial set of results is bad (i.e. it does not contain the answer), there is not much that can be worsened. To confirm this hypothesis, it is necessary to perform an evaluation over a QA system. Table 2 shows the runs of QA performed using the same set of questions of the intrinsic evaluation and the documents retrieved by the retrieval sets

Table 2: Correct Answers on AnswerFinder

Run	Exact
T	19.6%
TQ	28.6%
TQE	23.2%
TQA	28.6%
TQEA	32.1%

shown before.

What can be observed here is that the feedback technique (*TQE*) offers a better set of documents than the one using only the topics (*T*). However, they are still worse than the topic and question ones (*TQ*). An interesting result is that *TQEA* is the best run, which may show that the inclusion of entities can help improve QA. We have not yet performed a deep analysis of this case to verify its cause. Even though our process did not show improvements over the baseline techniques, it was very important to find that the results of the (intrinsic) evaluations of the IR component do not parallel the results of the (extrinsic) evaluation of the QA system. In spite of the fact that high precision, coverage and redundancy represent a better chance of finding answers, we show that they do not guarantee a better performance over a QA system.

Comparing the results of *T* and *TQ* it is possible to observe that they are very similar on the intrinsic evaluation and quite different on the QA system. Therefore, what appears to help question answering is the presence of more context words so that the answers not only appear in the document but are also present in the context of the questions. This is mostly due to the fact that most QA systems tend to work with full discourse units, such as sentences and paragraphs, and the selection of those are normally based on words from the topic and the question.

In summary our experiments did not confirm the hypothesis that named-entities feedback would help improving QA. But, in the ideal situations where the answers are identified and included in the queries, the improvements are clear under an intrinsic evaluation. The differences between the intrinsic evaluation and extrinsic one point out that there are many issues that IR metrics are not currently covering.

5 Concluding Remarks and Future Work

In this paper, we have looked at whether a pseudo relevance feedback mechanism could help the QA process, on the assumption that a good indication of a document relevancy for its usage on a QA system is the presence of named entities of the same class required as the answer for a certain question. Our assumption was based on the fact that documents not containing those entities are less likely to help provide the correct answer and every entity of the right type has a probability of being the answer.

We have described our evaluation of the hypothesis using known IR metrics and a QA system. Our main conclusions are:

- Because we have not yet reported satisfactory results, we believe that even though the method is conceptually sound, it will not produce good results unless a more sophisticated control over the introduced noise is achieved; and

- The evaluation of the technique brought to our attention the fact that it is not possible to state that a retrieval technique is better just by relying on conventional IR evaluation metrics. The differences on the intrinsic and extrinsic evaluations demonstrate that there are many hidden variables that are not taken into account in metrics such as precision, coverage and redundancy.

As further work, we plan to repeat our evaluation using different QA systems, since other QA systems may give preference to different text features offering a better insight on how the IR evaluation metrics correlate with the QA results. We are also planning to use the named-entity recogniser that is being developed by our research group and to extend the system to use a more advanced passage retrieval algorithm.

References

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: is more always better? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, Tampere, Finland. ACM Press.

David Graff. 2002. The AQUAINT corpus of english news text. CDROM. ISBN: 1-58563-240-6.

Karen Spärck Jones and Julia R. Galliers. 1996. *Evaluating Natural Language Processing Systems: An*

Analysis and Review. Springer-Verlag New York, Inc.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, pages 556–562, Taipei, Taiwan. Association for Computational Linguistics.

Diego Molla and Menno van Zaanen. 2006. Answerfinder at TREC 2005. In *The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, Maryland. National Institute of Standards and Technology.

Christof Monz. 2004. Minimal span weighting retrieval for question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

Jori Mur. 2004. Off-line answer extraction for dutch qa. In *Computational Linguistics in the Netherlands 2004: Selected papers from the fifteenth CLIN meeting*, pages 161–171, Leiden University.

Vanessa Murdock and W. Bruce Croft. 2004. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

Ian Roberts and Robert J. Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In Sharon McDonald and John Tait, editors, *ECIR*, volume 2997 of *Lecture Notes in Computer Science*, pages 72–84. Springer.

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. 1992. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30.

I. Ruthven and M. Lalmas. 2003. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47, Toronto, Canada. ACM Press.

Jorg Tiedemann. 2005. Optimizing information retrieval in question answering using syntactic annotation. In *Proceedings of RANLP 2005*, pages 540–546, Borovets, Bulgaria.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207, Athens, Greece. ACM Press.

Ellen M. Voorhees. 2005. Overview of the TREC 2004 question answering track. In *Text REtrieval Conference*.

Kieran White and Richard F. E. Sutcliffe. 2004. Seeking an upper bound to sentence level retrieval in question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

Menno Van Zaanen, Luiz Augusto Pizzato, and Diego Molla. 2005. Classifying sentences using induced structure. In *Proceedings of the Twelfth Symposium on String Processing and Information Retrieval (SPIRE-2005).*, Buenos Aires, Argentina, November.

Evidence for Gradient Salience: What Happens with Competing Non-salient Referents during Pronoun Resolution?

Ralph L. Rose
Faculty of International Communication
Gunma Prefectural Women's University
Gunma, Japan
rose@gpwu.ac.jp

Abstract

The necessity of a gradient approach to salience ranking of referents introduced in a discourse is evaluated by looking at (unbound) pronoun resolution preferences when there are competing non-salient referents. The study uses a sentence-completion technique in which participants had to resolve pronouns ("John sprayed the paint on the wall and then it ..."). Results suggest that a gradient salience model is necessary. Syntactic and semantic prominence effects on pronoun resolution were also compared with results showing that semantic prominence (i.e., agent $>$ patient) determined the salience ranking of competing referents.

1 Introduction

A pervasive theme in theories of discourse coherence is the concept of salience. It has proven to be a useful means of explaining how particular entities seem to receive some preferential treatment in both the production and perception of a discourse. For instance, it has long been observed that entities realized in certain structural positions (e.g., grammatical subject or first-mention) are preferred entities for topic continuation (Givón, 1983)—that is, they are preferentially referred to in the subsequent utterance. Similarly, pronominal reference to entities realized in certain structural positions (again, subject position, for example) is preferred to reference by repeated name (Gordon et al., 1993). In order to account for these observations, it has often been theorized that in the speaker's and hearer's mental representation of the discourse, these entities are salient (similar terms include focused or given).

To illustrate this line of thinking, consider (1). The pronoun in the second clause is preferentially interpreted as referring to LUKE rather than MAX. This has been observed in numerous psycholinguistic investigations (cf., Hudson-D'Zmura and Tanenhaus (1997); Mathews and Chodorow (1988); inter alia). In a simple salience-based account, it is hypothesized that LUKE is a salient entity after the first clause and that reference to salient entities should be pronominal.

(1) Luke$_i$ hit Max$_j$ and then he$_{i/\#j}$ ran home.

While many studies have investigated differences between pronominal reference to salient and non-salient entities, I have found no studies that have looked explicitly at what happens when a salient entity is not compatible with the pronoun, but more than one non-salient entity is. This is one of the main themes of the present study. Putting it as a question, what happens when there is competition among non-salient entities for pronoun interpretation? The answer to this question has some wider implications for how salience is to be understood. In particular, the answer to this question leads to conclusions about whether theoretical models require a gradient model of salience ranking or whether a categorical model is sufficient. In the following background section I will discuss this primary question further and introduce two related questions which must also be addressed. This will be followed by description of the experiment performed in this study. Briefly, results of the experiment are consistent with a gradient model of salience ranking. Implications of these findings are discussed in the final section.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 91–98.

2 Background

2.1 Pronoun Reference Resolution

In this paper I will be focusing on the interpretation of unbound pronouns. Much has been written on this area of anaphora resolution and only a cursory overview is possible in this paper (see Hirst (1981) for a comprehensive overview of earlier work and Mitkov (2002) for an overview of more recent work). In this section, I will describe a generalized model of pronoun resolution and how salience plays a role in this process as well as discuss in some detail how salience is determined.

When interpreting pronouns in discourse, readers search a list of previously evoked entities in memory. Following Karttunen (1976) and Heim (1982), I will call these *discourse referents* (or just *referents*, for short). The list of discourse referents is ranked according to salience.

Two basic approaches may be taken to salience ranking: a categorical approach in which at most one referent is salient and all others are then, by definition, not salient; or a gradient approach in which referents are ranked along a salience continuum. In computational implementations of pronoun resolution algorithms, a gradient approach is often used, perhaps by necessity (cf., Lappin and Leass (1994)). However, psycholinguistic studies are often not so explicit about the approach taken and the results of most studies can be explained in terms of a categorical salience ranking. For instance, Gernsbacher and Hargreaves (1988) present a model of comprehension in which order-of-mention determines salience ranking, but their experimental evidence only compares first and second mentioned entities. In another case, Hudson-D'Zmura and Tanenhaus (1997) seek to verify the basic predictions of Centering Theory (Grosz and Sidner, 1986; Grosz et al., 1995), one aspect of which is a syntactic hierarchy: $subjects > objects > others$. However, their experimental evidence really only demonstrates a categorical ranking: $subjects > others$.

The difference between the categorical and gradient approaches is important in the present study because if salience is categorical, then pronominal reference should show no preference among non-salient entities. On the other hand, if salience is gradient, then it should be possible to observe preferences even among non-salient (or perhaps more accurately here, less salient) entities.

I should note however, that while it is clear that those who take a gradient point of view must rule out a categorical point of view, I do not intend to imply that those studies that have a categorical approach (implied or otherwise) necessarily rule out a gradient approach. Many of those investigators may in fact be amenable to it. However, actual evidence of the necessity of a gradient approach in theory remains somewhat scarce. It is hoped that the present study will add to this evidence.

2.2 Computing Salience

Returning then to the model of pronoun resolution, the list of referents is first pruned to remove incompatible referents based on morphosyntactic features (Arnold et al., 2000; Boland et al., 1998). Then search for a referent should proceed with respect to the ranking (either categorical or gradient) of the referents in the list. But what determines this ranking? One of the most dominant factors has been shown to be syntactic prominence. However, in Rose (2005), I have argued that (in English, at least) syntactic and semantic information are often conflated. I will therefore discuss both of these factors below. In addition, another significant factor is coherence relations, also discussed this further below.

2.2.1 Syntactic Prominence

Several ways of determining the syntactic prominence of evoked entities have been discussed in the literature. These include left-to right ordering in which discourse referents introduced earlier are more prominent than those introduced later (Gernsbacher and Hargreaves, 1988), depth-first hierarchical search in which referents introduced higher and leftward in the syntactic tree are more prominent (Hobbs, 1978), and grammatical role in which referents introduced as subjects are more prominent than those introduced in other roles (Grosz et al., 1995). These different approaches typically make the same predictions when dealing with syntactically simpler constructions (i.e, no subordinate clauses or complex noun phrases), but may make different predictions with more complex constructions. The stimuli used in the experiment described below are all relatively syntactically simple and so in this paper I will not evaluate the differences among these various approaches.[1]

[1] See Rose (2005) for detailed discussion of these different approaches and a psycholinguistic and corpus linguistic comparison of hierarchical and grammatical role approaches.

However, for expository purposes, I will use grammatical role labels in discussion below.

When ranking referents according to the grammatical role in which they have been introduced, a categorical ranking predicts that the referent introduced as subject is salient and that any other referent is non-salient. This is the point of view implicitly taken in Stevenson et al. (2000), for example, in which they argue that a pronoun should refer to the referent introduced as subject of the preceding utterance.[2] A gradient salience approach, however, requires a more detailed prominence hierarchy such as that in (2). This is the point of view taken in most studies using the Centering framework (Grosz and Sidner, 1986; Grosz et al., 1995). An even more detailed gradient salience approach may be taken in which the points on the prominence hierarchy carry different (relative) weights. This is the approach taken in many practical applications such as the pronoun resolution algorithm of Lappin and Leass (1994).

(2) $subject > object > oblique > others$

2.2.2 Semantic Prominence

In English, syntactic role and semantic role are often conflated. That is, syntactic subjects are often semantic agents while syntactic objects are often semantic patients, and so on. Thus, it could be that the kind of pronoun resolution preferences previously observed and usually attributed to syntactic prominence effects might actually be attributed to semantic prominence effects. In other words, perhaps subject-preference is actually agent-preference.

In order to investigate this question, in Rose (2005), I used argument-reordering constructions: constructions which allow the order of arguments to vary—hence effecting a different relative syntactic prominence of discourse referents—with no (or minimal) change in their semantic role. For instance, so-called *psychological*-verbs have the alternate forms shown in (3)-(4).

(3) The audience admired the acrobats.
(4) The acrobats amazed the audience.

[2]More precisely, Stevenson et al. (2000), using the Centering framework (Grosz and Sidner, 1986; Grosz et al., 1995), argue that the backward-looking center, **Cb**, should refer to the subject of the preceding utterance. This is a simplification of the original Centering proposal in which it was suggested that the **Cb** refer to the highest-ranking member of the set of forward-looking centers in the previous utterance which is realized in the current utterance.

In (3), AUDIENCE is realized in a more syntactically prominent position than is ACROBATS: that is, in subject position. The reverse is true in (4). On the other hand, the semantic roles remain the same in both: ACROBATS is the stimulus while AUDIENCE is the experiencer. If syntactic prominence is most important, then a subsequent pronoun *they* should pick out AUDIENCE in (3) and ACROBATS in (4). On the other hand, if semantic prominence is most important, then a subsequent pronoun should pick out the same discourse referent in both alternatives. Assuming experiencer is higher on a semantic prominence hierarchy than stimulus (cf., thematic hierarchies in Jackendoff (1972); Speas (1990); inter alia), then this would be AUDIENCE.

In Rose (2005), I compared the effects of syntactic and semantic prominence on the salience of discourse referents in psycholinguistic experiments using two argument-reordering constructions: *tough*-constructions and *spray/load*-constructions. Results show that both syntactic and semantic prominence contribute to the salience of discourse referents. This suggests that experiments of this sort should carefully control for both syntactic and semantic prominence. In the present experiment, I do so by using an argument-reordering construction for the test stimuli.

2.2.3 Coherence Relations

Several investigators have theorized and observed that pronoun interpretation preferences differ when there is a causal connection between utterances compared to when there is a narrative connection (Hobbs, 1978; Kehler, 2002; Stevenson et al., 2000). For instance, in the narrative relation shown above in (1) (repeated below as (5)), the preference is for the pronoun to refer to LUKE. However, in (6) in which the utterances are related by a causal connection, the preference is for the pronoun to refer to MAX.

(5) Luke$_i$ hit Max$_j$ and then he$_{i/\#j}$ ran home.
(6) Luke$_i$ hit Max$_j$ because he$_{\#i/j}$ ran home.

Therefore, when investigating pronoun resolution, it is also necessary to take into account the influence of coherence relations by either controlling for these relations or making them another point of investigation. In the present study, I will take the latter course of action in order to see how coherence relations might influence pronoun resolution to competing non-salient entities. Previ-

ous accounts of the effects of coherence relations on pronoun resolution have taken the view that the kind of relation shifts attention to different aspects of the event being described (Stevenson et al., 1994; Stevenson et al., 2000). If an event has, for example, a start-state and an end-state, then a narrative relation will shift attention toward the start-state while a causal relation will shift attention toward the end-state. Subsequent pronominal reference will therefore prefer referents associated with these respective states, as illustrated in (5)-(6). Based on this argumentation, the prediction would be that pronominal reference might favor one non-salient referent over another if it is associated with that part of the event to which attention has been shifted by the coherence relation.

3 Experiment

Before describing the experiment, I'll review the primary and secondary questions which this experiment is designed to test. First, there is the question of what happens during pronoun resolution processes when there are competing non-salient referents. Answers to this question should provide evidence toward either a categorical or a gradient model of salience ranking. Furthermore, because investigating this question requires controlling for syntactic versus semantic prominence as well as coherence relation effects, two other secondary questions are also investigated. First, which is a more important factor in pronoun resolution: syntactic or semantic prominence? Second, what effect do coherence relations have on pronominal reference to non-salient entities?

3.1 Design

The research questions described above were investigated in this study using the well-known *spray/load*-constructions which exhibit the locative alternation (Levin, 1993) as shown in (7) and have synonymous alternative forms.[3]

(7) a. John sprayed some paint on a wall.
b. John sprayed a wall with some paint.

[3]There is some difference of opinion on whether the two forms of *spray/load*-constructions are actually synonymous. One central point of contention is whether the totality effects on the direct object (i.e., the judgment that the entity in direct object position is totally used up in the event) are consistent across both forms. In the judgment of Rappaport and Levin (1988), the totality effect applies only with the *with*-variant. In contrast, it is my judgment (Rose, 2005) and also that of Tenny (1994, see her data items (100) and (102)) that the effect applies across both forms.

According to prominence hierarchies in which the syntactic subject or the semantic agent is most prominent, then JOHN should consistently be regarded as the (most) salient referent while PAINT and WALL should be regarded as less or non-salient referents in these sentences. Thus, subsequent pronominal reference with the third-person singular pronoun, *it*, allows a test of the three different questions outlined above.

First, if a categorical approach to salience is sufficient, then there should be no overall preference for either PAINT or WALL. But if gradient salience is necessary for ranking, then it might be possible to observe a difference between the two.

The nature of this difference, however, might be more complex depending on the way salience ranking is determined. If syntactic prominence is the only relevant factor, then preferences should consistently favor the object (i.e, PAINT in (7a), WALL in (7b)) according to the well-established syntactic prominence hierarchy in (2) above. But if semantic prominence is the only factor, then preferences should favor either the theme (PAINT) or the location (WALL) depending on how the semantic prominence hierarchy is ordered. One prediction might be based on proposed thematic hierarchies (cf., Larson (1988), Speas (1990)) which place theme above location. According to such a hierarchy, PAINT should be consistently preferred. This is what I observed in Rose (2005).

Other differences may result from the kind of coherence relation used. However, for *spray/load*-constructions, this is a little difficult to predict. The two non-salient entities are both arguably a part of the end-state of the event—that is, together, they are the product of the agent's work. Thus, any motivation to distinguish between the two with respect to the coherence relation must come from some other feature of the event or its participants. I will address the possibility in the discussion section below.

3.2 Method

3.2.1 Participants

The participants in this experiment included 36 undergraduate students at Morehead State University in Kentucky. Students were recruited through fliers and classroom announcements and received five dollars for their participation.

3.2.2 Materials

Twenty-four stimulus items were prepared using *spray/load* verbs as the matrix verb. The agent/subject was a commonplace proper name (12 male and 12 female) and the themes and locations were all inanimate nouns presented in indefinite form. The *spray/load* sentence was then followed by one of two connectives: *and then* to force a narrative relation or *because* to force a causal relation. These connectives were then followed immediately by *it*. Each stimulus was then followed by a blank line for participants to fill in a completion for the sentence. The experiment was therefore a 2×2 design pitting ORDER of entities (theme-location or location-theme) against coherence RELATION (narrative or causal). (8) shows an example of the four variants of one stimulus item.

(8) a. John sprayed some paint on a wall
and then it _____
(theme-location, narrative)
b. John sprayed a wall with some paint
and then it _____
(location-theme, narrative)
c. John sprayed some paint on a wall
because it _____
(theme-location, causal)
d. John sprayed a wall with some paint
because it _____
(location-theme, causal)

Stimulus items were placed into twelve different tests such that each test contained only one variant of each item but conditions were balanced across all tests. The order of the items was pseudo-randomized such that consecutive items were not from the same experimental condition. The 24 items were combined with 101 items from an unrelated experiment to make a total of 125 items. Tests were printed in landscape orientation allowing every stimulus item to be followed by a blank line of at least three inches—ample space for participants to write their continuations.

3.2.3 Procedures

Participants were given the test forms and were asked to complete each sentence in the way that seemed most natural to them. Participants' responses were then analyzed and marked with one of four designators: If their completion showed that they interpreted the pronoun unambiguously as the theme of the *spray/load* verb then the response was marked THEME. Similarly, if they interpreted the pronoun as the location, then the response was marked LOCATION. If the response was ambiguous as to the participant's interpretation, then it was marked INDETERMINATE. Finally, if the response indicated pronominal reference to some other entity, or the pronoun was taken as an empty pronoun, then the response was marked OTHER.

3.3 Results

In total, there were 836 usable responses (23 responses were left blank and 5 were ungrammatical). 130 responses were judged INDETERMINATE and 65 responses were judged OTHER. Only the remaining 641 responses are therefore used in the analysis below.

In order to evaluate the results, it is useful to look at the participants' pronoun resolution preferences. However, there are two ways of looking at these preferences: syntactically or semantically. Thus, while it is somewhat more laborious for the reader, I will present the results from these two perspectives for the sake of completeness. The results are therefore presented in terms of object-preference as well as theme-preference. Object preference is calculated as the total number of choices for the object minus the total number of choices for the oblique. Theme-preference, on the other hand is calculated as the total number of choices for the theme minus the total number of choices for the location. These results by subjects and by items are shown in Table 1 and Table 2, respectively.

The results show that there was an overall preference for the location (i.e., *wall*) in both variants. This can be most readily seen by noting the consistently negative theme-preference values in Table 2. This is underscored by the significant main effect for ORDER in the object-preference results in contrast with the nonsignificant main effect for ORDER in the theme-preference results. This contrast also indicates that in this experiment, participants' pronoun resolution processes were guided by a salience ranking determined by semantic prominence and not syntactic prominence.

As for the main question of categorical versus gradient salience, the results point toward a gradient model of salience ranking. Participants showed a clear, consistent preference for one non-salient entity (location) over another (theme).

Table 1: Overall Results for Object-preference

by subjects

	ORDER	
	theme-location	location-theme
RELATION		
narrative	-1.75	2.50
causal	-0.89	1.14
Variant	$F(1, 35) = 58.2$	$p < 0.001$
Relation	$F(1, 35) < 1.0$	n.s.
Variant*Relation	$F(1, 35) = 8.4$	$p < 0.01$

by items

	ORDER	
	theme-location	location-theme
RELATION		
narrative	-2.62	3.75
causal	-1.33	1.71
Variant	$F(1, 23) = 18.3$	$p < 0.001$
Relation	$F(1, 23) < 1.0$	n.s.
Variant*Relation	$F(1, 23) = 3.2$	$p = 0.085$

Table 2: Overall Results for Theme-preference

by subjects

	ORDER	
	theme-location	location-theme
RELATION		
narrative	-1.75	-2.50
causal	-0.89	-1.14
Variant	$F(1, 35) = 2.8$	$p = 0.10$
Relation	$F(1, 35) = 8.4$	$p < 0.01$
Variant*Relation	$F(1, 35) < 1.0$	n.s.

by items

	ORDER	
	theme-location	location-theme
RELATION		
narrative	-2.62	-3.75
causal	-1.33	-1.71
Variant	$F(1, 23) = 2.2$	$p = 0.15$
Relation	$F(1, 23) = 3.2$	$p = 0.085$
Variant*Relation	$F(1, 23) < 1.0$	n.s.

Finally, the results pertaining to coherence relations are somewhat inconclusive. In order to discuss this, it is better to refer to the theme-preference results because semantic prominence has proven to be the dominating factor here. While there is a significant main effect of RELATION by subjects, the effect is, at best, marginal by items. It is possible that a more thorough investigation with more items could yield a clear, significant result. On the other hand, even if the current effect is somehow real, it is actually quite weak. Note that the theme-preference values. which are negative in the narrative condition, are merely less negative in the causal condition—not enough to flip-flop resolution preferences. So, it seems difficult to make the case here that coherence relations shift these preferences in any meaningful way.

4 Discussion

In the present study, there were three questions under investigation. Let me review these three questions in turn and what the results say about them. First there was the primary question of categorical versus gradient approaches to salience ranking. The results here are not consistent with a categorical approach and clearly suggest a gradient approach. In this respect, the study lends psycholinguistic support to the many implementations of pronoun resolution algorithms which incorporate a gradient ranking of candidate referents for resolution (e.g., Kennedy and Boguraev (1996); Lappin and Leass (1994)).

However, just how fine-grained an approach is necessary is not conclusive from this investigation since competition among only two non-salient referents was tested. A more thorough study with stimuli including a large number of referents would be necessary to draw further conclusions about the necessity of a fine-grained gradient model of salience ranking.

The second question in this study was the question of whether syntactic prominence or semantic prominence is more important for determining the salience of referents. Results quite clearly point toward semantic prominence. These results contrast with those of Rose (2005) in two ways. The psycholinguistic results in that study suggest first that *both* syntactic and semantic prominence play a role in determining salience and second that theme is higher than location on the semantic prominence hierarchy.

The first contrast might be attributed to differences in experimental technique. The fact that participants in the present experiment had to compose a completion to each sentence means that they may have spent more time focusing on the semantic representation of the situation. This may have inflated the semantic prominence effects while attenuating syntactic prominence effects.

The second contrast, however, is somewhat more difficult to resolve. But once again, it may be useful to appeal to differences in the experimental technique. In the present study, the process of composing a sentence continuation for the events described by *spray/load* verbs would have required visualizing the event in a more vivid way than might be required for mere reading comprehension. If this visualization process were to require participants to anchor their visualizations through some fixed objects in the representation, this might naturally lead them toward paying closer attention to the location than the theme. Further testing will be required to evaluate this hypothesis and disambiguate these contrasting results.

Finally, the third question in this study dealt with the influence of coherence relations on pronoun resolution to competing non-salient referents. The present study did not test this in a manner comparable to previous studies since unlike those studies, both target referents were associated with the end-state of the event. Nonetheless, results showed a weak (but inconclusive) tendency to shift resolution preferences from location toward (but not to) theme. While more evidence would be necessary to confirm this to be a real effect, if it does turn out to be real then it would be a very interesting result. Assuming for the sake of argument that it is, then this might suggest that participants do not see the theme argument of *spray/load* verbs as part of the end-state of the event. To illustrate how this might be so, consider a couple of examples. Once a bucket of paint has been sprayed onto a wall, it takes on certain properties of the wall—for instance, its texture and size. Similarly, hay loaded onto a cart also takes on certain properties of the cart such as its size and shape. It might then be the case that the end-state of a *spray/load* event is more centrally focused on the location argument than on the theme argument because it is the location which determines many of the main properties of the end-state.

Before concluding, I would like to suggest some applications of these findings. Computational implementations of resolution algorithms that use an explicit salience ranking mechanism can be adapted to incorporate semantic prominence information as one of the contributors to a candidate's overall salience index (e.g., as in Lappin and Leass (1994)). However, even implementations that do not have an explicit salience ranking mechanism might still incorporate semantic prominence information. The coreference resolution system described in Soon et al. (2001) and its more knowledge-rich extension in Ng and Cardie (2002) classify NP pairs as coreferent or not based on constraints learned from an annotated corpus. These constraints are based on a number of features. While the Ng and Cardie system does incorporate a syntactic role feature (i.e., whether or not either NP in a pair is a subject), neither system incorporates a semantic prominence feature. It would be interesting to see if any further gains could be made in these systems by incorporating such a feature in future work.

5 Conclusion

The main aim of this paper has been to explore the question of whether a gradient model of salience ranking for candidate referents in pronoun resolution is necessary, or if a categorical model is sufficient. In this endeavor, two other questions have been addressed along the way: the influence of syntactic and semantic prominence on salience ranking of referents and the influence of coherence relations on pronoun resolution preferences. Results point toward the necessity of a gradient model of salience in which salience ranking is primarily determined by semantic information. Results were inconclusive regarding the influence of coherence relations. However, further work is necessary to confirm that this is the case.

Acknowledgments

I am indebted to Stefan Kaufmann and Michael Dickey for their guidance on this research project and also to Katy Carlson for her help on the experimental portion of this study. I also express my gratitude to three anonymous reviewers whose constructive comments on an earlier draft were most helpful.

References

Jennifer Arnold, Janet Eisenband, Sarah Brown-Schmidt, and John Trueswell. 2000. The rapid use of gender information: evidence of the time course of pronoun resolution from eyetracking. *Cognition*, 76:B13–B26.

Julie Boland, M. Acker, and L. Wagner. 1998. The use of gender features in the resolution of pronominal anaphora. Cognitive Science Technical Report 17, The Ohio State University Center for Cognitive Science.

Morton Ann Gernsbacher and D. Hargreaves. 1988. Accessing sentence participants: The advantage of first mention. *Journal of Memory and Language*, 27:699–717.

Talmy Givón. 1983. Topic continuity in discourse: An introduction. In Talmy Givón, editor, *Topic Continuity in Discourse: A Quantitative Cross-Language Study*, pages 1–42. John Benjamins, Amsterdam.

Peter Gordon, Barbara Grosz, and Laura Gilliom. 1993. Pronouns, names, and the centering of attention in discourse. *Cognitive Science*, 17:311–347.

Barbara Grosz and Candace Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204.

Barbara Grosz, A. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21:203–225.

Irene Heim. 1982. *The Semantics of Definite and Indefinite Noun Phrases*. Ph.D. thesis, University of Massachusetts, Amherst.

Graeme Hirst. 1981. *Anaphora in Natural Language Understanding: A Survey*. Springer-Verlag, Berlin.

Jerry Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311–338.

Susan Hudson-D'Zmura and Michael Tanenhaus. 1997. Assigning antecedents to ambiguous pronouns: The role of the center of attention as the default assignment. In M. Walker, A. Joshi, and Ellen Prince, editors, *Centering Theory in Discourse*, pages 199–226. Clarendon Press, Oxford.

Ray Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press, Cambridge, MA.

Lauri Karttunen. 1976. Discourse referents. In James McCawley, editor, *Syntax and Semantics, Vol. 7: Notes from the Linguistic Underground*, pages 363–385. Academic Press, New York.

Andrew Kehler. 2002. *Coherence, Reference, and the Theory of Grammar*. CSLI Publications, Stanford University, CA.

Chris Kennedy and Branamir Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, pages 113–118, Copenhagen, Denmark.

Shalom Lappin and Herbert Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20:535–561.

Richard Larson. 1988. On the double object construction. *Linguistic Inquiry*, 19:335–392.

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

Alison Mathews and Martin Chodorow. 1988. Pronoun resolution in two-clause sentences: Effects of ambiguity, antecedent location, and depth of embedding. *Journal of Memory and Language*, 27:245–260.

Ruslan Mitkov. 2002. *Anaphora Resolution*. Longman, London.

Vincent Ng and Clarie Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40^{th} Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111, Philadelphia, PA.

Malka Rappaport and Beth Levin. 1988. What to do with θ-roles. In Wendy Wilkins, editor, *Syntax and Semantics 21: Thematic Relations*, pages 7–36. Academic Press, New York.

Ralph Rose. 2005. *The Relative Contribution of Syntactic and Semantic Prominence to the Salience of Discourse Entities*. Ph.D. thesis, Northwestern University.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27:521–544.

Margaret Speas. 1990. *Phrase Structure in Natural Language*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Rosemary Stevenson, Rosalind Crawley, and David Kleinman. 1994. Thematic roles, focus and the representation of events. *Language and Cognitive Processes*, 9:519–548.

Rosemary Stevenson, Alistair Knott, Jon Oberlander, and Sharon McDonald. 2000. Interpreting pronouns and connectives: Interactions among focusing, thematic roles and coherence relations. *Language and Cognitive Processes*, 15(3):225–262.

Carol Tenny. 1994. *Aspectual Roles and the Syntax-Semantics Interface*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Web Readability and Computer-Assisted Language Learning

Alexandra L. Uitdenbogerd
School of Computer Science and Information Technology
RMIT University
Melbourne, Australia

Abstract

Proficiency in a second language is of vital importance for many people. Today's access to corpora of text, including the Web, allows new techniques for improving language skill. Our project's aim is the development of techniques for presenting the user with suitable web text, to allow optimal language acquisition via reading. Some text found on the Web may be of a suitable level of difficulty but appropriate techniques need to be devised for locating it, as well as methods for rapid retrieval. Our experiments described here compare the range of difficulty of text found on the Web to that found in traditional hard-copy texts for English as a Second Language (ESL) learners, using standard readability measures. The results show that the ESL text readability range fall within the range for Web text. This suggests that an on-line text retrieval engine based on readability can be of use to language learners. However, web pages pose their own difficulty, since those with scores representing high readability are often of limited use. Therefore readability measurement techniques need to be modified for the Web domain.

1 Introduction

In an increasingly connected world, the need and desire for understanding other languages has also increased. Rote-learning and grammatical approaches have been shown to be less effective than communicative methods for developing skills in using language (Higgins, 1983; Howatt, 1984; Kellerman, 1981), therefore students who need to be able to read in the language can benefit greatly from extensively reading material at their level of skill (Bell, 2001). This reading material comes from a variety of sources: language learning textbooks, reading books with a specific level of vocabulary and grammar, native language texts, and on-line text.

There is considerable past work on measuring the readability of text, however, most of it was originally intended for grading of reading material for English-speaking school children. The bulk of readability formulae determined from these studies incorporate two main criteria for readability: grammatical difficulty — usually estimated by sentence length, and vocabulary difficulty, which is measured in a variety of ways (Klare, 1974). Publishers later decided to use the readability measures as a guideline for the writing of texts, with mixed success. However, new reading texts catering for foreign language learners of various languages are still being published. Most of these use specific vocabulary sizes as the main criterion for reading level. Others are based on specific language skills, such as the standard developed by the European community known as the "Common European Framework of Reference for Languages" (COE, 2003).

The goal of our research is to build an application that allows the user to improve their language skills through accessing appropriate reading material from the Web. This may incorporate personalised retrieval based on a user's level of skill in the target language, first language and specific vocabulary of interest. Greater detail about the application's requirements and potential implementation issues are discussed elsewhere (Uitdenbogerd, 2003).

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 99–106.

In order for appropriate documents to be presented to the user for reading practice, new readability measurement techniques that are more appropriate to on-line documents will need to be developed. Measures of distance between languages that are related to reading may be useful for finer-tuned readability (as opposed to the speaking-based measure developed elsewhere (Chiswick, 2004)). For many language pairs, cognates — words that are similar in both languages, help people to understand text. There is some evidence that these affect text readability of French for English speakers (Uitdenbogerd, 2005). Automatic detection of cognates is also part of our research program. Some work exists on this topic (Kondrak, 2001), but will need to be tested as part of readability formulae for our application.

Some applications that allow the location or sorting of suitable on-line reading material already exist. One example is Textladder (Ghadirian, 2002), a program that allows the sorting of a set of texts based on their vocabulary, so that users will have learnt some of the words in earlier texts before tackling the most vocabulary-rich text in the set. However, often vocabulary is not the main criterion of difficulty (Si and Callan, 2001; Uitdenbogerd, 2003; Uitdenbogerd, 2005). SourceFinder (Katz and Bauer, 2001) locates materials of a suitable level of readability given a list of URLs. It is simply a crawler that accepts a web page of URLs such as those produced by Google, and then applies a readability measure to these to rank them. The software was developed with the aim of finding material of the right level of difficulty for school children learning in their native language.

Using English as a test case, the research questions we raise in this work are:

- What is the range of difficulty of text on the web?
- How does the range of text difficulty found on the web compare to texts especially written for language learners?

If there is overlap in the readability ranges between web documents and published ESL texts, then the combination of the two may be adequate for language learning through reading once learners are able to comfortably read published texts. In fact, we have found that ESL texts fit within the range of readability found on the Web, but that there are problems with assessing readability of the Web pages due to the types of structures found within them.

In future work we intend to develop readability formulae that take into account bulleted lists and headings. It is known from usability studies that these increase readability of text for native readers of technical documents (Redish, 2000; Schriver, 2000). We will then be in a position to better determine how the readability factors differ for people with different language backgrounds and skills within a Web context. We have already examined the case of French as a foreign language for those whose main language is English and found that standard readability formulae developed for native English speakers are less closely correlated to French reading skill than a simple sentence length measure (Uitdenbogerd, 2005). However, this work was based on prose and comic-book text samples, not HTML documents.

This article is structured as follows. We review the literature on language learning via reading, as well as describe past research on readability. We then describe our current work that examines the readability of English text on the Web. This is compared to the readability of reading books for students with English as a second language. These results are then discussed in the context of improving language skills via the Web.

2 BACKGROUND

Two main research areas are of relevance to the topic of computer-assisted language acquisition via reading: readability and language acquisition. Readability measures allow us to quickly evaluate the appropriateness of reading material, and language acquisition research informs us how best to use reading material in order to acquire language.

2.1 Readability

Readability has been studied for most of the twentieth century, and has more recently become a topic of interest to information retrieval researchers. There have been several phases in its development as a research topic. In the initial and most influential era, readability measures were developed by applying regression to data collected from children's comprehension tests. Later, Cloze tests were used as a simpler method of collecting human readability data (Bormuth, 1968; Davies, 1984). The output of this era included a vast ar-

ray of formulae, mostly incorporating a component representing vocabulary difficulty, such as word length, as well as a grammatical difficulty component, which usually is represented by sentence length (Klare, 1974). The majority of published work was on English language readability for native speakers, however, some work from this era examined other languages, again in a native speaker context. More recently the language modelling approach has been applied to readability estimation of text (Si and Callan, 2001).

Despite the success of the techniques, they fell out of favour within some research and education communities due to their simplicity (Chall and Dale, 1995; Redish, 2000; Schriver, 2000) and failure to handle hand-picked counter-examples (Gordon, 1980). Other criticism was of their abuse in writing texts or in enforcing reading choices for children (Carter, 2000). Researchers tried to capture more complex aspects of readability such as the conceptual content. Dale and Chall, in response to the criticism, updated their formula to, not only use a more up-to-date vocabulary, but to allow conceptual content to be catered for. They emphasized however, that grammatical and vocabulary difficulty are still the dominant factors (Chall and Dale, 1995). In work on readability for English-speaking learners of French, we found further evidence that conceptual aspects are of minor importance compared to grammatical complexity (Uitdenbogerd, 2005). For example, the well-known fairy tale Cinderella was consistently perceived as more difficult than many unknown stories, due to the relative grammatical complexity.

The readability measures used in the experiments reported here are those implemented in the unix-based `style` utility. The measures used were the Kincaid formula, Automated Readability Index (ARI), Coleman-Liau Formula, Flesch reading ease, Fog index, Lix, and SMOG. Some readability formulae are listed below.

The ARI formula as calculated by `style` is:

$$ARI = 4.71 * Wlen + 0.5 * WpS - 21.43 \quad (1)$$

where Wlen is the length of the word and WpS is the average number of words per sentence.

The Flesch formula for *reading ease* (RE) as described by Davies, is given as:

$$\begin{aligned} \mathrm{RE} = 206.835 \quad & -(0.846 \times \mathrm{NSYLL}) \\ & -(1.015 \times \mathrm{W/S}) \ , \end{aligned} \quad (2)$$

where NSYLL is the average number of syllables per 100 words and W/S is the average number of words per sentence (Davies, 1984).

The Dale-Chall formula (not used in our experiments) makes use of a vocabulary list in addition to sentence length:

$$S = 0.1579p + 0.0496s + 3.6365 \ , \quad (3)$$

where p is the percentage of words on the Dale list of 3,000, and s is the average number of words per sentence. The resulting score represents a reading grade.

The above formulae illustrate three ways of determining vocabulary difficulty: word length in characters, number of syllables, and membership of a list of words known by children with English as their native language. Most formulae use one of these techniques in addition to the sentence length.

More recent research into readability has involved the application of language models (Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005). Using unigram models allowed very small samples of text to be used to predict a grade level for the text (Collins-Thompson and Callan, 2004). The technique was shown to be more robust than a traditional readability measure for estimating web page readability. However, the unigram approach is unlikely to be effective for the case of foreign languages, where grammatical complexity is a much more important factor than vocabulary for at least one language pair (Uitdenbogerd, 2005).

Schwarm and Ostendorf (2005) built a readability classifier that incorporated a wide variety of features, including traditional readability measure components, as well as n-gram models, parse-tree based features to model grammatical complexity, and features representing the percentage of unusual words. The classifier was trained and evaluated using articles written for specific grade levels. It is possible that the approach and feature set used may be applicable to foreign language learning.

2.2 Second and Foreign Language Acquisition via Reading

The idea of language acquisition via reading at an appropriate level was first formally studied

by Michael West. He found that his techniques of English teaching with Bengali boys were far more successful than other approaches of the time (West, 1927). He controlled the introduction of vocabulary to no more than one new word per 60 words of text. The concept remains with us today and is known as "controlled-vocabulary". Occasionally the reading approach falls out of favour and conversation becomes a more prominent technique. Then reading is rediscovered (Kellerman, 1981).

Studies of different ways of reading for language acquisition conclude that extensive reading at a comfortable level is superior to intensive reading at a more challenging level (Bell, 2001), and the use of glosses and multimedia improve vocabulary acquisition (Lomicka, 1998; Al-Seghayer, 2001). Looking up word meanings is more likely to lead to retention, but words can be learnt through repeated exposure and meaning inference. However, due to the need for repetition, inference is only useful for fairly common words (Krantz, 1991).

3 EXPERIMENTS

The main experiment that we discuss here is an analysis of a corpus of English web text. The corpus is a subset of the TREC web 10G collection consisting of 93,064 documents. The collection is a general snapshot of the web, including a wide variety of types of web pages.

We extracted the text and punctuation from each document in the corpus, and applied several standard readability measures to them, as implemented by the unix-based `style` utility. The measures used were the Kincaid formula, ARI, Coleman-Liau Formula, Flesch reading ease, Fog index, Lix, and SMOG.

In a second experiment we applied the same readability measures to extracts from reading books written for students of English as a second or foreign language. The statistics for the two sets of text were compared.

Results

The first part of Table 1 shows statistics describing the range of readability scores found in the collection. For the Flesch Index, the highest value represents the easiest to read, whereas for the other measures the lowest value is the easiest.

It is clear by looking at the extreme values that there are difficulties in processing web documents compared to normal text. In this section we look at the types of problem documents that are classified as very easy by a naïve application of readability formulae.

Documents with extreme scores

We examined several web documents that had extreme values for each readability measurement type.

All measures except Coleman-Liau agreed as to which was the hardest document in the collection — a large document listing access statistics of Internet domains. There were few true sentences in this document.

There were many documents (49) that had the minimum score of -3.4 using the *Kinkaid* measurement. On close inspection of a couple of these, we found they were devoid of punctuation, containing a few headings and links only. The same documents received the maximum (easiest) score of 121.2 in the *Flesch* reading ease measure.

The *Fog* measure also shared the same easiest documents, however, it also included other documents amongst those with its lowest score of 0.4. An example that was in this set of extra documents was a page of links to images, with duration times listed next to the image. The only terminating punctuation was in an email address. The *Lix* measure had a similar but not identical set of 48 documents receiving its lowest score of 1.

Two documents received the lowest value -12.1 using the *ARI* measure. In the first, the only untagged text was within `title` tags: "V.I.She: Pharmacy". The second document contained the same title and some labelled links without punctuation.

The lowest value using the *Coleman-Liau* measure was associated with a short document in which most of the words had their letters interspersed with spaces, for example "C H A N G I N G". The second lowest consisted of a heading, links to images, with their sizes, such as "127.1 KB" shown next to them, and a single sentence.

The SMOG score was less discriminating, giving 2,967 documents the same lowest score of 3.

Published Reading Books

The second part of Table 1 shows the readability of nine published ESL books. Interestingly, the readability results bear little resemblance to the levels advertised by the publishers. For example,

Table 1: Distribution of readability scores in the Web collection and of a range of books written for learners of ESL. For all measures except Flesch, the highest value represents the most difficult text to read. The ESL book measured as most and least difficult are indicated with a † and an asterisk ('*') symbol respectively.

Quartile	Kinkaid	ARI	Coleman-Liau	Flesch	Fog	Lix	SMOG
Min	-3.4	-12.1	-8.3	-62809.2	0.4	1	3
LQ	6.4	7.4	11.1	46.2	9.4	37	8.9
Med	9.1	10.7	13.2	60.8	12.3	46	10.9
UQ	12.2	14.3	15.6	73.4	15.8	55.3	13.2
Max	24174.6	30988	130.4	121.2	24798	61997.5	219.6
Average	11.809	14.20	13.4176	54.09	15.13571	52.1665	11.28505
Book	Kinkaid	ARI	Coleman-Liau	Flesch	Fog	Lix	SMOG
card	5.3	†6.1	†9.0	†84.7	†8.2	28.6	†7.8
christmas	3.5	2.9	8.2	88.5	5.8	22.1	6.6
dead	1.1	-0.0	6.4	100.6	3.8	16.6	5.2
ghost	3.4	3.1	7.6	91.3	6.1	24.0	6.5
lovely	2.5	2.7	7.5	96.9	5.0	21.5	5.2
murders	†5.4	5.9	7.7	86.7	7.8	28.7	6.5
presidents	*0.2	-0.1	6.2	*107.0	3.2	14.0	*4.2
simon	1.3	*-0.9	*5.9	97.0	*3.1	*12.6	4.7
thirty	5.2	5.3	7.2	87.7	7.9	†28.9	7.0
Min	0.2	-0.9	5.9	84.7	3.1	12.6	4.2
Max	5.4	6.1	9.0	107.0	8.2	28.9	7.8

Table 2: Web pages out of $93,064$ with readability scores within the range of sample ESL texts.

	Kinkaid	ARI	Coleman-Liau	Flesch	Fog	Lix	SMOG
Count	16123	17321	7863	8176	14748	8166	11344
Percent	17	19	8	9	16	9	12

The Card is described as a level 3 story with 1,000 headwords, making it in the middle of the range of 5 levels of difficulty. However, five of the readability measures identified it as the most difficult book of the set. In contrast, Simon the Spy and The President's Murder are both identified as easy texts, which is in agreement with the advertised beginner level of these stories.

When the levels are compared to those of the analysed web pages, it is clear that the ranges fall well within the extremes found on the web. However, as we have already seen, these extremes are often pathological cases, and not usually of interest for reading practice. As a percentage, the set of suitable texts for those that require the reading level found in ESL books, is probably quite small, given that the lower quartiles of web readability exceed the maximum scores for the range of books tested. In fact, depending on the reference readability measure, the percentage of web texts falling within the same range of readability is in the range 8 to 19% (See Table 2 for details).

In Figure 1 we show a few examples of web text that fall in the range of readability found in the ESL texts. These examples illustrate a few types of content found in web pages: links and message headers.

Discussion

While there is a wide range of values for the readability measures in the web collection studied, a very large proportion of documents with low scores are arguably not very useful for reading practice. The `style` utility assumes that the input consists of normal text in the form of sentences. If these are not found, or if there are too many non-sentences in the document, then the utility fails. In addition, documents that do contain sufficient text may still consist largely of headings, links, and lists. It is unclear how useful these documents would be for reading practice.

For a reading recommender to be successful, further criteria than just a readability score will be needed. Some preprocessing of the documents for better readability assessment may be necessary. It was observed in the documents receiving low scores that there were sentences without punctuation. Web authors often include instructions without punctuation, both within links and in normal displayed text. Some examples found in the

low-scoring documents are “Click on books”, “To view Shockedmovies, you need to have Netscape 2.0 and the Shockwave plug-in”, “Last modified on December 10, 1995”, and “Update Your Profile”. Inserting punctuation may make readability scores more reliable, however, automatic techniques for deciding when to do so are not completely obvious. As mentioned in the introduction, readability measures that take into consideration the use of bulleted lists and headings would be of utility for web page assessment, since these structures are frequently used in web pages, and often are the only textual content within a page. Collins-Thompson and Callan’s approach avoids this issue by using unigram word models exclusively to measure readability (Collins-Thompson and Callan, 2004). However, for the bilingual case, particularly in language pairs such as French and English, this is likely to be ineffective (Uitdenbogerd, 2005).

An alternative approach is to filter the pool of web pages to be analysed, either by crawling suitable subdomains, or by applying a set of rules to ensure sufficient suitable text on a page before inclusion.

Another important consideration is how interesting the document will be to the user, as a person’s comprehension skills vary with their interest in the text. Indeed, the documents should be sufficiently interesting for users to want to use the proposed system. An existing technique for increasing the chance of interesting documents being presented to the user is collaborative filtering, which relies on user feedback, whether explicit or implicit. Another possibility involves the examination of content words and phrases within documents.

4 CONCLUSIONS

The purpose of this preliminary work towards a utility for assisting users in improving foreign language skills via reading, was to find evidence that sufficient documents of suitable readability are likely to exist on the web. We determined the readability of over 90,000 web pages written in English, using the unix `style` utility and found a considerable range of readability scores. The range of readability scores found in ESL books fell within the lower quartile of web page readability scores, representing 8 to 19% of documents in the collection. This could mean that there are many suitable pages for reading practice which a readability-based reading recommender system could retrieve for users. However, due to the artifacts of web pages and the readability measures, not all pages with low scores in readability are suitable for reading practice. The automated location of those that *are* suitable is part of the future research plans of this project. An additional factor that must be incorporated is prediction of how interesting the documents are likely to be for users.

Our analysis used web pages written in English and compared these to ESL texts under the broad assumption that similar distributions of readability would occur in other languages. However, cultural and political differences of the countries speaking different languages may influence the types of text available, and hence the readability range.

Learners of English are relatively fortunate in that there are many reading books specifically written for them. This is not the case for many other languages. It is possible that the Internet may be an even more important reading resource for languages other than English.

Acknowledgements

We thank Vasundhara Srikantha for her preliminary work on web text analysis.

References

K. Al-Seghayer. 2001. The effect of multimedia annotation modes on L2 vocabulary acquisition: a comparative study. *Language Learning and Technology*, 5(1):202–232, January.

T. Bell. 2001. Extensive reading: speed and comprehension. *The Reading Matrix*, 1(1), April.

J. R. Bormuth. 1968. Cloze test readability: criterion reference scores. *Journal of Educational Measurement*, 5:189–196.

B. Carter. 2000. Formula for failure. *School Library Journal*, 46(7):34–37, July.

J. S. Chall and E. Dale. 1995. *Readability revisited: the new Dale-Chall readability formula*. Brookline Books, Massachusetts, USA.

B. R. Chiswick. 2004. Linguistic distance: A quantitative measure of the distance between English and other languages. Technical Report 1246, Institute for the Study of Labor (IZA).

2003. Common European framework of reference for languages: Learning, teaching, assessment. `http://www.coe.int/T/DG4/Linguistic/CADRE_EN.asp#TopOfPage`. Accessed 8 September, 2006.

K. Collins-Thompson and J. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the HLT/NAACL 2004 Conference*, pages 193–200, Boston.

A. Davies. 1984. Simple, simplified and simplification: what is authentic? Applied linguistics and language study, chapter 9, pages 181–198. Longman.

S. Ghadirian. 2002. Providing controlled exposure to target vocabulary through the screening and arranging of texts. *Language Learning and Technology*, 6(1):147–164, January.

R. M. Gordon. 1980. The readability of unreadable text. *English Journal*, pages 60–61, March.

J. Higgins. 1983. Can computers teach? *Calico Journal*, pages 4–6, September.

A. P. R. Howatt. 1984. *A history of English language teaching*. Oxford University Press, Oxford, UK.

I. R. Katz and M. I. Bauer. 2001. Sourcefinder: Course preparation via linguistically targeted web search. *Educational Technology and Society*, 4(3):45–49.

M. Kellerman. 1981. *The forgotten third skill*. Pergamon Press, Oxford.

G. R. Klare. 1974. Assessing readability. *Reading Research Quarterly*, X:62–102.

G. Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In K. Knight, editor, *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 103–110, Pittsburgh, PA, USA, June. NAACL.

G. Krantz. 1991. *Learning vocabulary in a foreign language: a study of reading strategies*. Ph.D. thesis, University of Göteborg, Sweden.

L. L. Lomicka. 1998. "to gloss or not to gloss": an investigation of reading comprehension online. *Language Learning and Technology*, 1(2):41–50, January.

J. Redish. 2000. Readability formulas have even more limitations than Klare discusses. *Journal of Computer Documentation*, 24(3):132–137, August.

K. A. Schriver. 2000. Readability formulas in the new millennium: What's the use? *Journal of Computer Documentation*, 24(3):138–140.

S. E. Schwarm and M. Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the Association for Computational Linguistics*.

L. Si and J. Callan. 2001. A statistical model for scientific readability. In L. Liu and D. Grossman, editors, *Proc. International Conference on Information and Knowledge Management*, volume 10, pages 574–576, Atlanta, Georgia, USA, November. ACM, ACM.

A. L. Uitdenbogerd. 2003. Using the web as a source of graded reading material for language acquisition. In W. Zhou, P. Nicholson, B. Corbitt, and J. Fong, editors, *International Conference on Web-based Learning*, volume 2783 of *Lecture Notes in Computer Science*, pages 423–432, Melbourne, Australia, August. Springer.

A. L. Uitdenbogerd. 2005. Readability of French as a foreign language and its uses. In A. Turpin and R. Wilkinson, editors, *Australasian Document Computing Symposium*, volume 10, December.

M. West. 1927. The construction of reading material for teaching a foreign language. *Dacca University Bulletin*, (13). Republished as part of "Teaching English as a foreign language, 1912 – 1936: Pioneers of ELT, Volume V: Towards Carnegie", R. Smith editor.

a)

```
You must complete at least 9 credits of graduate work with a GPA of
3.00 (B) and not more than one grade of B-.

Back to Communicative Disorders

Back To The Graduate Programs Catalog Page

Back To The Graduate Catalog Page
Back To The Catalog Home Page

Back To The UWSP Home Page
```

b)

```
Exchange logo
[ Post Message [post] ] [ Home [/index.html] ] [ Newsgroups [USENET]
]
- gold rule -
Li'l Builder [/entryform.html]

Did You Win? [/win1196.html] November's Software Award generously
provided by Borland International

December's Giveaway [/entryform.html] Sponsored by: Net-It Software,
makers of Net-It Now!

- gold rule - To win great intranet software, register
[/entryform.html] once, then post [post] at least twice a week each
month. Winners are chosen based on the quality and frequency of
contributions.
The Intranet Exchangesm
Intranet Standards [msg/1120.html] - Dan Boarman 16:03:36 12/31/96
(0)
How can I open EXCEL file from CGI ? [msg/1108.html] - Katsumi Yajima
12:03:34 12/30/96 (1)
Re: How can I open EXCEL file from CGI ? [msg/1118.html] - Brett
Kottmann 15:40:08 12/31/96 (0)
Telecommuting on intranet [msg/1092.html] - Erick Pijoh 07:57:16
12/29/96 (7)
Re: Telecommuting on intranet [msg/1119.html] - Brett Kottmann
15:57:36 12/31/96 (0)
```

Figure 1: Sample Web Documents with Readability Matching Typical ESL Texts. Both the above documents received a score of 5.9 on the Coleman-Liau readability measure, thus equivalent to the easiest ESL texts in our study. Item a) shows the complete text extracted from the document. Item b) is an extract of the document with the largest number of words and a score of 5.9.

Questions require an answer: A deductive perspective on questions and answers

Willemijn Vermaat
Centre for Logic, Language and Computation
Victoria University Wellington, New Zealand
vermaat@mcs.vuw.ac.nz

Abstract

Direct questions such as "Who saw Mary?" intuitively request for a certain type of answer, for instance a noun phrase "John" or a quantified noun phrase such as "A man". Following the structured meaning approach to questions, we propose an analysis of wh-questions in type-logical grammar that incorporates the requirement for a certain type of answer into the type assigned to wh-phrases. Interestingly, the syntactic and semantic decomposition leads to a derivability pattern between instances of wh-phrases. With this pattern we can explain the difference between wh-pronouns ('who') and wh-determiners ('which'), and derive wh-questions that require multiple answers.

1 Introduction

In this paper, we discuss the uniform basis of different types of wh-questions focusing on the dependency relation between questions and answers. In loose terms, a wh-question can be interpreted as a sentence which still requires an answer. The answer to a question such as "'Who saw Mary?" serves as an argument of the main or embedded verb clause. In more formal terms, the meaning assembly of the above wh-question may be represented by the lambda term, $\lambda x.((\textbf{see}\ \textbf{m})\ x)$. We show that by incorporating the dependency relation between questions and answers into the lexical type-assignments of wh-phrases, wh-questions can be instantiated in an uniform way.

Section 2 gives a short introduction in type-logical grammar and introduces the basic setup of the grammatical reasoning system. In section 3, we briefly discuss two approaches to the semantics of questions: the proposition set approach and the structured meaning approach. Section 4 provides the syntactic and semantic type of wh-questions and introduces a wh-type schema to identify wh-phrases. Additionally, we show how meaning assembly for wh-questions is derived on the basis of a structured meaning approach to questions and answers. In section 5, we show how we can derive type alternations for wh-phrases which lead to derivability schemata between instances of wh-type schema. Finally, in section 6, we analyze different types of question-answer combinations and multiple wh-questions in English on the basis of these derivability schema. We finish with the conclusion and some pointers for future research.

2 Type-logical grammar

Type-logical grammar (Moortgat, 1997) offers logical tools that can provide an understanding of both the constant and the variable aspects of linguistic form and meaning.[1] Type-logical grammar is a strongly lexicalised grammar formalism, which, in the case of a categorial system, means that a derivation is fully driven by the types assigned to lexical elements: these types are the basic declarative units on which the computational system acts. The basis for the type system is a set of atomic or basic types. The full set of types is then built out of these basic types by means of a set of type-forming operations. We consider unary and binary type-forming operations. The unary type-forming operations are $\diamondsuit$ (diamond) and $\square$ (box). The binary ones are the two slashes $/, \backslash$ (forward and backward slash) and $\bullet$ (prod-

[1] Combinatory categorial grammar (Steedman, 2000) is a related approach with a comparable notation. However, note the differences in notation and the proof-theoretic setup.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 107–114.

uct). In this paper, we will only consider the binary operators concentrating on the meaning of wh-questions. The unary operators are not visible in the analyses discussed in this paper, but play a role in deriving the right word-order of the wh-expressions. The inductive definition below characterises the full set of types built out of a set of atomic or basic types A.

$$\mathcal{F} ::= \mathcal{A} \mid \mathcal{F}/\mathcal{F} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F}\backslash\mathcal{F} \mid \Box\mathcal{F} \mid \Diamond\mathcal{F}$$

The type system is used to classify groups of expressions with a similar grammatical behavior. An expression belongs to a certain category depending on its grammatical relation to other expressions. The basic categories n, np and s are used to classify for *nouns*, *noun phrases* and *sentences*, expressions that are complete in themselves, i.e. expressions for which we have grammaticality judgments that do not depend on their relation with other expressions. Slash categories express incompleteness with respect to some other expressions. A product category represents the composition of two expressions. An expression of category A/B is incomplete with respect to an expression of category B on its right (symmetrically for $B\backslash A$). A category such as vp for verbs is not needed as a basic category because verbs can be defined in relation to their arguments. In particular, tensed intransitive verbs are characterised as compound categories of type $np\backslash s$. The type specifies that the verb is incomplete and needs an expression of category np on its left to form an expression of category s.

Complex expressions are built from their subpart using a deductive reasoning system. The goal is to proof that a complex expression belongs to a certain category. In this paper, we use the sequent-style presentation originally due to Gentzen to present derivations. An expression Γ of category A is represented as $\Gamma \vdash A$. The proof for a certain expression consists of an deductive analysis over the different types of formulas. Each operator comes with a set of introduction and elimination rules ($[\backslash E], [\backslash I], [/E], [/I]$). The derivation of a complex expression is a relation between a structure and a formula.

Structures are built out of elementary structures, formulas, that are built with structure building operations. In this paper the structure building operator is restricted to the binary operator $(\cdot \circ \cdot)$ which combines two substructures and preserves linear order and dominance with respect to the subformulas. In the structures, instead of writing formulas, we write the headword that belongs to a certain category (cf. sleeps $\vdash np\backslash s$). To save space, we will display the lexical insertion, the axioms, as follows:

$$\frac{\textsf{sleeps}}{np\backslash s}$$

For a more elaborate introduction in the proof-theoretical aspects of type-logical grammar, we refer the reader to Vermaat (2006).

3 Semantics of questions

Many theories that account for the semantics of questions relate the meaning of a question to its possible answers (for an overview, see Groenendijk and Stokhof (1997). Two approaches of relating questions and answers are the *proposition set approach* (Hamblin, 1958; Karttunen, 1977) in which questions represent propositions; and the approach which Krifka (2001) named the *structured meaning approach*, also referred to as the *functional* or *categorial approach* (Groenendijk and Stokhof, 1984). In this latter approach, the interrogative in combination with its answer forms a statement.

The proposition set approach (Hamblin, 1958) influenced the logical approach to the semantics of questions (Karttunen, 1977; Groenendijk and Stokhof, 1984). Hamblin (1958) stated that to determine the meaning of an interrogative one has to inspect what kind of statement can serve as a response:"an answer to a question is a sentence, or statement". The theory implements the idea that the semantic status of an answer is a proposition and that the syntactic form of an answer is irrelevant.

The structured meaning approach is sometimes referred to as the *functional* or *categorial* approach. The approach is developed by logicians and semanticists and supports the idea that the meaning of a question is dependent on the meaning of the answer and vice versa. Along similar lines, Hiż (1978) points out that questions and their answers are not autonomous sentences, but that they form a semantic unit — a question-answer pair. We briefly discuss the structured meaning approach.

Structured meaning approach An appropriate answer to a single constituent question may be any

type of syntactic object. This might be a generalized quantifier phrase or a verb phrase, as well as a noun phrase or prepositional phrase. Additionally, in multiple wh-questions, different combinations of syntactic objects can be used as an answer. The wh-question directs the kind of answers that can be expected.

(1) a. '*Who saw Mary?*' *John, nobody, John's sister, ...*

b. '*Which man did John see?*' *His father, the neighbor, ...*

c. '*Who saw whom?*'
pair list reading: *John (saw) Bill, Mary (saw) Sue, ...*
functional reading: *every professor/his student, John/his sister*

As the sentences illustrate, the answers have a direct relation to the interrogative phrase in the question. To capture the relation between the question and its possible answer type, the structured meaning approach formulates the idea that the question and answer form a unit, both syntactically and semantically. Syntactically, the interrogative in combination with its answer forms an indicative sentence or a question-answer sequence. This syntactic unit is reflected in the semantics where the question meaning is a function that yields a proposition when applied to the meaning of answer (Krifka, 2001).

Within the type-logical grammar framework, a functional view on question and answer types comes quite naturally, as shown in work of Hausser (1983) and more recently in Bernardi and Moot (2003). We will follow the structured meaning approach and show that the diversity in answer types can be derived from uniformly typed wh-phrases.

4 Question and answer types

In a structured meaning approach questions are expected to be functions that, when applied to an answer, yield a proposition. In this section, we spell out wh-questions as types that reflect the functor-argument relation between a wh-question and its response. In section 4.1 and 4.2, we show how this relation is captured in the syntactic type definition of wh-questions and wh-phrases. In section 4.3, we determine the effects on the meaning assembly of wh-questions.

4.1 Type definition of wh-questions

Adopting a structured meaning approach of questions, we incorporate the type of possible answers into the type of the wh-question. Generalizing over the possible types of answers and questions, we decompose wh-questions into the following type:

syntactic category		**semantic type**
$\overline{B/_? A}$	$=$	$\overline{A} \rightarrow \overline{B}$

The semantic type $\overline{A} \rightarrow \overline{B}$ is a direct mapping from the components of the syntactic category $B/_? A$. $\overline{A}$ is the semantic type of category A which is the type of the expected answer. $\overline{B}$ is the semantic type of category B which is the type of the question-answer sequence.

Notice that the type connective has an additional index $?$. We use this index to capture a compositional difference between predicates and arguments on a sentential level (structural composition relation: $\circ$) and between questions and answers on a dialogue level (structural composition relation: $\circ_?$). Following the structured meaning approach, we assume question-answer sequences to form a syntactic and semantic unit. Syntactically, we assume the question-answer sequence to belong to category s. Semantically, the question-answer sentence is a proposition which has a certain truth value, similar to declarative clauses. Before we look at how this question type determines the meaning of wh-questions, we need to know how wh-phrases are categorised.

4.2 Wh-type schema

We use an abbreviated type schema, a three-place operator, to lexically identify wh-elements. The selectional requirements of wh-phrases are encoded into this operator type schema and result in an uniform interpretation of wh-questions[2]. The type schema can be decomposed into the usual type-connectives of the base logic (Moortgat, 1997; Vermaat, 2006).

We adopt the q-type schema which was proposed by Moortgat (1991) to account for in-situ binding of generalized quantifier phrases. We propose a three-place type schema, WH, ranging over three subtypes: $\text{WH}(A, B, C)$. The three variables

[2] In Vermaat (2006), we recognise three structural variants of the wh-type schema that account for cross-linguistics variation in the word-order of wh-questions.

indicate the categories of substructures where a wh-phrase acts on. B is the category of the body of the wh-question; A is the category of the expression that the wh-phrase represents; C is the type of the result of merging the body of the wh-question with the wh-phrase. Variable A is the category of the 'gap' in the question body, which in this framework is introduced as an hypothesis, and occupies a structural position relative to the predicate.

The following inference rule defines the merging an arbitrary wh-phrase (= Γ) and a question body which contains an hypothesis of category A (= $\Delta[A]$).[3] The result of merging the wh-phrase and the body is a structure $\Delta[\Gamma]$ in which the wh-phrase replaces the gap hypothesis.

$$\dfrac{\begin{array}{c}\Delta[A] \vdash B \\ \vdots\ \Gamma \vdash \textsc{wh}(A, B, C)\end{array}}{\Delta[\Gamma] \vdash C}$$

Example We analyze the direct question '*Who saw Bill?*'. The wh-phrase '*who*' is categorised as the wh-type schema, $\textsc{wh}(np, s, s/_?(s/(np\backslash s)))$. When the wh-phrase is applied to its question body it yields a wh-question of category $s/_?(s/(np\backslash s))$, a sentence which is incomplete for a generalized quantifier. For ease of exposition we abbreviate $s/(np\backslash s)$ to gq. The reason for choosing this type for '*who*' is that the answer could be a np typed phrase as well as generalized quantifier phrase (section 6).

The following derivation shows the analysis of the wh-question in a natural deduction style with the abbreviated inference rule for merging the wh-phrase.

$$\dfrac{\dfrac{[np \vdash np]^1 \quad \dfrac{\dfrac{\textsf{saw}}{(np\backslash s)/np} \quad \dfrac{\textsf{bill}}{np}}{\textsf{saw} \circ \textsf{bill} \vdash np\backslash s}\,[/E]}{np \circ (\textsf{saw} \circ \textsf{bill}) \vdash s}\,[\backslash E]}{\begin{array}{l}\vdots\ \textsf{who} \vdash \textsc{wh}(np, s, s/_? gq)_{,1} \\ \textsf{who} \circ (\textsf{saw} \circ \textsf{bill}) \vdash s/_? gq\end{array}}$$

The main clause is built as usual, only the subject argument phrase is a hypothesised np argument instead of an actual noun phrase. After the body of the clause s is derived, the wh-phrase merges with the question body and replaces the np hypothesis, yielding a clause of type $s/_? gq$.

[3] $\Gamma[\Delta]$ is the representation of a structure Γ, a sequence of formulas which contains a substructure Δ.

4.3 Meaning assembly of wh-questions

To get a good understanding of the meaning representation of a wh-question, it's good to be aware of the type construction in the semantic type language. The semantic type that corresponds to the wh-type schema takes the corresponding semantic types of each subtype in the type schema and arranges them. Wh-type schema $\textsc{wh}(A, B, C)$ maps to the following semantic type:

$$(\overline{A} \rightarrow^{(2)} \overline{B}) \rightarrow^{(1)} \overline{C}$$

The semantic type reveals the inherent steps encoded in the rule schema. $\rightarrow^{(1)}$ is the application step, merging a wh-phrase with the body. $\rightarrow^{(2)}$ represents abstraction of the hypothesis, withdrawing the gap from the body of the wh-question.

Following the Curry-Howard correspondence each syntactic type formula is mapped to a corresponding semantic type. In turn, we interpret each expression by providing a semantic term that matches the semantic type. The semantic term assigned to wh-type schema $\textsc{wh}(A, B, C)$ is term operator ω which corresponds to the above semantic type. After merging the wh-phrase and the question body, the syntactic derivation yields the following semantic term for wh-questions:

$$(\omega\ \lambda x^{\overline{A}}.\textsc{body}^{\overline{B}})^{\overline{C}}$$

In this term, BODY is the term computed for the body of the wh-question which contains the hypothesis A associated with term variable x. Applying the ω-operator to the lambda abstraction of x over the term of the question body yields a term of the expected semantic type, $\overline{C}$.

Example We present the last step in the derivation of the wh-question '*Who saw Bill?*' illustrating the the semantic composition of the wh-phrase with the question body.

$$\dfrac{\begin{array}{c}x : np \circ (\textsf{saw} \circ \textsf{bill}) \vdash ((\textbf{see}\ \textbf{b})\ x) : s \\ \vdots\ \textsf{who} \vdash \omega : \textsc{wh}(np, s, s/_? gq)\end{array}}{\textsf{who} \circ (\textsf{saw} \circ \textsf{bill}) \vdash (\omega\ \lambda x.((\textbf{see}\ \textbf{b})\ x)) : s/_? gq}$$

The precise meaning representation of a wh-question depends, however, on the kind of wh-phrase that constitutes a wh-question. We argue that, at least for argument wh-phrases, different wh-type schema can be derived from a single wh-type schema. The basic case for wh-phrases is a wh-type schema that ranges over higher-order

typed answers: $\text{WH}(np, s, s/_?gq)$. The ω-operator that captures the meaning assembly of this wh-type schema can be regarded as a logical constant. The definition of the ω-operator generalises over different types of wh-phrases:

$$\omega \quad = \quad \lambda P^{\overline{A}\to\overline{B}}.\lambda Q^{(\overline{A}\to\overline{B})\to\overline{B}}.(Q\ P)$$

Example The meaning assembly for the wh-question '*Who saw Bill?*' is derived from the syntactic analysis of the sentence. The syntactic category and the lexical meaning assembly of the wh-phrase '*who*' is:

$$\text{who} \vdash \lambda P^{(et)}.\lambda Q^{(et)t}.(Q\ P) : \text{WH}(np, s, s/_?gq)$$

The semantic term assignment to '*who*' derives the right meaning assembly for a wh-question '*Who saw Bill?*'.

$$\text{Who saw Bill?} \vdash \lambda Q.(Q\ \lambda x.((\textbf{see m})\ x)) : s/_?gq$$

On the basis of this type-assignments for wh-phrases, we can derive different instances of the wh-type schema using axioms in the semantic type language (Moortgat, 1997).

5 Derivability patterns

Incorporating the answer type into the wh-type schema enables us to derive different instances of wh-type schema. On the basis of this derivability pattern, we can account for answer restrictions of certain wh-phrases and for the derivation of multiple wh-questions in section 6.

5.1 Semantic derivability

The derivability pattern of wh-type schema is based on three theorems that are derivable in semantic type language: *type-lifting*, *geach* and *exchange*. We illustrate each rule in semantic type language and present the meaning assembly for each type-shifting rule.

$$\begin{array}{lrcl}
[\textit{type-lifting}] & A & \vdash & (A \to B) \to B \\
 & x & \mapsto & \lambda y.(y\ x) \\
[\textit{geach}] \quad B \to A & & \vdash & (C \to B) \to (C \to A) \\
 & x & \mapsto & \lambda y.\lambda z.(x\ (y\ z))
\end{array}$$

$$\begin{array}{lrcl}
[\textit{exchange}] & C \to (D \to E) & \vdash & D \to (C \to E) \\
 & x & \mapsto & \lambda z.\lambda y.((x\ y)\ z)
\end{array}$$

Using these theorems, we can derive two additional laws *argument lowering* and *dependent geach*.

argument lowering The type-lifting rule lifts any arbitrary type A to a type $(A \to B) \to B$. The type lifting may alter the answer type to fit the answer type requested by the wh-question. From the type-lifting rule, we can also derive the rule for *argument lowering* which encodes the alternation of the answer type in the wh-type schema.

$$\begin{array}{lrcl}
((A \to B) \to B) \to C & \vdash & A \to C \\
x & \mapsto & \lambda y.(x\ \lambda z.(z\ y))
\end{array}$$

dependent geach The geach rule adds an additional dependent to both the main clause type A and its argument type B. Again, each type may be a complex type. The exchange rule captures the reordering of two dependents. If the geach rule is applied to a complex type $(D \to E) \to (B \to A)$, the result type is the complex type $(C \to (D \to E)) \to (C \to (B \to A))$. Additionally, we apply exchange to the consequent and the antecedent of the geach type and shift the order of the dependent types. We obtain a type-shifting rule which we refer to as *dependent geach* by combining the two rules.

$$\begin{array}{l}
(D \to E) \to (B \to A) \vdash \\
\qquad (D \to (C \to E)) \to (B \to (C \to A)) \\
x \quad \mapsto \quad \lambda z.\lambda y.\lambda v.((x\ \lambda u.((z\ u)\ v))\ y)
\end{array}$$

The theorems in the semantic type language reveal that under certain assumptions a number of type alternations are also derivable in the syntactic formula language. In Vermaat (2006), we show that argument lowering and dependent geach are derivable in the grammatical reasoning system. Applying the two rules to different instances of wh-type schema gives us derivability patterns between instances of wh-type schema. In figure 1, the syntactic derivability pattern of wh-type schemata is presented abstractly[4]. The syntactic pattern maps to the meaning assembly pattern as presented in figure 2.

6 Linguistic application

The syntactic decomposition of wh-question types into types that are part of an question-answer sequence adds polymorphism to the wh-type schemata. The semantic representation of wh-questions reflects the question's requirement for

[4]For the actual syntactic derivation, we need to reason structurally over unary operators $\Diamond$ and $\Box$, see Vermaat (2006).

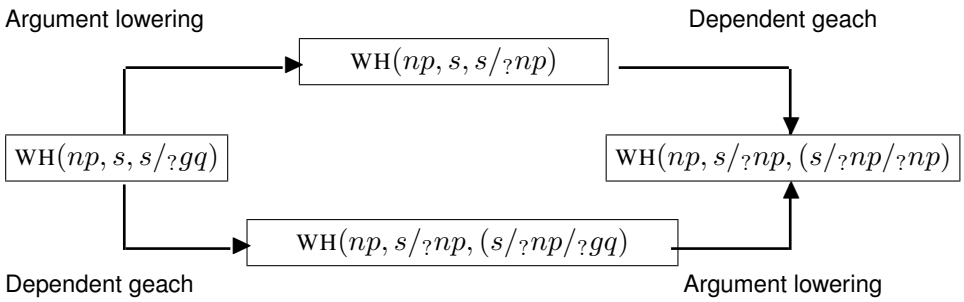

Figure 1: Syntactic derivability pattern of wh-type schemata

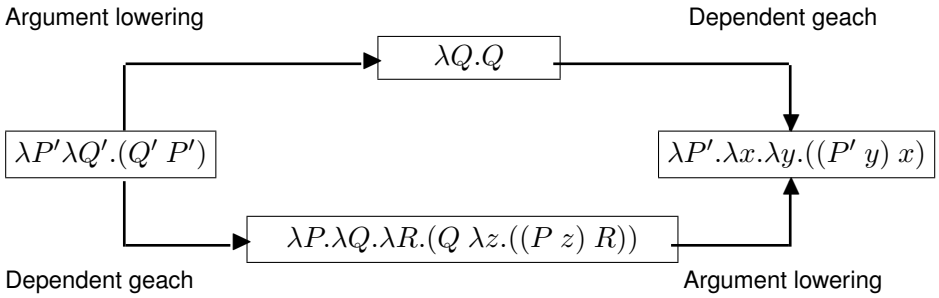

Figure 2: Meaning assembly of derivability patterns

certain types of answers. In this section, we explore the linguistic application of the derivability pattern for wh-question formation.

In section 6.1, we focus on the derivation of single constituent questions in English. We discuss the syntactic and semantic consequences of argument lowering for the derivation of question-answer sequences in local wh-questions. In section 6.2, we discuss multiple wh-questions in English. We show that we can account for the derivation of multiple wh-questions on the basis of deriving geach types for both ex-situ and in-situ type schema. And as a result derive the correct meaning assembly for multiple wh-questions.

6.1 Single constituent questions

A single constituent question requires a single constituent answer. We concentrate here on argument wh-phrases to illustrate the relation between a wh-question and possible answers. We will look at direct questions where the associated gap hypothesis appears in the local domain.

In direct questions in English a fronted wh-phrase associates with a np gap hypothesis. The expected answer, however, depends on the kind of wh-phrase. Wh-questions with argument wh-phrases such as '*what*' or '*who*' expect either a referential or a quantified noun phrase. Wh-questions with *which*-determiners only expect a referential noun phrase as an answer. On the basis of the derivability pattern of wh-ex-situ types we can account for the distinction between the two types of wh-phrases. First, we discuss the lexical type-assignments of wh-pronouns. Then, we present the contrast with wh-determiners.

Wh-pronouns A suitable answer to a wh-question such as '*Who saw Bill?*' might be a referential noun phrase e.g. '*John*', as well as a generalized quantifier phrase e.g. '*everyone*'. To allow both types of answers, '*who*' and '*whom*' are assigned the following wh-type schema in the lexicon.

$$\text{who(m)} \vdash \lambda P^{et}.\lambda Q^{(et)t}.(Q\ P)$$
$$\text{WH}(np, s, s/_?(s/(np\backslash s)))$$

The sentence in 2 is an example of the different kinds of question-answer sequences that can be derived using the given type-assignments for wh-pronouns. The type that is derived for subject wh-questions is a s-typed clause which is incomplete for a lifted np type, $(s/(np\backslash s))$. A generalized quantifier phrase can be merged directly, while referential noun phrases such as '*John*' in example 2b have to be lifted before they can be merged. Along with the syntactic category, lifting alters the semantic type of the answer in such a way that the lifted type matches the semantic type requested

by the interrogative clause. The semantic term is computed as usual. The same line of reasoning applies to the derivation of question-answer pairs with non-subject argument wh-questions.

(2) Who saw Mary? $\vdash$
$\lambda Q^{(et)t}.(Q\ \lambda x.((\textbf{see m})\ x)) : s/_?(s/(np\backslash s)$

a. Answer: '*every man*' $\vdash gq$
$\forall y((\textbf{man}\ y) \rightarrow ((\textbf{see m})\ y))$

b. Answer: '*John*' $\vdash np$
$(\lambda P.(P\ \textbf{j})\ \lambda x.((\textbf{see m})\ x))$
$\rightsquigarrow_\beta (\lambda x.((\textbf{see m})\ x)\ \textbf{j})$
$\rightsquigarrow_\beta ((\textbf{see m})\ \textbf{j})$

Wh-determiners Suitable answers to wh-questions that are built with wh-determiners like '*which*' are restricted to definite noun phrases. The semantic difference between wh-phrases and wh-determiners lies in the specific denotation of the *which*-phrases. For instance, the wh-question '*Which man saw Mary?*' can be paraphrased as '*Who is **the** man that saw Mary?*'. The person who utters the question and the hearer already have the background knowledge that *the person who saw Mary* is a man. A definite answer is the most likely response. This gives us evidence to assume that a wh-determiner has a minimal type-assignments that derives a question of type: $s/_?np$. On the basis of this assumption, wh-determiners belong to a wh-type that yields a question of type $s/_?np$. The semantic term that matches this type reveals the definiteness of the answer that is requested.

which $\vdash \text{WH}(np, s, s/_?np)/n$
$\lambda V.\lambda P.\lambda x.(x = \iota y.((V\ y) \wedge (P\ y)))$

On the basis of this type-assignments we can derive the following question-answer sequence in example 3a, while the answer in 3b is underivable.

(3) Which man saw Mary? $\vdash s/_?np$
$\lambda x.(x = \iota y.(\textbf{man}\ y) \wedge ((\textbf{see m})\ y))$

a. Answer: '*John*' $\vdash np$
$\textbf{j} = \iota y.((\textbf{man}\ y) \wedge ((\textbf{see m})\ y))$

b. Which man saw Mary? $\vdash s/_?np$
Answer: $*$ '*every man*' $\vdash gq$

6.2 Multiple wh-questions

With the derivability pattern of wh-type schema using dependent Geach, as presented in section 5, we can derive multiple wh-questions from a single type-assignments to a wh-phrase in the lexicon. Multiple wh-questions in English are recognised by a single wh-phrase that appears at the front of the main clause, whereas additional wh-phrases appear embedded. In Vermaat (2006), we have explored the syntax of multiple wh-phrases. Wh-phrases that occur in-situ are lexically categorised as:

wh-in-situ $\vdash \text{WH}_{in}(np, s/_?np, (s/_?np)/_?np)$

This type encodes that the wh-phrase may only appear in-situ in a wh-question body of type $s/_?np$, i.e. a sentence which already contains a wh-phrase. The wh-type schema encodes that a wh-phrase merges with a question body of type $s/_?np$, which contains a gap hypothesis of type np. Notice that the wh-in-situ type schema can be derived from $\text{WH}_{in}(np, s, s/_?gq)$ using argument lowering and dependent geach. By assigning wh-in-situ phrases the above type, we correctly derive that '*whom*' can never occur in-situ in a phrase that does not have a fronted wh-phrase. With this minimal type-assignments the wh-in-situ phrase is always dependent on the occurrence of another wh-phrase $(s/_?gq)$. This dependency is reflected in both syntax and semantics.

Syntactically, the wh-in-situ phrase is dependent on the occurrence of the subject wh-phrase. Semantically, the lambda abstraction binds the type of the subject wh-phrase over the object wh-phrase.

ex-situ who $\vdash \lambda R.\lambda Q.(Q\ R) : \text{WH}(np, s, s/_?gq)$
in-situ whom $\vdash \lambda P.\lambda x.\lambda y.((P\ y)\ x) :$
$\text{WH}_{in}(np, s/_?np, (s/_?np)/_?np)$

On the basis of this type-assignments and the usual wh-type schema assigned to the subject wh-phrase, we derive the multiple wh-question '*Who saw whom*' in Fig. 3. In the derivation the inference steps are represented as structure $\vdash$ *type* whereas the meaning assembly is written below the sequent.

7 Conclusion and future research

In this paper, we have discussed the syntactic and semantic consequences of a structured meaning approach to wh-questions. In a structured meaning approach, wh-questions are taken to be incomplete sentences that are part of a question-answer sequence. We have proposed to decompose wh-questions into a type $A/_?B$ where A is the type

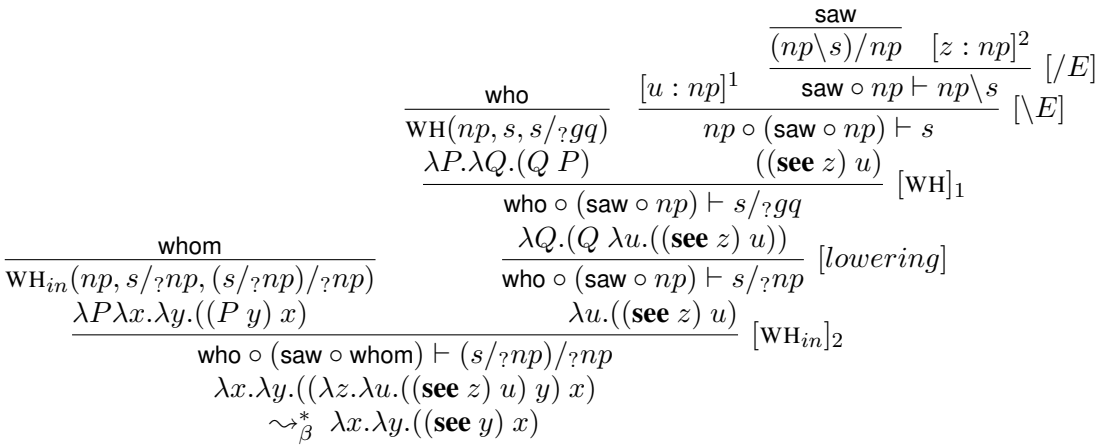

Figure 3: Derivation of multiple wh-question

of the question-answer sequence and B is the type of the answer. With the syntactic decomposition of wh-types, we have been able to express the semantic decomposition of the semantic ω-operator as a λ-term.

Additionally, the syntactic and semantic decomposition of the type for wh-questions leads to a derivability pattern of wh-type schemata. This pattern provides generalizations for different question answer sequences. For instance, the difference between wh-pronouns and wh-determiners and the derivation of multiple wh-questions. The presented sentences have been computed using the on-line parser for type-logical grammars. See http://grail.let.uu.nl/~vermaat for further analyses of this specific grammar fragment and that of other languages.

The theoretical results in this paper have been limited to argument wh-phrases. Next step is to see how the derivability schema and the wh-type schema apply to other types of wh-phrases, such as adverbial wh-phrases. Additionally, we would like to investigate additional logical axioms that may lead to further generalizations for natural language analysis. For a practical purpose, it would be interesting to see whether the theoretical issues addressed in this paper could be used in existing question-answer dialogue systems, for example to validate the answer.

Acknowledgments

The Netherlands Organisation for Scientific Research (NWO) for providing the Rubicon grant.

References

Raffaella Bernardi and Richard Moot. 2003. Generalized quantifiers in declarative and interrogative sentences. *Logic Journal of the IGPL*, 11(4):419–34.

Jeroen Groenendijk and Martin Stokhof. 1984. *Studies on the semantics of questions and the pragmatics of answers*. Ph.D. thesis, University of Amsterdam.

Jeroen Groenendijk and Martin Stokhof. 1997. Questions. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 19, pages 1055–1124. Elsevier, Amsterdam.

C.J. Hamblin. 1958. Questions. *Australasian Journal of Philosophy*, 36:159–68.

Roland Hausser. 1983. The syntax and semantics of english mood. In Ferenc Kiefer, editor, *Questions and Answers*, pages 97–158. Reidel, Dordrecht.

Henry Hiż. 1978. Difficult questions. In Henry Hiż, editor, *Questions*. Reidel, Dordrecht/Boston.

Lauri Karttunen. 1977. Syntax and semantics of questions. *Linguistics and Philosophy*, 1:3–44.

Manfred Krifka. 2001. For a structured meaning account of questions and answers. In C. Fery and W. Sternefeld, editors, *Audiatur Vox Sapientia. A Festschrift for Arnim von Stechow*, pages 287–319. Akademie Verlag, Berlin.

Michael Moortgat. 1991. Generalized quantifiers and discontinuous type constructors. In W. Sijtsma and A. van Horck, editors, *Discontinuous constituency*. De Gruyter.

Michael Moortgat. 1997. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2, pages 93–177. Elsevier, Amsterdam.

Glyn Morrill. 1994. *Type Logical Grammar. Categorial Logic of Signs*. Kluwer, Dordrecht.

Mark Steedman. 2000. *The syntactic process*. The MIT Press, Cambridge, MA.

Willemijn Vermaat. 2006. *The Logic of Variation: a Cross-linguistic account of Wh-question Formation in type-logical grammar*. Ph.D. thesis, Utrecht University, UiL-OTS, January.

Towards the Evaluation of Referring Expression Generation

Jette Viethen
Centre for Language Technology
Macquarie University
Sydney NSW 2109
jviethen@ics.mq.edu.au

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

Abstract

The Natural Language Generation community is currently engaged in discussion as to whether and how to introduce one or several shared evaluation tasks, as are found in other fields of Natural Language Processing. As one of the most well-defined subtasks in NLG, the generation of referring expressions looks like a strong candidate for piloting such shared tasks. Based on our earlier evaluation of a number of existing algorithms for the generation of referring expressions, we explore in this paper some problems that arise in designing an evaluation task in this field, and try to identify general considerations that need to be met in evaluating generation subtasks.

1 Introduction

In recent years, the inclusion of an evaluation component has become almost obligatory in any publication in the field of Natural Language Processing. For complete systems, user-based and task-oriented evaluation are almost standard practice in both the Natural Language Understanding (NLU) and Natural Language Generation (NLG) communities. A third, more competitive, form of evaluation has become increasingly popular in NLU in the form of shared-task evaluation campaigns (STECs). In a STEC, different approaches to a well-defined problem are compared based on their performance on the same task. A large number of different research communities within NLP, such as Question Answering, Machine Translation, Document Summarisation, Word Sense Disambiguation, and Information Retrieval, have adopted a shared evaluation metric and in many cases a shared-task evaluation competition.

The NLG community has so far withstood this trend towards a joint evaluation metric and a competitive evaluation task, but the idea has surfaced in a number of discussions, and most intensely at the 2006 International Natural Language Generation Conference (see, for example, Bangalore et al. (2000), Reiter and Sripada (2002), Reiter and Belz (2006), Belz and Reiter (2006), Belz and Kilgarriff (2006), Paris et al. (2006), and van Deemter et al. (2006)).

Amongst the various component tasks that make up Natural Language Generation, the generation of referring expressions is probably the subtask for which there is the most agreement on problem definition; a significant body of work now exists in the development of algorithms for generating referring expressions, with almost all published contributions agreeing on the general characterisation of the task and what constitutes a solution. This suggests that, if formal shared tasks for NLG are to be developed, the generation of referring expressions is a very strong candidate.

In (Viethen and Dale, 2006), we argued that the evaluation of referring expression generation algorithms against natural, human-generated data is of fundamental importance in assessing their usefulness for the generation of understandable, natural-sounding referring expressions. In this paper, we discuss a number of issues that arise from the evaluation carried out in (Viethen and Dale, 2006), and consider what these issues mean for any attempt to define a shared task in this area.

The remainder of this paper has the following structure. In Section 2, we briefly describe the evaluation experiment we carried out for three well-established referring expression generation

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 115–122.

algorithms, and report the performance of these algorithms in the chosen test domain. This leads us to identify three specific issues that arise for the evaluation of referring expression generation algorithms, and for NLG systems in general; we discuss these in the subsequent sections of the paper. Section 3 looks at the problem of input representations; Section 4 explores how the wide variety of acceptable outputs, and the lack of a single correct answer, makes it hard to assess generation algorithms; and Section 5 explores whether we can usefully provide a numeric measure of the performance of a generation algorithm. Finally, in Section 6 we point to some ways forward.

2 An Evaluation Experiment

In (Viethen and Dale, 2006), we observed that surprisingly little existing work in natural language generation compares its output with natural language generated by humans, and argued that such a comparison is essential. To this end, we carried out an experiment consisting of three steps:

1. the collection of natural referring expressions for objects in a controlled domain, and the subsequent analysis of the data obtained;
2. the implementation of a knowledge base corresponding to the domain, and the re-implementation of three existing algorithms to operate in that domain; and
3. a detailed assessment of the algorithms' performance against the set of human-produced referring expressions.

In the remainder of this section we briefly describe these three stages. As we are mainly concerned here with the evaluation process, we refer to (Viethen and Dale, 2006) for a more detailed account of the experimental settings and an in-depth discussion of the results for the individual algorithms.

2.1 The Human-Generated Data

Our test domain consists of four filing cabinets, each containing four vertically arranged drawers. The cabinets are placed directly next to each other, so that the drawers form a four-by-four grid as shown in Figure 1. Each drawer is labelled with a number between 1 and 16 and is coloured either blue, pink, yellow, or orange. There are four drawers of each colour distributed randomly over the grid.

1 (blue)	2 (orange)	3 (pink)	4 (yellow)
8 (blue)	7 (blue)	6 (yellow)	5 (pink)
9 (orange)	10 (blue)	11 (yellow)	12 (orange)
16 (yellow)	15 (pink)	14 (orange)	13 (pink)

Figure 1: The filing cabinets

The human participants were given, on a number of temporally-separated occasions, a random number between 1 and 16, and then asked to provide a description of the corresponding drawer to an onlooker without using any of the numbers; this basically restricted the subjects to using either colour, location, or some combination of both to identify the intended referent. The characterisation of the task as one that required the onlooker to identify the drawer in question meant that the referring expressions produced had to be *distinguishing descriptions*; that is, each referring expression had to uniquely refer to the intended referent, but not to any of the other objects in the domain.

The set of natural data we obtained from this experiment contains 140 descriptions. We filtered out 22 descriptions that were (presumably unintentionally) ambiguous or used reference to sets of drawers rather than only single drawers. As none of the algorithms we wanted to test aims to produce ambiguous referring expressions or handle sets of objects, it is clear that they would not be able to replicate these 22 descriptions. Thus the final set of descriptions used for the evaluation contained 118 distinct referring expressions.

Referring expression generation algorithms typically are only concerned with selecting the semantic content for a description, leaving the details of syntactic realisation to a later stage in the language production process. We are therefore only interested in the semantic differences between the descriptions in our set of natural data, and not in superficial syntactic variations. The

primary semantic characteristics of a referring expression are the properties of the referent used to describe it. So, for example, the following two referring expressions for drawer d3 are semantically different:

(1) The pink drawer in the first row, third column.

(2) The pink drawer in the top.

For us these are distinct referring expressions. We consider syntactic variation, on the other hand, to be spurious; so, for example, the following two expressions, which demonstrate the distinction between using a relative clause and a reduced relative, are assumed to be semantically identical:

(3) The drawer that is in the bottom right.

(4) The drawer in the bottom right.

We normalised the human-produced data to remove syntactic surface variations such as these, and also to normalise synonymic variation, as demonstrated by the use of the terms *column* and *cabinet*, which in our context carry no difference in meaning.

The resulting set of data effectively characterises each human-generated referring expression in terms of the semantic attributes used in constructing those expressions. We can identify four absolute properties that the human participants used for describing the drawers: these are the colour of the drawer; its row and column; and in those cases where the drawer is located in one of the corners of the grid, what we might call cornerhood. A number of participants also made use of relations that hold between two or more drawers to describe the target drawer. The relational properties that occurred in the natural descriptions were: above, below, next to, right of, left of and between. However, relational properties were used a lot less than the other properties: 103 of the 118 descriptions (87.3%) did not use relations between drawers.

Many referring expression generation algorithms aim to produce minimal, non-redundant descriptions. For a referring expression to be minimal means that all of the facts about the referent that are contained in the expression are essential for the hearer to be able to uniquely distinguish the referent from the other objects in the domain. If any part of the referring expression was dropped, the description would become ambiguous; if any other information was added, the resulting expression would contain redundancy.

Dale and Reiter (1995), in justifying the fact that their Incremental Algorithm would sometimes produce non-minimal descriptions, pointed out that human-produced descriptions are often not minimal in this sense. This observation has been supported more recently by a number of other researchers in the area, notably van Deemter and Halldórsson (2001) and Arts (2004). However, in the data from our experiment it is evident that the participants tended to produce minimal descriptions: only 24.6% of the descriptions (29 out of 118) contain redundant information.

2.2 The Algorithms

Many detailed descriptions of algorithms are available in the literature on the generation of referring expressions. For the purpose of our evaluation experiment, we focussed here on three algorithms on which many subsequently developed algorithms have been based:

- The Full Brevity algorithm (Dale, 1989) uses a greedy heuristic for its attempt to build a minimal distinguishing description. At each step, it always selects the most discriminatory property available.

- The Relational Algorithm from (Dale and Haddock, 1991) uses constraint satisfaction to incorporate relational properties into the framework of the Full Brevity algorithm. It uses a simple mechanism to avoid infinite regress.

- The Incremental Algorithm (Reiter and Dale, 1992; Dale and Reiter, 1995) considers the available properties to be used in a description via a predefined preference ordering over those properties.

We re-implemented these algorithms and applied them to a knowledge base made up of the properties evidenced collectively in the human-generated data. We then analysed to which extent the output of the algorithms for each drawer was semantically equivalent to the descriptions produced by the human participants. The following section gives a short account of this analysis.

2.3 Coverage of the Human Data

Out of the 103 natural descriptions that do not use relational properties, the Full Brevity Algorithm is able to generate 82 by means of at least one preference ordering over the object properties, providing a recall of 79.6%. The recall achieved by the Incremental Algorithm is 95.1%: it generates 98 of the 103 descriptions under at least one preference ordering. The relational descriptions from the natural data are not taken into account in evaluating the performance of these two algorithms, since they are not designed to make use of relational properties.

Both the Full Brevity Algorithm and the Incremental Algorithm are able to replicate all the minimal descriptions found in the natural data. Against its specification to avoid all redundancy, the Full Brevity Algorithm also generates nine of the redundant descriptions; the Incremental Algorithm replicates 24 of the 29 redundant descriptions produced by humans.

Perhaps surprisingly, the Relational Algorithm does not generate *any* of the human-produced descriptions. The particular strategy adopted by this algorithm is quite at odds with the human-generated descriptions in our data; we refer the reader to (Viethen and Dale, 2006) for a discussion of this failure, since it does not have a direct bearing on the present topic.

We now go on to discuss some of the key issues for NLG evaluation that became evident in this experiment.

3 Deciding on Input Representations

3.1 A Key Problem in NLG

It is widely accepted that the input for NLG systems is not as well-defined as it is in NLU tasks. In NLU the input will always be natural language, which is processed according to the task and transformed into *a machine-usable format of some kind*. In NLG, on the other hand, we are working in the other direction: there exists no consensus of what exact form the input into the system should take. The input is a knowledge base in *a machine-usable format of some kind*, whereas it is the desired format of the output—natural language—that is clear. As Yorick Wilks is credited with observing, Natural Language Understanding is like counting from 1 to infinity, but Natural Language Generation is like the much more perplexing task of counting from infinity to 1. The problem of determining what the generation process starts from is probably one of the major reasons for the lack of shared tasks in the field: each researcher chooses a level of representation, and a population of that level of representation, that is appropriate to exploring the kinds of distinctions that are central to the research questions they are interested in.

3.2 A Problem for Referring Expression Generation

As alluded to earlier, the generation of referring expressions seems to avoid this problem. The task is generally conceived as one where the intended referent, and its distractors in the domain, are represented by symbolic identifiers, each of which is characterised in terms of a collection of attributes (such as colour and size) with their corresponding values (red, blue, small, large...).

However, this apparent agreement is, ultimately, illusory. A conception in terms of symbolic identifiers, attributes, and values provides only a schema; to properly be able to compare different algorithms, we still need to have agreement on the specific attributes that are represented, and the values these attributes can take.

As we employed a new domain for the purpose of our evaluation experiment, we had to first decide how to represent this domain. Some of our representational primitives might seem to be non-contentious: the choice of colour, row and column seem quite straightforward. However, we also explicitly represented a more controversial attribute position, which took the value corner for the four corner drawers. Although cornerhood can be inferred from the row and column information, we added this property explicitly because it seems plausible to us that it is particularly salient in its own right.

This raises the general question of what properties should be encoded explicitly, and which should be inferred. In our experiment, we explicitly encode relational properties that could be computed from each other, such as left-of and right-of. We also chose not to implement the transitivity of spatial relations. Due to the uniformity of our domain the implementation of transitive inference would result in the generation of unnatural descriptions, such as *the orange drawer (two) right of the blue drawer* for d_{12}. Since none of the algorithms explored in our experiment uses inference over knowledge base properties, we opted here

to enable a fairer comparison between human-produced and machine-produced descriptions and decided against any inferred properties.

The decisions we took regarding the representation of cornerhood, inferrable properties in general, and transitive properties, were clearly influenced by our knowledge of how the algorithms to be tested work. If we had only assessed different types of relational algorithms, we might have implemented corners, and possibly even columns and rows, as entities that drawers are spatially related to. If the assessed algorithms had been able to handle inferred properties, cornerhood might have been implemented only implicitly as a result of the grid information about a drawer. The point here is that our representational choices were guided by the requirements of the algorithms, and our intuitions about salience as derived from our examination of the data; other researchers might have made different choices.

3.3 Consequences

From the observations above, it is evident that, in any project that focusses on the generation of referring expressions, the design of the underlying knowledge base and that of the algorithms that use that knowledge base are tightly intertwined. If we are to define a shared evaluation task or metric in this context, we can approach this from the point of view of assessing only the algorithms themselves, or assessing algorithms in combination with their specific representations. In the first case, clearly the input representation should be agreed by all ahead of time; in the second case, each participant in the evaluation is free to choose whatever representation they consider most appropriate.

The latter course is, obviously, quite unsatisfactory: it is too easy to design the knowledge base in such a way as to ensure optimal performance of the corresponding algorithm. On the other hand, the former course is awash with difficulty: even in our very simple experimental domain, there are representational choices to be made for which there is no obvious guidance. We have discussed this problem in the context of what, as we have noted already, is considered to be a generation subtask on which there is considerable agreement; the problem is much worse for other component tasks in NLG.

4 Dealing with Determinism

4.1 There is More than One Way to Skin a Cat

One very simple observation from the natural data collected in our experiment is that people do not always describe the same object the same way. Not only do different people use different referring expressions for the same object, but the same person may use different expressions for the same object on different occasions. Although this may seem like a rather unsurprising observation, it has never, as far as we are aware, been taken into account in the development of any algorithm for the generation of referring expressions. Existing algorithms typically assume that there is a best or most-preferred referring expression for every object.

How might we account for this variation in the referring expressions that are produced by people? Where referring expressions are produced as part of natural dialogic conversation, there are a number of factors we might hypothesize would play a role: the speaker's perspective or stance towards the referent, the speaker's assumptions about the hearer's knowledge, the appropriate register, and what has been said previously. However, it is hard to see how these factors can play an important role in the simple experimental setup we used to generate the data discussed here: the entities are very simple, leaving little scope for notions of perspective or stance; and the expressions are constructed effectively *ab initio*, with no prior discourse to set up expectations, establish the hearer's knowledge, or support alignment. The sole purpose of the utterances is to distinguish the intended referent from its distractors.

We noted earlier that one regard in which multiple different descriptions of a referent may vary is that some may be redundant where others are not. Carletta (1992) distinguishes *risky* and *cautious* behaviour in the description task: while some participants would use only the briefest references, hoping that these would do the job, others would play safe by loading their descriptions with additional information that, in absolute terms, might make the overall description redundant, but which would make it easier or less confusing to interpret. It is possible that a similar or related speaker characteristic might account for some of the variation we see here; however, it would still not provide a basis for the variation even within the redundant

and minimal subsets of the data.

Of course, it can always be argued that there is no 'null context', and a more carefully controlled and managed experiment would be required to rule out a range of possible factors that predispose speakers to particular outcomes. For example, an analysis in terms of how the speakers 'come at' the referent before deciding how to describe it might be in order: if they find the referent by scanning from the left rather than the right (which might be influenced by the ambient lighting, amongst other things), are different descriptions produced? Data from eye-tracking experiments could provide some insights here. Or perhaps the variation is due to varying personal preferences at different times and across participants.

Ultimately, however, even if we end up simply attributing the variation to some random factor, we cannot avoid the fact that there is no single best description for an intended referent. This has a direct bearing on how we can evaluate the output of a specific algorithm that generates references.

4.2 Evaluating Deterministic Algorithms

The question arising from this observation is this: why should algorithms that aim to perform the task of uniquely describing the drawers in our domain have to commit to exactly one 'best' referring expression per drawer? In the context of evaluating these algorithms against human-generated referring expressions, this means that the algorithms start out with the disadvantage of only being able to enter one submission per referent into the competition, when there are a multitude of possible 'right' answers.

This issue of the inherent non-determinism of natural language significantly increases the degree of difficulty in evaluating referring expression algorithms, and other NLG systems, against natural data. Of course, this problem is not unique to NLG: recent evaluation exercises in both statistical machine translation and document summarisation have faced the problem of multiple gold standards (see Akiba et al. (2001) and Nenkova and Passonneau (2004), respectively). However, it is not obvious that such a fine-grained task as referring expression generation can similarly be evaluated by comparison against a gold standard set of correct answers, since even a large evaluation corpus of natural referring expressions can never be guaranteed to contain all acceptable descriptions for an object. Thus an algorithm might achieve an extremely low score, simply because the perfectly acceptable expressions it generates do not happen to appear in the evaluation set. Just because we have not yet seen a particular form of reference in the evaluation corpus does not mean that it is incorrect.

We might try to address this problem by encouraging researchers to develop non-deterministic algorithms that can generate many different acceptable referring expressions for each target object to increase the chances of producing one of the correct solutions. The evaluation metric would then have to take into account the number of referring expressions submitted per object. However, this would at most alleviate, but not entirely solve, the problem.

This poses a major challenge for attempts to evaluate referring expression generation algorithms, and many other NLG tasks as well: for such tasks, evaluating against a gold standard may not be the way to go, and some other form of comparative evaluation is required.

5 Measuring Performance

Related to the above discussion is the question of how we measure the performance of these systems even when we do have a gold standard corpus that contains the referring expressions generated by our algorithms. In Section 2.3, we noted that the Incremental Algorithm achieved a recall of 95.1% against our human-produced data set, which is to say that it was able to produce 95.1% of the descriptions that happened to appear in the data set; but as noted in the previous section, we cannot simply consider this data set to be a gold standard in the conventional sense, and so it is not really clear what this number means.

The problem of counting here is also impacted by the nature of the algorithm in question: as noted in Section 2.3, this performance represents the behaviour of the algorithm in question *under at least one preference ordering*.

The Incremental Algorithm explicitly encodes a preference ordering over the available properties, in an attempt to model what appear to be semi-conventionalised strategies for description that people use. The properties are considered in the order prescribed by the preference list and a particular property is used in the referring expression if it provides some discriminatory power, oth-

erwise it is skipped.

However, even within a single domain, one can of course vary the preference ordering to achieve different effects. It was by means of manipulation of the preference ordering that we were able to achieve such a high coverage of the human-produced data. We chose to view the manipulation of the preference ordering as the tweaking of a parameter. It could be argued that each distinct preference ordering corresponds to a different instantiation of the algorithm, and so reporting the aggregate performance of the collection of instantiations might be unfair. On the other hand, no single preference ordering would score particularly highly; but this is precisely because the human data represents the results of a range of different preference orderings, assuming that there is something analogous to the use of a preference ordering in the human-produced referring expressions. So it seems to us that the aggregated results of the best performing preference orderings provide the most appropriate number here.

Of course, such an approach would also likely produce a large collection of referring expressions that are not evidenced in the data. This might tempt us to compute precision and recall statistics, and assign such an algorithm some kind of F-score to measure the balance between under-generation and over-generation. However, this evaluation approach still suffers from the problem that we are not sure how comprehensive the gold standard data set is in the first place.

Ultimately, it seems that performance metrics based on the notion of coverage of a data set are fundamentally flawed when we consider a task like referring expression generation. We have argued above that asking the question 'Does the algorithm generate the correct reference?' does not make sense when there are multiple possible correct answers. The question 'Does the algorithm generate one of the correct answers?' on the other hand, is impracticable, because we don't have access to the full set of possible correct answers. Although it is not clear if a data-driven evaluation approach can fully achieve our purpose here, a better question would be: 'Does this algorithm generate a reference that a person would use?'

6 Conclusions

It is widely agreed that the requirement of numerical evaluation has benefitted the field of NLP by focussing energy on specific, well-defined problems, and has made it possible to compare competing approaches on a level playing field. In this paper, we have attempted to contribute to the debate as to how such an approach to evaluation might be brought into the field of NLG. We did this by exploring issues that arise in the evaluation of algorithms for the generation of referring expressions, since this is the area of NLG where there already seems to be something like a shared task definition.

By examining the results of our own experiments, where we have compared the outputs of existing algorithms in the literature with a collection of human-produced data, we have identified a number of key concerns that must be addressed by the community if we are to develop metrics for shared evaluation in the generation of referring expressions, and in NLG more generally.

First, it is essential that the inputs to the systems are agreed by all, particularly in regard to the nature and content of the representations used. This is a difficult issue, since NLG researchers have typically constructed their own representations that allow exploration of the research questions in their particular foci of interest; agreement on representations will not come easily. One could look to representations that exist for separately motivated tasks, thus providing an independent arbiter: for example, one might use tabular data corresponding to stock market results or meteorological phenomena. However, such representations considerably under-represent the content of texts that might describe them, leaving considerable scope for researchers to add their own special ingredients.

Second, we observe that there are many ways in which language can say the same thing or achieve the same result. Any attempt to assess the output of a language generation system has to contend with the fact that there are generally many correct answers to the problem, and there are no easy solutions to producing a reference set that contains all the possible answers. This suggests that an alternative paradigm might need to be developed for assessing the quality of NLG system output. Task-based evaluations (for example, testing if a user is able to complete a particular task given a machine-generated set of instructions) are an option to circumvent this problem, but are too coarse-grained to give us insights into the quality of the generated

output.

Finally, and related to the point above, it is not at all obvious that numeric measures like precision and recall make any sense in assessing generation system output. A generation system that replicates most or all of the outputs produced by humans, while overgenerating as little as possible, would clearly be highly adequate. However, we cannot automatically penalise systems for generating outputs that have not, so far, been seen in human-produced data.

Our analysis makes it seem likely that the impracticability of constructing a gold standard data set will prove itself as the core problem in designing tasks and metrics for the evaluation of systems for the generation of referring expressions and of NLG systems in general. There are various ways in which we might deal with this difficulty, which will need to be examined in turn. One possible way forward would be to take a more detailed look at the solutions that other tasks with output in the form of natural language, such as machine translation and text summarisation, have found for their evaluation approaches. We might also come to the conclusion that we can make do with a theoretically 'imperfect' evaluation task that works well enough to be able to assess any systems conceivably to be developed in the near or medium term.

Although we concede that a lot of groundwork still needs to be done, we are convinced that a more standardised evaluation approach is important for the advancement of the field of NLG.

References

Akiba, Y., Imamura, K., and Sumita, E. 2001. Using multiple edit distances to automatically rank machine translation output. In *Proceedings of the MT Summit VIII*, 15–20, Santiago de Compostela, Spain.

Arts, A. 2004. *Overspecification in Instructive Texts*. Ph.D. thesis, Tilburg University.

Bangalore, S., Rambow, O., and Whittaker, S. 2000. Evaluation metrics for generation. In *Proceedings of the First International Natural Language Generation Conference*, 1–8, Mitzpe Ramon, Israel.

Belz, A. and Kilgarriff, A. 2006. Shared-task evaluations in HLT: Lessons for NLG. In *Proceedings of the Fourth International Natural Language Generation Conference*, 133–135, Sydney, Australia.

Belz, A. and Reiter, E. 2006. Comparing automatic and human evaluation of NLG systems. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics*, 313–320, Trento, Italy.

Carletta, J. 1992. *Risk-taking and Recovery in Task-Oriented Dialogue*. Ph.D. thesis, University of Edinburgh.

Dale, R. and Haddock, N. 1991. Generating referring expressions involving relations. In *Proceedings of the Fifth Conference of the European Chapter of the ACL*, 161–166, Berlin, Germany.

Dale, R. and Reiter, E. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Dale, R. 1989. Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 68–75, Vancouver, British Columbia.

Nenkova, A. and Passonneau, R. 2004. Evaluating content selection in summarization: The pyramid method. In *Main Proceedings of HLT-NAACL 2004*, 145–152, Boston, Massachusetts, USA.

Paris, C., Colineau, N., and Wilkinson, R. 2006. Evaluations of nlg systems: Common corpus and tasks or common dimensions and metrics? In *Proceedings of the Fourth International Natural Language Generation Conference*, 127–129, Sydney, Australia. Association for Computational Linguistics.

Reiter, E. and Belz, A. 2006. Geneval: A proposal for shared-task evaluation in NLG. In *Proceedings of the Fourth International Natural Language Generation Conference*, 136–138, Sydney, Australia.

Reiter, E. and Dale, R. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics*, 232–238, Nantes, France.

Reiter, E. and Sripada, S. 2002. Should corpora texts be gold standards for NLG? In *Proceedings of the Second International Natural Language Generation Conference*, 97–104, New York, USA.

van Deemter, K. and Halldórsson, M. M. 2001. Logical form equivalence: The case of referring expressions generation. In *Proceedings of the Eighth European Workshop on Natural Language Generation*, Toulouse, France.

van Deemter, K., van der Sluis, I., and Gatt, A. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *Proceedings of the Fourth International Natural Language Generation Conference*, 130–132, Sydney, Australia.

Viethen, J. and Dale, R. 2006. Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the Fourth International Natural Language Generation Conference*, 63–70, Sydney, Australia.

Error correction using utterance disambiguation techniques

Peter Vlugter and **Edwin van der Ham** and **Alistair Knott**
Dept of Computer Science
University of Otago

Abstract

This paper describes a mechanism for identifying errors made by a student during a computer-aided language learning dialogue. The mechanism generates a set of 'perturbations' of the student's original typed utterance, each of which embodies a hypothesis about an error made by the student. Perturbations are then passed through the system's ordinary utterance interpretation pipeline, along with the student's original utterance. An utterance disambiguation algorithm selects the best interpretation, performing error correction as a side-effect.

1 Introduction

The process of identifying and correcting the errors in an input utterance has been extensively studied. Kukich (1992) discusses three progressively more difficult tasks. The first task is the identification of **nonwords** in the utterance. A nonword is by definition a word which is not part of the language, and which therefore must must have been misspelled or mistyped. The difficulty in identifying nonwords is due to the impossibility of assembling a list of all the actual words in a language; some classes of actual words (in particular proper names) are essentially unbounded. So the system should have a reliable way of identifying when an unknown word is likely to belong to such a class.

The second task is to suggest corrections for individual nonwords, based purely on their similarity to existing words, and a model of the likelihood of different sorts of errors. This task is performed quite well by the current generation of spellcheckers, and is to some extent a solved problem.

The third task is to detect and correct **valid word errors**—that is, errors which have resulted in words (for instance *their* misspelled as *there*). This task requires the use of context: the error can only be detected by identifying that a word is out of place in its current context. Probabilistic language models which estimate the likelihood of words based on their neighbouring words have been used quite successfully to identify valid word errors (see e.g. Golding and Schabes (1996); Mangu and Brill (1997); Brill and Moore (2000)); however, there are situations where these techniques are not able to identify the presence of errors, and a richer model of context is needed, making reference to syntax, semantics or pragmatics. In summary, two of the outstanding problems in automated error correction are identifying proper names and detecting and correcting errors which require a sophisticated model of context.

In this paper, we consider a domain where these two problems arise with particular force: computer-aided language learning dialogues (or **CALL dialogues**. In this domain, the system plays the role of a language tutor, and the user is a student learning a target language: the student engages with the system in a dialogue on some preset topic. One of the system's key roles is to identify errors in the student's utterances and to correct these errors (either indirectly, by prompting the student, or directly, by reporting what the student should have said). In either case, it is crucial that the system makes correct diagnoses about student errors. While a regular spell-checker is relatively passive, simply identifying possibly misspelled words, a language tutor frequently takes interventions when detecting errors, and initiates subdialogues aimed at correcting them. (Of course, a tutor may choose to ignore some of the errors she

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 123–130.

identifies, to avoid overwhelming the student with negative feedback, or to concentrate on a particular educational topic. However, it is in the nature of a tutorial dialogue that the tutor frequently picks up on a student's errors.) In this domain, therefore, it is particularly important to get error correction right.

The focus of the paper is on our system's mechanism for error correction, which is tightly integrated with the mechanism for utterance disambiguation. Our claim is that a semantically rich utterance disambiguation scheme can be extended relatively easily to support a sophisticated model of error correction, including the syntactic and semantic errors which are hard for surface based n-gram models of context. We will begin in Section 2 by reviewing the kinds of error found in language-learning dialogues. In Section 3, we discuss some different approaches to modelling language errors, and outline our own approach. In Section 4 we introduce our dialogue-based CALL system. In Section 5 we discuss our approach to error correction in detail: the basic suggestion is to create **perturbations** of the original sentence and interpret these alongside the original sentence, letting the regular utterance disambiguation module decide which interpretation is most likely. In Section 6 we discuss some examples of our system in action, and in Section 7, we discuss how the model can be extended with a treatment of unknown words.

2 The types of error found in language-learning dialogues

Learners of a language can be expected to make more errors than native speakers. If we restrict our errors to those present in typed utterances, some types of error are essentially the same—in particular, we can expect a similar proportion of typos in learners as in native speakers. Other types of error will be qualitatively similar to those made by native speakers, but quantitatively more prevalent—for instance, we expect to find more spelling mistakes in learners than in native speakers, but the mechanisms for detecting and correcting these are likely to be similar. However, there are some types of error which we are likely to find only in language learners. We will consider two examples here.

Firstly, there are **grammatical errors**. The learner of a language does not have a firm grasp of the grammatical rules of the language, and is likely to make mistakes. These typically result in syntactically ill-formed sentences:

(1) T: How are you feeling?[1]
S: I feeling well.

It may be that a bigram-based technique can identify errors of this kind. But it is less likely that such a technique can reliably *correct* such errors. Corrections are likely to be locally suitable (i.e. within a window of two or three words), but beyond this there is no guarantee that the corrected sentence will be grammatically correct. In a CALL system, great care must be taken to ensure that any corrections suggested are at least syntactically correct.

Another common type of errors are **vocabulary errors**. Learners often confuse one word for another, either during interpretation of the tutor's utterances or generation of their own utterances. These can result in utterances which are syntactically correct, but factually incorrect.

(2) T: Where is the bucket?
S: It is on the flour. [meaning 'floor']

(Note that vocabulary errors can manifest themselves as grammatical errors if the wrongly used word is of a different syntactic category.) To detect errors of this sort, the system must have a means of checking utterances against a model of relevant facts in the world.

Thirdly, there are **pragmatic errors**, which involve an utterance which is out of place in the current dialogue context.

(3) T: How are you feeling?
S: You are feeling well.

These errors can result from a failure to comprehend something in the preceding dialogue, or from a grammatical or vocabulary error which happens to result in a syntactically well-formed sentence. To detect and correct errors of this type, a model of coherent dialogue is needed—in particular, a model of the relationship between questions and answers.

These three types of error are relatively common in language-learning dialogues. Detecting them requires relatively deep syntactic and semantic processing of the utterances in the dialogue.

[1] T stands for 'tutor' in these examples, and S stands for 'student'.

Such processing is not yet feasible in an unrestricted dialogue with a native speaker—however, in a language-learning dialogue, there are several extra constraints which make it more feasible. Firstly, a language learner has a much smaller grammar and vocabulary than a native speaker. Thus it may well be feasible to build a grammar which covers all of the constructions and words which the user currently knows. If the system's grammar is relatively small, it may be possible to limit the explosion of ambiguities which are characteristic of wide-coverage grammars. Secondly, the semantic domain of a CALL dialogue is likely to be quite well circumscribed. For one thing, the topics of conversation are limited by the syntax and vocabulary of the student. In practice, the topic of conversation is frequently dictated by the tutor; there is a convention that the tutor is responsible for determining the content of language-learning exercises. Students are relatively happy to engage in semantically trivial dialogues when learning a language, because the content of the dialogue is not the main point; it is simply a means to the end of learning the language.

In summary, while CALL dialogues create some special problems for an error correction system, they also well suited to the deep utterance interpretation techniques which are needed to provide the solutions to these problems.

3 Alternative frameworks for modelling language errors

There are several basic schemes for modelling language errors. One scheme is to construct specialised **error grammars**, which explicitly express rules governing the structures of sentences containing errors. The parse tree for an error-containing utterance then provides very specific information about the error that has been made. We have explored using a system of this kind (Vlugter *et al.*, (2004), and others have pursued this direction quite extensively (see e.g. Michaud *et al.* (2001); Bender *et al.* (2004); Foster and Vogel (2004)). This scheme can be very effective—however, creating the error rules is a very specialised job, which has to be done by a grammar writer. We would prefer a system which makes it easy for language teachers to provide input about the most likely types of error made by students.

Another scheme is to introduce ways of relaxing the constraints imposed by a grammar if a sentence cannot be parsed. The relaxation which results in a successful parse provides information about the type of error which has occurred. This technique has been used effectively by Menzel and Schröder (1998), and similar techniques have been used by Fouvry (2003) for robust parsing. However, as Foster and Vogel note, the technique has problems dealing with errors involving additions or deletions of whole words. In addition, the model of errors is again something considerably more complicated than the models which teachers use when analysing students' utterances and providing feedback.

In the scheme we propose, the parser is left unchanged, only accepting syntactically correct sentences; however, more than one initial input string is sent to the parser. In our scheme, the student's utterance is first permuted in different ways, in accordance with a set of hypotheses about word-level or character-level errors which might have occurred. There are two benefits to this scheme. Firstly, hypotheses are expressed a 'surface' level, in a way which is easy for non-specialists to understand. Secondly, creating multiple input strings in this way allows the process of error correction to be integrated neatly with the process of utterance disambiguation, as will be explained below.

4 Utterance interpretation and disambiguation in our dialogue system

Our CALL dialogue system, called Te Kaitito (Vlugter *et al.* (2004); Knott (2004); Slabbers and Knott (2005)) is designed to assist a student to learn M¯aori. The system can 'play' one or more characters, each of which enters the dialogue with a private knowledge base of facts and an agenda of dialogue moves to make (principally questions to ask the student about him/herself). Each lesson is associated with an agenda of grammatical constructions which the student must show evidence of having assimilated. The system supports a mixed-initiative multi-speaker dialogue: system characters generate initiatives which (if possible) are relevant to the current topic, and feature grammatical constructions which the student has not yet assimilated. System characters can also ask 'checking' questions, to explicitly check the student's assimilation of material presented earlier in the dialogue.

The system's utterance interpretation mechanism takes the form of a pipeline. An utterance

by the student is first parsed, using the LKB system (Copestake (2000)). Our system is configured to work with a small grammar of M¯aori, or a wide-coverage grammar of English, the English resource grammar (Copestake *et al.* (2000)). Each syntactic analysis returned by the parser is associated with a single semantic representation. From each semantic representation a set of **updates** is created, which make explicit how the presuppositions of the utterance are resolved, and what the role of the utterance is in the current dialogue context (i.e. what **dialogue act** it executes). Utterance disambiguation is the process of deciding which of these updates is the intended sense of the utterance.

To disambiguate, we make use of information derived at each stage of the interpretation pipeline. At the syntactic level, we prefer parses which are judged by the probabilistic grammar to be most likely. At the discourse level, we prefer updates which require the fewest presupposition accommodations, or which are densest in successfully resolved presuppositions (Knott and Vlugter (2003)). At the dialogue level, we prefer updates which discharge items from the dialogue stack: in particular, if the most recent item was a question, we prefer a dialogue act which provides an answer over other dialogue acts. In addition, if a user's question is ambiguous, we prefer an interpretation to which we can provide an answer.

Our system takes a 'look-ahead' approach to utterance disambiguation (for details, see Lurcock *et al.* (2004); Lurcock (2005)). We assume that dialogue-level information is more useful for disambiguation than discourse-level information, which is in turn more useful than syntactic information. By preference, the system will derive all dialogue-level interpretations of each possible syntactic analysis. However, if the number of parses exceeds a set threshold, we use the probability of parses as a heuristic to prune the search space.

Each interpretation computed receives an **interpretation score** at all three levels. Interpretation scores are normalised to range between 0 and 10; 0 denotes an impossible interpretation, and 10 denotes a very likely one. (For the syntax level, an interpretation is essentially a probability normalised to lie between 0 and 10, but for other levels they are more heuristically defined.) When interpretations are being compared within a level, we assume a constant **winning margin** for that level, and treat all interpretations which score within this margin of the top-scoring interpretation as joint winners at that level.

If there is a single winning interpretation at the dialogue level, it is chosen, regardless of its scores at the lower levels. If there is a tie between several interpretations at the highest level, the scores for these interpretations at the next level down are consulted, and so on. To resolve any remaining ambiguities at the end of this process, clarification questions are asked, which target syntactic or referential or dialogue-level ambiguities as appropriate.

5 The error correction procedure

Like disambiguation, error correction is a process which involves selecting the most contextually appropriate interpretation of an utterance. If the utterance is uninterpretable as it stands, there are often several different possible corrections which can be made, and the best of these must be selected. Even if the utterance is already interpretable, it may be that the literal interpretation is so hard to accept (either syntactically or semantically) that it is easier to hypothesise an error which caused the utterance to deviate from a different, and more natural, intended reading. The basic idea of modelling error correction by hypothesising intended interpretations which are easier to explain comes from Hobbs *et al.* (1993); in this section, we present our implementation of this idea.

5.1 Perturbations and perturbation scores

Each error hypothesis is modelled as a **perturbation** of the original utterance (Lurcock (2005)). Two types of perturbation are created: **character-level** perturbations (assumed to be either typos or spelling errors) and **word-level perturbations** (assumed to reflect language errors). For character-level perturbations, we adopt Kukich's (1992) identification of four common error types: **insertion** of an extra character, **deletion** of a character, **transposition** of two adjacent characters and **substitution** of one character by another. Kukich notes that 80% of misspelled words contain a single instance of one of these error types. For word-level perturbations, we likewise permit insertion, deletion, transposition and substitution of words.

Each perturbation created is associated with a

'perturbation score'. This score varies between 0 and 1, with 1 representing an error which is so common that it costs nothing to assume it has occurred, and 0 representing an error which never occurs. (Note again that these scores are not probabilities, though in some cases they are derived from probabilistic calculations.) When the interpretation scores of perturbed utterances are being compared to determine their likelihood as the intended sense of the utterance, these costs need to be taken into account. In the remainder of this section, we will describe how perturbations are created and assigned scores. Details can be found in van der Ham (2005).

5.1.1 Character-level perturbations

In our current simple algorithm, we perform all possible character-level insertions, deletions, substitutions and transpositions on every word. ('Space' is included in the set of characters, to allow for inappropriately placed word boundaries.) Each perturbation is first checked against the system's lexicon, to eliminate any perturbations resulting in nonwords. The remaining perturbations are each associated with a score. The scoring function takes into account several factors, such as phonological closeness and keyboard position of characters. In addition, there is a strong penalty for perturbations of very short words, reflecting the high likelihood that perturbations generate new words simply by chance.

To illustrate the character-level perturbation scheme, if we use the ERG's lexicon and English parameter settings, the set of possible perturbations for the user input word *sdorted* is *sorted*, *sported* and *snorted*. The first of these results from hypothesising a character insertion error; the latter two result from hypothesising character substitution errors. The first perturbation has a score of 0.76; the other two both have a score of 0.06. (The first scores higher mainly because of the closeness of the 's' and 'd' keys on the keyboard.)

5.1.2 Word-level perturbations

As already mentioned, we employ Kukich's taxonomy of errors at the whole word level as well as at the single character level. Thus we consider a range of whole-word insertions, deletions, substitutions and transpositions. Clearly it is not possible to explore the full space of perturbations at the whole word level, since the number of possible words is large. Instead, we want error hypotheses to be driven by a model of the errors which are actually made by students.

Our approach has been to compile a database of commonly occurring whole-word language errors. This database consists of a set of sentence pairs $\langle S_{err}, S_c \rangle$, where S_{err} is a sentence containing exactly one whole-word insertion, deletion, substitution or transposition, and S_c is the same sentence with the error corrected. This database is simple to compile from a technical point of view, but of course requires domain expertise: in fact, the job of building the database is not in fact very different from the regular job of correcting students' written work. Our database was compiled by the teacher of the introductory M$\bar{\text{a}}$ori course which our system is designed to accompany. Figure 1 illustrates with some entries in a (very simple) database of English learner errors. (Note how missing words

Error sentence	Correct sentence	Error type
I saw GAP dog	I saw a dog	Deletion (a)
I saw GAP dog	I saw the dog	Deletion (the)
He plays the football	He plays GAP football	Insertion (the)
I saw a dog big	I saw a big dog	Transposition

Figure 1: Extracts from a simple database of whole-word English language errors

in the error sentence are replaced with the token "GAP".)

Given the student's input string, we consult the error database to generate a set of candidate word-level perturbations. The input string is divided into positions, one preceding each word. For each position, we consider the possibility of a deletion error (at that position), an insertion error (of the word following that position), a substitution error (of the word following that position) and a transposition error (of the words preceding and following that position). To generate a candidate perturbation, there must be supporting evidence in the error database: in each case, there must be at least one instance of the error in the database, involving at least one of the same words. So, for instance, to hypothesise an insertion error at the current position (i.e. an error where the word w following that position has been wrongly inserted and needs to be removed) we must find at least one instance in the database of an insertion error involving the word w.

To calculate scores for each candidate perturbation, we use the error database to generate a probability model, in which each event is a *rewriting* of

a given word sequence S_{orig} as a perturbed word sequence S_{pert} (which we write as $S_{orig} \rightarrow S_{pert}$.) The database may contain several different ways of perturbing the word sequence S_{orig}. The relative frequencies of the different perturbations can be used to estimate perturbation probabilities, as follows:

$$P(S_{orig} \rightarrow S_{pert}) \approx \frac{count(S_{orig} \rightarrow S_{pert})}{count(S_{orig} \rightarrow \_)}$$

(The denominator holds the count of all perturbations of S_{orig} in the error database.)

Naturally, if we want useful counts, we cannot look up a complete sentence in the error database. Instead, we work with an n-gram model, in which the probability of a perturbed sentence is approximated by the probability of a perturbation in an n-word sequence centred on the perturbed word. In our model, the best approximation is a perturbed trigram; thus if the student's input string is *I saw dog*, the probability of a perturbation creating *I saw the dog* is given by

$$\frac{count(saw\ GAP\ dog \rightarrow saw\ the\ dog)}{count(saw\ GAP\ dog \rightarrow \_)}$$

Again, it is unlikely these counts are going to be high enough, so we also derive additional backed-off estimates, two based on bigrams and one based on unigrams:

$$\frac{count(GAP\ dog \rightarrow the\ dog)}{count(GAP\ dog \rightarrow \_)}$$

$$\frac{count(saw\ GAP \rightarrow saw\ the)}{count(saw\ GAP \rightarrow \_)}$$

$$\frac{count(GAP \rightarrow the)}{count(GAP \rightarrow \_)}$$

See van der Ham (2005) for details of the backoff and discounting schemes used to derive a single probability from these different approximations.

5.2 Integrating perturbations into utterance disambiguation

When an incoming utterance is received, a set of perturbations is generated. Naturally, we do not want to hypothesise all possible perturbations, but only the most likely ones—i.e. those whose score exceeds some threshold. The threshold is currently set at 0.8. We also want to keep the number of hypothesised perturbations to a minimum. Currently we only allow one perturbation per utterance, except for a special class of particularly common spelling mistakes involving placement of macron accents, of which we allow three. (Where there are multiple perturbations, their scores are multiplied.)

Each perturbed sentence is passed to the utterance interpretation module. Most of the perturbations result in ungrammatical sentences, and so fail at the first hurdle. However, for any which can be parsed, one or more full updates is created. The complete set of updates produced from the original sentence and all its selected perturbations are then passed to the disambiguation module.

The disambiguation module must now take into account both the interpretation score and the perturbation score when deciding between alternative interpretations. At any level, the module computes an **aggregate score** S_{agg}, which is the product of the perturbation score S_{pert} (weighted by a perturbation penalty) and the interpretation score S_{int}:

$$S_{agg} = \frac{S_{pert}}{pert\_penalty} \times S_{int}$$

(The perturbation penalty is a system parameter, which determines the importance of perturbation scores relative to interpretation scores; it is currently set to 1.) To choose between alternative interpretations at a given level, we now take all interpretations whose aggregate score is within the winning margin of the highest aggregate score.

5.3 Responding to the user's utterance

After the utterance disambiguation process is complete, either a single interpretation remains, or a set of interpretations whose aggregate scores are too close to call at any of the three levels. In either case, how the system responds depends on whether the remaining interpretations derive from the unperturbed utterance or from a perturbed version.

If a single interpretation remains, then if it derives from the original utterance, the dialogue manager responds to it in the usual way. However, if it derives from a perturbed utterance, then the system is confident that an error has occurred, and that it knows what the error is. In this case the system enters a subdialogue with the user to address the error. Our system's current strategy is simply to report the error explicitly:

(4) I think you mean [perturbed utterance]. Please try again!

(If the student's utterance was responding to a forward-looking dialogue act—e.g. a question—the system then reiterates this forward-looking act, to recreate the context for the student's second attempt.) Note that a good tutor would probably give the student an opportunity to correct her error herself; we are still exploring ways of doing this without irritating the student.

If more than one interpretation remains when utterance disambiguation is complete, what happens again depends on where the interpretations come from. If they all come from the unperturbed utterance, an ordinary clarifiction question is asked (see Lurcock (2005) for details of how clarification questions are generated). If they all come from *a single* perturbed utterance, we simply present the suggested correction, as above; if the student then enters the hypothesised correction, a regular clarification question will be asked. However, it is also possible that the interpretations come from several different perturbations. In this case, we formulate our error feedback as a question:

(5) Do you mean [perturbation 1] or (...) or [perturbation n]?[2]

6 Some examples

In this section, we give some examples of our error-correction scheme in action. First, a simple character-level correction:

(6)
U: I am hapy[3]
S: I think you mean 'I am **happy**'.

The perturbation here is motivated by syntactic well-formedness: the original utterance does not parse, but the perturbed utterance does.

Here is another character-level correction, this time informed by contextual appropriateness:

(7)
U: I saw my chief
S: Okay
U: The chef is happy
S: I think you mean 'The **chief** is happy'.

There are two things to note about this example. Firstly, note that the user's original utterance is syntactically correct, so a full interpretation will be derived for this utterance as well as for the version perturbing *chef* to *chief*. When these two interpretations are compared by the disambiguation module, the perturbed version is preferred, because it is cheaper to incorporate into the current dialogue context: *the chief* refers back to an existing discourse entity, while *the chef* requires the accommodation of a new one.

Here is a final example, this time at the level of whole-word perturbations:

(8)
S: What is your name?
U: Your name is Sally.
S: I think you mean '**My** name is Sally'.

The error database contains enough instances of the perturbation *your→my* to cause the system to create this candidate perturbation; an interpretation for this perturbation is thus created alongside that of the original utterance. Again, the interpretation deriving from the perturbation is easier to incorporate into the dialogue context, since it answers the system's question, so the perturbed sentence is preferred over the original, even though the original contains no syntactic errors.

7 Future work: incorporating a treatment of unknown words

The error correction scheme has performed reasonably well in informal user trials. However there is one fairly major problem still to be addressed, relating to unknown words. If a word in the student's utterance is not found in the system's lexicon, there are two possibilities: either the student has made an error, or the word is one which the system simply does not know. In the current scheme, only the first possibility is considered.

We have already implemented a treatment of unknown words, in which the system assumes an unknown word is of a lexical type already defined in the grammar, and proceeds by asking the user questions embedding the word in example sentences to help identify this type (see van Schagen and Knott (2004)). However, word-authoring subdialogues would be a distraction for a student; and in any case, it is fairly safe to assume that all unknown words used by the student are proper names. We therefore use a simpler treatment related to the constraint-relaxation scheme of Fouvry (2003), in which the system temporarily adds an unknown word to the class of proper names and then attempts to reparse the sentence. A successful parse is then interpreted as evidence that the unknown word is indeed a proper name.

[2] The question is formulated as a multiple choice question, using the same format as some types of syntactic clarifi cation question.

A problem arises with this scheme when it is used in conjunction with error-correction: wherever it is possible to use a proper name, hypothesising a proper name gives a higher aggregate score than hypothesising an error, all other things being equal. The problem is serious, because grammars typically allow proper names in many different places, in particular as preposed and postposed sentence adverbials functioning as addressee terms (see Knott *et al.* (2004)). To remedy this problem, it is important to attach a cost to the operation of hypothesising a proper name, comparable to that of hypothesising an error.

In our (as-yet unimplemented) combined unknown-word and error-correction scheme, if there is an unknown word which can be interpreted as a proper name, the lexicon is updated prior to parsing, and perturbations are created as usual. A special **unknown word cost** is associated with the original utterance and with each of these perturbations, except any perturbations which alter the unknown word (and thus do not rely on the hypothesised lexical item). The unknown word cost is another number between 0 and 1, and the aggregate score of an interpretation is multiplied by this number when deciding amongst alternative interpretations. The number is set to be lower than the average perturbation score. If any perturbations of the unknown word survive the parsing process, they stand a good chance of being preferred over the proper name hypothesis, or at least being presented as alternatives to it. We will experiment with this extension to the error-correction algorithm in future work.

References

E Bender, D Flickinger, S Oepen, A Walsh, and T Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL/ICALL Symposium*, pages 83–87.

E Brill and R Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL*.

A Copestake and D Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using hpsg. In *Proceedings of LREC 2000*, Athens, Greece.

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

J Foster and C Vogel. 2004. Parsing ill-formed text using an error grammar. *Artificial Intelligence Review*, 21(3–4):269–291.

F Fouvry. 2003. Lexicon acquisition with a large-coverage unifi cation-based grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics, Research notes and demos*, Budapest, Hungary.

R Golding and Y Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*.

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

A Knott and P Vlugter. 2003. Syntactic disambiguation using presupposition resolution. In *Proceedings of the 4th Australasian Language Technology Workshop (ALTW2003)*, Melbourne.

A Knott, I Bayard, and P Vlugter. 2004. Multi-agent human-machine dialogue: issues in dialogue management and referring expression semantics. In *Proceedings of the 8th Pacific Rim Conference on Artificial Intelligence (PRICAI 2004)*, pages 872–881, Auckland. Springer Verlag: Lecture Notes in AI.

K Kukich. 1992. Techniques for automatically correcting words in text. *Computing Surveys*, 24(4):377–439.

P Lurcock, P Vlugter, and A Knott. 2004. A framework for utterance disambiguation in dialogue. In *Proceedings of the 2004 Australasian Language Technology Workshop (ALTW)*, pages 101–108, Macquarie University.

P Lurcock. 2005. Techniques for utterance disambiguation in a human-computer dialogue system. MSc thesis, Dept of Computer Science, University of Otago.

L Mangu and E Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning*.

W Menzel and I Schröder. 1998. Constraint-based diagnosis for intelligent language tutoring systems.

L Michaud, K Mccoy, and L Stark. 2001. Modeling the acquisition of english: an intelligent CALL approach. In *Proceedings of The 8th International Conference on User Modeling*, pages 13–17, Sonthofen, Germany.

N Slabbers. 2005. A system for generating teaching initiatives in a computer-aided language learning dialogue. Technical Report OUCS-2005-02, Department of Computer Science, University of Otago, Dunedin, New Zealand.

E van der Ham. 2005. Diagnosing and responding to student errors in a dialogue-based computer-aided language-learning system. Technical Report OUCS-2005-06, Department of Computer Science, University of Otago, Dunedin, New Zealand.

M van Schagen and A Knott. 2004. Tauira: A tool for acquiring unknown words in a dialogue context. In *Proceedings of the 2004 Australasian Language Technology Workshop (ALTW)*, pages 131–138, Macquarie University.

P Vlugter, A Knott, and V Weatherall. 2004. A human-machine dialogue system for CALL. In *Proceedings of InSTIL/ICALL 2004: NLP and speech technologies in Advanced Language Learning Systems*, pages 215–218, Venice.

Using Dependency-Based Features to Take the "Para-farce" out of Paraphrase

Stephen Wan[†‡] **Mark Dras**[†] **Robert Dale**[†]
[†]Center for Language Technology
Div of Information Communication Sciences
Macquarie University
Sydney, NSW 2113
swan,madras,rdale@ics.mq.edu.au

Cécile Paris[‡]
[‡]Information and Communication Technologies
CSIRO
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

As research in text-to-text paraphrase generation progresses, it has the potential to improve the quality of generated text. However, the use of paraphrase generation methods creates a secondary problem. We must ensure that generated novel sentences are not inconsistent with the text from which it was generated. We propose a machine learning approach be used to filter out inconsistent novel sentences, or *False Paraphrases*. To train such a filter, we use the Microsoft Research Paraphrase corpus and investigate whether features based on syntactic dependencies can aid us in this task. Like Finch et al. (2005), we obtain a classification accuracy of 75.6%, the best known performance for this corpus. We also examine the strengths and weaknesses of dependency based features and conclude that they may be useful in more accurately classifying cases of False Paraphrase.

1 Introduction

In recent years, interest has grown in paraphrase generation methods. The use of paraphrase generation tools has been envisaged for applications ranging from abstract-like summarisation (see for example, Barzilay and Lee (2003), Daumé and Marcu (2005), Wan et al. (2005)), question-answering (for example, Marsi and Krahmer (2005)) and Machine Translation Evaluation (for example, Bannard and Callison-Burch (2005) and Yves Lepage (2005)). These approaches all employ a loose definition of paraphrase attributable to Dras (1999), who defines a 'paraphrase pair' operationally to be "a pair of units of text deemed to be interchangeable". Notably, such a definition of paraphrase lends itself easily to corpora based methods. Furthermore, what the more modern approaches share is the fact that often they generate new paraphrases from raw text not semantic representations. The generation of paraphrases from raw text is a specific type of what is commonly referred to as *text-to-text generation* (Barzilay and Lee, 2003).

As techniques for generating paraphrases improve and basic concerns such as grammaticality are less of an issue, we are faced with an additional concern. That is, we must validate whether or not the generated novel sentence is in fact a paraphrase. It may be detrimental in some applications, for example abstract-like summarisation, to allow a novel sentence that is inconsistent with the content of the input text to be presented to the end user.

As an example of the type of inconsistencies that can arise from paraphrase generation, in Figure 1, we present two examples of generated sentences. In each example, a sentence pair is presented in which the second sentence was generated from an input news article statistically using a four-gram language model and a probabilistic word selection module. Although other paraphrase generation approaches differ in their underlying mechanisms[1], most generate a novel sentence that cannot be found verbatim in the input text.

The generated second sentence of the example is intended to be a paraphrase of the article headline. One might be convinced that the first exam-

[1]The details of the generation algorithm used for this example are peripheral to the focus of this paper and we direct the interested reader to Wan et al. (2005) for more details.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 131–138.

Example 1:
Original Headline:
European feeds remain calm on higher dollar.

Generated Sentence:
The European meals and feeds prices were firm on a stronger dollar; kept most buyers in this market.

Example 2:
Original Headline:
India's Gujral says too early to recognise Taleban.

Generated Sentence:
Prime Minister Inder Kumar Gujral of India and Pakistan to recognise the Taleban government in Kabul.

Figure 1: Two examples of generated novel sentences. Articles from the Reuters corpus were fed as input to the statistical summary generation system.

ple passes as a paraphrase, however the second is clearly inconsistent. We would like to identify this sentence pair as a false paraphrase. In addition to the ambiguity in the subject noun phrase (The Prime Minister of Pakistan is not the same as that of India), the generated sentence seems to ignore the adverbial phrase "too early" resulting in a farcical sentence that is almost the polar opposite of the headline.

We propose that an automatic classifier be employed to identify and filter out inconsistent novel sentences. To do so, we couch Paraphrase Classification as a supervised machine learning task and train a classifier on the Microsoft Research Paraphrase (MSR) Corpus (Dolan et al., 2004), a corpus specifically collected for this task. In particular, we are especially interested in exploring the use of syntactic dependency information in making this classification.

In this paper, we present our findings in training, testing and evaluating a paraphrase classifier. Section 2, we describe the research problem and outline related work in paraphrase classification. In Section 3, we present the features used in our classifier. Our classification experiments and results are described in Section 4. Before concluding, we discuss the strengths and weaknesses of dependency-based features in Section 5.

2 Paraphrase Classification and Related Work

In general, our task is to compare two sentences and produce a binary classification indicating if one is interchangeable with the other. To do so, we adopt the 'entailment decision' problem as put forward by the Pascal Recognising Textual Entailment (RTE) challenge (Dagan et al., 2005). The challenge requires participant systems to decide, given a pair of sentences, if the first sentence (referred to as the *hypothesis*) is entailed by the second sentence (referred to as the *text*). Although the task is that of logical entailment, participant systems are free to use any method, logic-based or not, to decide if sentence pairs are entailments. Crucial to this exercise is the simplification that the entailment decision be made on the basis of information within the sentences alone, and not on extensive representations of extensive world knowledge.

Similarly in our task, two sentences are checked for 'entailment'. In contrast to the RTE challenge, the MSR corpus has been collected based on a definition of paraphrase pairs as bi-directional entailment. That is, we must decide if one sentence is 'entailed' by its paired sentence, and vice versa. Sentence pairs were annotated as being True paraphrases if they were judged to be 'more or less semantically equivalent'. Otherwise, sentence pairs were annotated as being False Paraphrases.

Previous approaches to this classification task have focused on semantic equivalence at both the word and syntax level. Papers standardly report classification accuracy which is defined as the number of correctly classified test cases divided by the total number of test cases. Corley and Mihalcea (2005) use word equivalence features resulting in a classification accuracy of 71.5%. Zhang and Patrick (2005) examine string edit distance features and ngram overlap features collected on pairs of sentences in their canonical form. An overall accuracy of 71.9% is obtained.

Qiu et al. (2006) also focus on the detection of False Paraphrases. In this work, features based on predicate-argument information designed to indicate dissimilarity between the two sentences are collected. Using a support vector machine, this method results in an accuracy of 72.0%.

The best published result for this classification task is obtained by Finch et al. (2005) who obtained a classification accuracy of 74.96% using a

1 unigram recall
2 unigram precision
3 lemmatised unigram precision
4 lemmatised unigram recall
5 Bleu precision
6 Bleu recall
7 lemmatised Bleu precision
8 lemmatised Bleu recall
9 fmeasure
10 dependency relation precision
11 dependency relation recall
12 lemmatised dependency relation precision
13 lemmatised dependency relation recall
14 tree-edit distance (Zhang and Sasha algorithm)
15 lemmatised tree-edit distance (Zhang and Sasha algorithm)
16 difference in sentence length (in words)
17 absolute difference in sentence length (in words)

Figure 2: A list of all possible features

support vector machine trained on relatively simple features based on ngram overlap.

3 Features

In this paper, we decided to explore features encoding information about the relative difference between the structures of the two sentence. We thus experimented with a range of features ranging from differences in sentence length, to word overlap, to syntax dependency tree overlap, where the latter approximately represent predicate and argument structure. Figure 2 presents an overview of our features. We now describe each of these features.

3.1 *N*-gram Overlap: Features 1 to 9

We used variety of features based on word overlap and word-sequence overlap, where tokenisation is delimited by white space. We considered unigram overlap and explored two metrics, recall (feature 1) and precision (feature 2), where a precision score is defined as:

$$\text{precision} = \frac{\textit{word-overlap}(sentence_1, sentence_2)}{\textit{word-count}(sentence_1)}$$

and recall is defined as:

$$\text{recall} = \frac{\textit{word-overlap}(sentence_1, sentence_2)}{\textit{word-count}(sentence_2)}$$

For each of the unigram overlap features described, we also computed a lemmatised variant. Both sentences were parsed by the Connexor parser[2] which provides lemmatisation information. For both sentences, each original word is replaced by its lemma. We then calculated our unigram precision and recall scores as before (features 3 and 4).

The Bleu metric (Papineni et al., 2002), which uses the geometric average of unigram, bigram and trigram precision scores, is implemented as feature 5. The score was obtained using the original Bleu formula[3] with a brevity penalty set to 1 (that is, the brevity penalty is ignored). Note that in our usage, there is only one 'reference' sentence. By reversing which sentence was considered the 'test' sentence and which was considered the 'reference', a recall version of Bleu was obtained (feature 6). Lemmatised versions provided features 7 and 8.

Finally, because of the bi-directionality property of paraphrase, the F-Measure[4], which combines both precision and recall into a single score using the harmonic mean, was implemented as feature 9.

3.2 Dependency Relation Overlap: Features 10 to 13

Overlap of dependency tuples has been cited by other researchers as being a useful approximate representation of sentence meaning (Mollá, 2003). Indeed, Rouge-BE (Hovy et al., 2005), a recall-based metric similar to this feature, is currently being used in summarisation evaluations to measure the content overlap of summaries with source documents.

We again make use of the Connexor parser, this time to provide a dependency structure analysis of a sentence. Each sentence was parsed resulting in a set of dependency relations (one set per sentence). A relation is simply a pair of words in a parent-child relationship within the dependency tree[5], refered to as head-modifier relationships. In this paper, we ignored the label of the relationships which indicates the semantic role. The next series of features examines the use of features based on an overlap of such head-modifier relations (hereafter, *relations*) between sentences.

Feature 10 is the precision score calculated from the overlap according to the following formula:

[2] see http://www.connexor.com/software/syntax/
[3] http://www.ics.mq.edu.au/∼szwarts/downloads/Bleu.cpp
[4] http://www.ics.mq.edu.au/∼szwarts/downloads/FMeasure.cpp
[5] That is, an edge and the two nodes on either side

$$\text{precision}_d = \frac{|\textit{relations}(\text{sentence}_1) \cap \textit{relations}(\text{sentence}_2)|}{|\textit{relations}(\text{sentence}_1)|}$$

where precision_d stands for *dependency precision* and $\text{relations}(\text{sentence}_i)$ is the set of head-modifier relations for some sentence.

A recall variant of this feature was also used (feature 11) and is defined as:

$$\text{recall}_d = \frac{|\textit{relations}(\text{sentence}_1) \cap \textit{relations}(\text{sentence}_2)|}{|\textit{relations}(\text{sentence}_2)|}$$

Lemmatised versions of these features were used in feature 12 and 13.

3.3 Dependency Tree-Edit Distance: Features 14 and 15

As another measure of how alike or different the two sentences are from each other, we decided to examine how similar their respective dependency trees were. Ordered tree-edit distance algorithms are designed to find the least costly set of operations that will transform one tree into another. In our case, we want to find the cost of transforming dependency parse trees.

Our implementation is based on the dynamic programming algorithm of Zhang and Shasha (1989). The algorithm finds the optimum (cheapest) set of tree-edit operations in polynomial time. This algorithm has been used in the past in Question-Answering as a means of scoring similarity between questions an candidate answers (Punyakanok et al., 2004). In a similar vein to our work here, it has also been used in the RTE challenge (Kouylekov and Magnini, 2005).

We calculated the tree-edit distance over the syntactic dependency parse trees returned by the Connexor parser. Inserting, deleting and renaming nodes, or words, into a dependency tree, were all given an equal cost.

The cost returned by the algorithm is simply the sum of all operations required to transform one tree into the other. This cost was normalised by the number nodes in the target dependency tree to produce a value between 0 and 1 (feature 14). A lemmatised variant of this feature was obtained by first lemmatising the two dependency trees (feature 15).

3.4 Surface Features: Features 16 and 17

Finally, we looked at the difference in length of the two sentences as measured in words by subtracting one length from the other. This difference (feature 16) could be a negative or positive integer. An absolute variant was used in Feature 17.

4 Experiments

4.1 Data and Software

The Microsoft Paraphrase Corpus (MSR) (Dolan et al., 2004) is divided into a training set and a test set. In the original training set, there were 2753 True Paraphrase pairs and 1323 False Paraphrase pairs. The original test set contained 1147 True Paraphrases pairs and 578 False Paraphrases pairs.

We first parsed the MSR paraphrase corpus using the Connexor parser. While Connexor is by no means a perfect parser, it usually produces partial parses if a more complete one is not possible. Our experience with Connexor is that these partial parses have tended to be useful. We are currently comparing Connexor to other dependency parsers to see what kinds of errors it introduces. However, due to time constraints, utilising this information is left for future work.

Because there were several cases which broke our parsing scripts (due to an occasional non-XML character), our training and test sets were slightly smaller. These included 2687 True Paraphrase pairs and 1275 False Paraphrase pairs in our training set, and 1130 True Paraphrase pairs and 553 False Paraphrases pairs in our test set.

We used the open source WEKA Data Mining Software (Witten and Frank, 2000). A selection of commonly used techniques was experimented with including: a Naive Bayes learner (bayes.NaiveBayes), a clone of the C4.5 decision tree classifier (trees.J48), a support vector machine with a polynomial kernel (functions.SMO), and K-nearest neighbour (lazy.IBk). Each machine learning technique was used with the default configurations provided by WEKA. The baseline learning technique (rules.ZeroR) is simply the performance obtained by choosing the most frequent class. We report only the results obtained with the support vector machine as this machine learning method consistently outperformed the other methods for this task.

Finally, we tested for significance between correct and incorrect classifications of the two systems being compared using the Chi-squared test

Features	Acc.	C1-prec.	C1-recall	C1-Fmeas.	C2-prec	C2-recall	C2-Fmeas.
lemma'd 1-grams	0.69	0.52	0.56	0.54	0.78	0.75	0.76
1-grams	0.73	0.63	0.39	0.49	0.75	0.89	0.81
ZeroR	0.66	0	0	0	0.67	1	0.80
Finch	0.75	0.69	0.46	0.55	0.77	0.90	0.83
Best Features	0.75	0.70	0.46	0.55	0.77	0.90	0.83

Table 1: Classification performance of the best feature vector found. C1 denotes False Paraphrase pairs, C2 denotes True Paraphrase pairs. C1 scores for the best system in Finch et al. (2005) were calculated from the C2 scores published.

Features	Acc.	C1-prec.	C1-recall	C1-Fmeas.	C2-prec	C2-recall	C2-Fmeas.
Dependencies	0.75	0.67	0.45	0.54	0.77	0.89	0.82
Bleu	0.75	0.69	0.45	0.55	0.77	0.90	0.83

Table 2: Classification performance comparison between dependency features and *n*-gram features. C1 denotes False Paraphrase pairs, C2 denotes True Paraphrase pairs.

implemented in the R-Statistical package.

4.2 Best Performing Feature Set

Through experimentation, we found the best performing classifier used all features except for lemmatised unigrams[6]. The results on the test set are presented in Table 1. Accuracy is the number of correctly classified test cases (regardless of class) divided by the total number of test cases. Recall for True Paraphrase class is defined as the number of cases correctly classified as True Paraphrase divided by the total number of True Paraphrase test cases. Precision differs in that the denominator is the total number of cases (correct or not) classified as True Paraphrase by the system. The F-Measure is the harmonic mean of recall and precision. Likewise, the recall, precision and f-measure for the False Paraphrase class is defined analogously.

We note an improvement over majority class baseline, a unigram baseline and a lemmatised unigram baseline. In particular, the addition of our features add a (3%) improvement in overall accuracy compared to the best performing baseline using (unlemmatised) unigram features. Improvement over this baseline (and hence the other baselines) was statistically significant (χ-squared = 4.107, df = 1, p-value = 0.04271). Our performance was very close to that reported by (Finch et al. (2005) is not statistically significant. The system employed by Finch et al. (2005) uses features that are predominantly based on the Bleu metric.

The improvement of the unigram-based classifier is 6 percentage points above the majority class is also significant (χ-squared = 11.4256, df = 1, p-value = 0.0007244). Interestingly, results from using just precision and recall unigram features[7] without lemmatisation are comparable to Finch et al. (2005). Indeed, a principal components analysis showed that unigram features were the most informative accounting for 60% of cases.

Oddly, the results for the lemmatised unigram features are poorer even the majority class baseline, as demonstrated by a lower True Paraphrase F-Measure. Why this is so is puzzling as one would expect lemmatisation, which abstracts away from morphological variants, to increase the similarity between two sentences. However, we note that two sentences can differ in meaning with the inclusion of a single negative adverb. Thus, an increased similarity for all training cases may simply make it much harder for the machine learning algorithm to differentiate effectively between classes.

5 The Strengths and Weaknesses of Dependency Features

The previous experiment showed that together, Bleu-based features and dependency-based features were able to achieve some improvement. We were also interested in comparing both feature types to see if one had any advantage over the other.

We note that bigrams and dependencies in ac-

[6] features: 1,2,5,6,7,8,9,10,11,12,13,14,15,16,17

[7] features: 1,2

2685	True		False		1275
	Bleu	Dep	Bleu	Dep	
A: 44	F	T	F	T	B: 41
C: 2342	T	T	T	T	D: 654
E: 50	T	F	T	F	F: 49
G: 249	F:	F	F	F	H: 531

Table 3: Error Analysis showing the number of cases in which the Bleu-based classifier disagreed with the Dependency-based classifier. 'T' and 'F' stand for predicted 'TRUE' and predicted 'FALSE'. The other capital letters A to H are cell labels for ease of reference.

tuality encode very similar types of information, that is a pairing of two words. In the case of dependency relations, the words are connected via some syntactic dependency structure, whereas word pairs in bigrams (for example) are merely 'connected' via the property of adjacency. However, Collins (1996) points out that in English, around 70% of dependencies are in fact adjacent words. Thus one would think that Bleu and dependency features have similar discriminative power.

Two versions of the classifier were trained based on two separate sets of features that differed only in that one included four Bleu features[8] whereas the other included four dependency overlap features[9]. All other features were kept constant.

The results obtained on the test set are presented in Table 2. As expected, the two seem to perform at the same level of performance and were not statistically different. This is consistent with the same levels of performance observed between our system and that of Finch et al. (2005) in Table 1. However, it would also be interesting to know if each feature might be more suitable for different types of paraphrase phenomena.

5.1 Differences in Predictions

To understand the strengths and weaknesses of *n*-gram and dependency features, we performed an analysis of the cases where they differed in their classifications. We tested the two classifiers in Table 2 on the training set to gives us an indication of the ideal situation in which the training data reflects the testing data perfectly. Table 3 presents this analysis. For example, Cell A indicates that there were 44 true paraphrase cases that were correctly classified by the dependency-based classifier but misclassified by the bleu-based classifier.

For the dependency-based classifier to outperform the Bleu-based classifier in classifying True Paraphrases, Cell A must be greater than Cell E. That is, the number of cases in which the Dependency-based classifier improves the true positive count must outweigh the false negative count. Unfortunately, this isn't the case.

Correspondingly, for the dependency-based classifier to outperform the Bleu-based classifier in classifying True Paraphrases, Cell F must be greater than Cell B. In this case, the dependency-based classifier does performs better than the Bleu-based classifier.

One could summarise this analysis by saying that the dependency-based classifier tended make pairs look more dissimilar than the Bleu-based classifier. To gain some insight as to how to create features that build on the strengths of the two feature types, for example using dependency based features to better classify False Paraphrase cases, we manually examined the sentence pairs from the training set in which the two classifiers disagreed in the hopes of identifying reasons for the erroneous classifications.

5.2 Wrongly Predicting 'True' on False Paraphrase cases

Table 3 suggest that dependency-features might improve the precision and recall of the False Paraphrase class. Thus, we focused on the cases where the dependency-based classifier incorrectly classified False Paraphrase cases. We found several situations where this was the case. Often, some portion of both sentences would share a high degree of word overlap that we suspect was confusing our classifier.

In the case of Sentence Pair 1, a title is quoted in both increasing the textual similarity. However, on closer inspection the clauses are different, specifically the main clause verb and subject. In Sentence Pair 2, we notice this also happened with long noun phrases relating to organisations.

Sentence Pair 1:
Details of the research appear in the Nov. 5 issue of the Journal of the American Medical Association.
The results, published in the Journal of the American Medical Association, involved just 47 heart attack patients.

Sentence Pair 2:
The Securities and Exchange Commission has also initiated an informal probe of Coke.

[8] features: 1,2,5,6,7,8,16,17
[9] features: 1,2,10,11,12,13,16,17

That federal investigation is separate from an informal inquiry by the Securities and Exchange Commission.

Similarly, despite high overlap in both words and dependency relations, some sentences pairs simply differed in the focus of the main clause as in Sentence Pair 3. We see a similar problem in Sentence Pair 4 in which the main clause of the first sentence matches the subordinate clause of the second but the focus of each is different.

Sentence Pair 3:
He replaces Ron Dittemore, who announced his resignation in April.
Dittemore announced his plans to resign on April 23.

Sentence Pair 4:
Peterson told police he fished alone in San Francisco Bay on Christmas Eve, returning to an empty house.
Peterson told police he left his wife at about 9:30 a.m. on Dec. 24 to fish alone in San Francisco Bay.

5.3 Follow-on Experiment

One of the reasons why our use of dependencies leads to the problem exemplified by Sentence Pairs 1 to 4, is that all dependency pairs are treated equal. However, clearly, some are more equal than others. Dependency relations concerning the main verb and subject ought to count for more.

The simplest way to model this inequality is to give more weight to relations higher up in the tree as these will tend to express the semantics of the main clause.

Our extra set of features represent the weighted dependency precision, the weighted dependency recall, and the lemmatised versions of both those feature types. In total four new features were added.

To begin with nodes were scored with the size of their subtrees. We then traversed the tree breadth-first where siblings were traversed in decreasing order with respect to the size of their respective subtrees. Nodes were given a position number according to this traversal. Each node was then weighted by the inverse of its position in this ordering. Thus, the root would have weight 1 and it's heaviest child node would receive a weight of 0.5. The relation weight is simply the product of the weights of the nodes.

The overall score for the sentence pair is simply the sum of relation weights normalised accordingly to yield precision and recall scores.

The results on the test set are presented in Table 4. Note that this result differs drastically from all the previous systems reported. In contrast to these systems, our last classifier seems to produce good precision results (83%) for the True Paraphrase class at the expense of recall performance. Consequently, it has the best performing recall for False Paraphrase (71%) out of all the systems tested. This gain in recall, while compensated by a loss in precision, ultimately leads to the highest F-Measure observed for this class (61%), an improvement on Finch et al. (2005). This seems to suggest that our additional features are doing what we hoped they would, improve the classification of the False Paraphrase class. However, this effect also has an overall harmful effect on our classifier which may be over-classifying cases as False Paraphrase. Thus, a drop in accuracy is observed. Avenues to integrate the benefits of these new features without harming our overall accuracy remain further work.

6 Conclusion

In this paper, we presented work on Paraphrase Classification with the Microsoft Research Paraphrase Corpus. We show that dependency-based features in conjunction with bigram features improve upon the previously published work to give us the best reported classification accuracy on this corpus, equal with Finch et al. (2005). In addition, using weighted dependency overlap seems to provide promise, yielding the best F-Measure for False Paraphrase classification seen so far. We conclude that dependency features may thus be useful in more accurately classifying cases of False Paraphrase. In future work, we will build upon the strengths of the weighted dependency features to improve the classifier further.

We also argue that Paraphrase Classification be used as a means to validate whether or not, in the context of abstract-like summarisation, a generated paraphrase reflected the source material. For this purpose, performance of precision and recall of the False Paraphrase classification seems more important, as we do not want to waste the end user's time by generation misleading information.

7 Acknowledgements

This work was funded by the Centre for Language Technology at Macquarie University and the CSIRO Information and Communication Technology Centre. We would like to thank the research groups of both organisations as well as the anonymous reviewers for useful comments and

Features	Acc.	C1-prec.	C1-recall	C1-Fmeas.	C2-prec	C2-recall	C2-Fmeas.
Best Features	75.63	0.70	0.46	0.55	0.77	0.90	0.83
All Features	71.00	0.55	0.71	0.61	0.83	0.72	0.77

Table 4: Classification performance of the best feature vector found and the feature vector including weighted dependency overlap. C1 denotes False Paraphrase pairs, C2 denotes True Paraphrase pairs.

feedback.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, Michigan.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003: Main Proceedings*.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, San Francisco.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, Michigan.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Hal Daumé and Daniel Marcu. 2005. Bayesian multi-document summarization at mse. In *Proceedings of the Workshop on Multilingual Summarization Evaluation (MSE)*, Ann Arbor, MI.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of Coling 2004*, Geneva, Switzerland.

Mark Dras. 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University, Australia.

Andrew Finch, Young Sook Hwang, and Eiichiro Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of The 3rd International Workshop on Paraphrasing (IWP2005)*.

Eduard Hovy, Chin-Yew Lin, and Liang Zhou. 2005. Evaluating duc 2005 using basic elements. In *Proceedings of Document Understanding Conference (DUC 2005)*, Vancouver, B.C. Canada.

Milen Kouylekov and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the PASCAL Challenges Workshop: Recognising Textual Entailment Challenge*.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *The Proceedings of the European Workshop on Natural Language Generation 2005*, Aberdeen, Scotland.

Diego Mollá. 2003. Towards semantic-based overlap measures for question answering. In *Proceedings of the Australasian Language Technology Workshop (ALTW 2003)*, Melbourne.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, July.

V. Punyakanok, D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI & Math*.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2005. Towards statistical paraphrase generation: preliminary evaluations of grammaticality. In *Proceedings of The 3rd International Workshop on Paraphrasing (IWP2005)*, Jeju Island, South Korea.

Ian H. Witten and Eibe Frank. 2000. *Data mining: practical machine learning tools and techniques with Java implementations*. San Francisco, CA, USA.

Etienne Denoual Yves Lepage. 2005. Automatic generation of paraphrases to be used as translation references in objective evaluation measures of machine translation. In *Proceedings of The 3rd International Workshop on Paraphrasing (IWP 2005)*.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *Proceedings of the Australasian Language Technology Workshop 2005*, Sydney, Australia.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6).

Verb Sense Disambiguation Using Selectional Preferences Extracted with a State-of-the-art Semantic Role Labeler

Patrick Ye and **Timothy Baldwin**
Dept. of Computer Science and Software Engineering
University of Melbourne
Victoria 3010 Australia

{jingy,tim}@csse.unimelb.edu.au

Abstract

This paper investigates whether multi-semantic-role (MSR) based selectional preferences can be used to improve the performance of supervised verb sense disambiguation. Unlike conventional selectional preferences which are extracted from parse trees based on hand-crafted rules, and only include the direct subject or the direct object of the verbs, the MSR based selectional preferences to be presented in this paper are extracted from the output of a state-of-the-art semantic role labeler and incorporate a much richer set of semantic roles. The performance of the MSR based selectional preferences is evaluated on two distinct datasets: the verbs from the lexical sample task of SENSEVAL-2, and the verbs from a movie script corpus. We show that the MSR based features can indeed improve the performance of verb sense disambiguation.

1 Introduction

Verb sense disambiguation (VSD) is the task of examining verbs in a given context and specifying exactly which sense of each verb is the most appropriate in that context. VSD is a subtask of word sense disambiguation (WSD). Given a verb sense inventory and a set of verb senses, VSD is essentially a classification task (Yarowsky, 2000).

VSD has not received much attention in the literature of WSD until recently. Most of WSD systems disambiguate verbs in the same way as nouns using mostly collocation based features, and this has led to rather poor VSD performance on corpora such as SENSEVAL-2. One useful but often ignored source of disambiguation information for verbs is the patterns of verb-argument structures. In this paper, we will attempt to capture these patterns through the use of selectional preferences.

In general, selectional preferences describe the phenomenon that predicating words such as verbs and adjectives tend to favour a small number of **noun classes** for each of their arguments. For example, the verb *eat* ("take in solid food") tends to select nouns from to the ANIMATED_THING class as its EATER role, and nouns from the EDIBLE class as its EATEE role.

However, it is possible to extend the concept of selectional preferences to include nouns which function as adjuncts to predicating words. For example, the verb *hit* in the sentence *I hit him with my fists* stands for "deal a blow to, either with the hand or with an instrument", but in the sentence *I hit him with a car*, it stands for "to collide with", with the only difference between the two instances of *hit* being their manner modifiers. Intuitively, the inclusion of verb adjuncts can enrich the semantic roles (SRs) and provide additional disambiguation information for verbs. Therefore, in the rest of this paper, the concept of "semantic role" will include both the arguments and adjuncts of verbs.

All the selectional preference based WSD systems to date have only used the subject and direct object of verbs as semantic roles, extracting the necessary argument structure via hand-crafted heuristics (Resnik, 1997; McCarthy and Carroll, 2003, inter alia). As a result, it is difficult to extend the selectional preferences to anything else. However, with recent progress in Semantic Role Labelling (SRL) technology, it is now possible to obtain additional semantic roles such as the indirect object of ditransitive verbs and the locational, temporal and manner adjuncts.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 139–148.

Given the lack of research in VSD using multi-semantic-role based selectional preferences, the main contribution of the work presented in this paper is to show that it is possible to use the multi-semantic-role based selectional preferences extracted using the current state-of-the-art SRL systems to achieve a certain level of improvement for verb VSD. We also give detailed descriptions for all the features, feature selection algorithms and the tuning of the machine learner parameters that we have used in the construction of our VSD system, so that our system can be easily reproduced.

The remainder of this paper is organized as follows: we first introduce the framework for constructing and evaluating the VSD systems in Section 2; we then give detailed descriptions of all the feature types that we experimented with during our research in Section 3; Section 4 introduces the two datasets used to train and evaluate the features; the feature selection algorithms are presented in Section 5; the results of our experiments are presented in Section 6; and finally we conclude in Section 7.

2 VSD Framework

There are three components in our VSD Framework: extraction of the disambiguation features from the input text (feature extraction), selection of the best disambiguation features with respect to unknown data (feature selection), and the tuning of the machine learner's parameters. Since feature extraction is explained in detail in Section 3, we will only disscuss the other two components here.

Within our framework, feature selection is performed only on the training set. We first use the feature selection algorithms described in Section 5 to generate different feature sets, which are used to generate separate datasets. We then perform cross validation (CV) on each dataset, and the feature set with the best performance is chosen as the final feature set.

The machine learning algorithm used in our study is Maxium Entropy (MaxEnt: Berger et al. (1996)[1]). MaxEnt classifiers work by modelling the probability distribution of labels with respect to disambiguation features, the distribution of which is commonly smoothed based on a Gaussian prior. Different values for the Gaussian prior often lead to significant differences in the classification of new data, motivating us to include the tuning of the Gaussian prior in VSD framework.[2]

The tuning of the Gaussian prior is performed in conjunction with the feature selection. The CV for each feature set is performed multiple times, each time with a different parameterisation of the Gaussian prior. Therefore, the final classifier incorporates the best combination of feature set and parameterisation of the Gaussian prior for the given dataset.

3 Features Types

Since selectional preference based WSD features and general WSD features are not mutually exclusive of each other, and it would be less convincing to evaluate the impact of selectional preference based features without a baseline derived from general WSD features, we decided to include a number of general WSD features in our experiments. The sources of these features include: Part-of-Speech tags extracted using a tagger described in Gimnez and Mrquez (2004); parse trees extracted using the Charniak Parser (Charniak, 2000); chunking information extracted using a statistical chunker trained on the Brown Corpus and the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1993); and named entities extracted using the system described in Cohn et al. (2005).

3.1 General WSD Features

There are 3 broad types of general WSD features: n-gram based features of surrounding words and WordNet noun synsets, parse-tree-based syntactic features, and non-parse-tree based syntactic features.

3.1.1 N-gram based features

The following n-gram based features have been experimented with:

Bag of Words Lemmatized open class words in the entire sentence of the target verb. Words that occur multiple times are only counted once.

Bag of Synsets The WordNet (Fellbaum, 1998) synsets for all the open class words in the entire

[1]We used Zhang Le's implementation which is available for download at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

[2]We experimented with the following settings for the standard deviation (with a mean of 0) of the Gaussian prior in all of our experiments: 0.1, 0.5, 1.0, 5.0, 10.0, 50.0, 100.0, 500.0, 1000.0.

sentence; hypernyms for nouns and verbs are also included.

Bag of POS Tags The POS tag of each word within a window of 5 words surrounding the target verb, paired with its relative position and treated as a separate binary feature.

Bag of Chunk Tags The chunk tags (in Inside-Outside-Beginning (IOB) format: Tjong Kim Sang and Veenstra (1999)) surrounding and including the target verb within a window of 5 words, paired with their relative positions.

Bag of Chunk Types The chunk types (e.g. NP, VP) surrounding and including the target verb within a window of 5 words.

Bag of Named Entities The named entities in the entire sentence of the target verb.

Left Words Each lemmatized word paired with its relative position to the left of the target verb within a predefined window.

Right Words Each lemmatized word paired with its relative position to the right of the target verb within a predefined window.

Surrounding Words The union of **Left Words** and **Right Words** features.

Left Words with Binary Relative Position Each lemmatized word and its position[3] to the left of the target verb within a predefined window.

Right Words with Binary Relative Position Each lemmatized word and its binary position to the right of the target verb within a predefined window.

Surrounding Words with Binary Relative Position The union of **Left Words with Binary Position** and **Right Words with Binary Position** features.

Left POS-tags The POS tag of each word to the left of the target verb within a predefined window, paired with its relative position.

Right POS-tags The POS tag of each word to the right of the target verb within a predefined window is paired with its relative position.

Surrounding POS Tags The Union of the **Left POS-tags** and **Right POS-tags** features.

It may seem redundant that for the same window size, the Surrounding-Words (POS) features are the union of the Left-Words (POS) features and the Right-Words (POS) features. However, this redundancy of features makes it convenient to investigate the disambiguation effectiveness of the word collocations before and after the target verb, as well as the syntactic pattern before and after the target verb. Furthermore, we have also experimented with different window sizes for the Surrounding-Words (POS), Left-Words (POS) and Right-Words (POS) to determine the most appropriate window size.[4]

3.1.2 Parse tree based features

The parse tree based syntactic features are inspired by research on verb subcategorization acquisition such as Korhonen and Preiss (2003), and are intended to capture the differences in syntactic patterns of the different senses of the same verb. Given the position of the target verb v in the parse tree, the basic form of the corresponding parse tree feature is just the list of nodes of v's siblings in the tree. Figure 1 shows the parse tree for a sentence containing the ambiguous verb *call*. Given the position of the target verb *called* in the parse tree, the basic form of the features that can be created will be (NP, PP). However, there are 3 additional types of variations that can be applied to the basic features. The first variation is to include the relative positions of the sibling node types as part of the feature: this variation will change the basic feature for *call* to $\{(1, NP), (2, PP)\}$. The second variation is to include the binary relative direction of the siblings to the target verb as part of the feature, i.e. is the sibling to the left or right of the target verb: this variation will change the basic feature for *call* to $\{(right, NP), (right, PP)\}$. The third variation is to include the parent node type as part of the sibling node type to add more information in the syntactic pattern. Figure 2 shows what the original parse tree looks like when every nonterminal is additionally annotated with its parent type. Since the third variation is compatible with the first two variations, we decided to combine them to create the following parse tree

[3] All words to the left of the target verb are given the "left" position, and all the to the right of the target verb are given the "right" position.

[4] In the ranking based evaluation method described in Section 5, only the Surrounding-Words (POS) feature types with the largest window size are used.

based features:

Type 1 Original sibling node types (zero level of parent node annotated) with relative positions.

Type 2 Original sibling node types with relative binary directions.

Type 3 One level of parent-node-annotated sibling node types with relative positions. Using the *call* example, this feature will be $\{(1, \textit{VP-NP}), (2, \textit{VP-PP})\}$.

Type 4 One level of parent-node-annotated sibling node types with relative binary directions. Using the *call* example, this feature will be $\{(right, \textit{VP-NP}), (right, \textit{VP-PP})\}$.

Type 5 Two levels of parent-node-annotated sibling node types with relative positions.

Type 6 Two levels of parent-node-annotated sibling node types with relative binary directions.

On top of the 6 types of purely syntactic based parse tree features, there is an additional type of parse tree based feature designed to capture the **verb-argument structure** of the target verb. The type of features only cover four particular types of parse tree nodes which are immediately after the pre-terminal node of the target verb. The four types of parse tree nodes are: ADVP, NP, PP and clausal nodes.

For an ADVP node, we extract its head adverb H_ADV, and treat the tuple of (**ADVP**, H_ADV) as a separate binary feature.

For an NP node, we first extract its head nouns, then replace each head noun with its WordNet synsets and the hypernyms of these synsets, and treat each of these synsets as a separate binary feature. In order to cover the cases in which the head noun of a NP node is a quantity noun, e.g. *a glass of water*, the head nouns of PPs attached to the NP nodes are also included as head nouns. Furthermore, head nouns which are named entities identified by the system described in Cohn et al. (2005) are replaced by appropriate WordNet synsets.

For a PP node, we first extract its head preposition, then we extract the head noun synsets in the same way as the NP node, and finally we combine each synset with the head preposition to form a separate binary feature.

For a clausal node which is an SBAR, we extract the list of node types of its direct children and arrange them in their original order, and treat this list as a single binary feature.

For a clausal node which is not an SBAR, but has a single non-terminal child node, we first extract the type of this child node, then we extract the list of node types for the children of this child node. The tuple of (child-node-type, list-of-grandchildren-node-types) is then treated as a separate binary feature.

3.1.3 Non-parse tree based syntactic features

There are 3 types of non-parse-tree based syntactic features:

Voice of the verb The voice of the target verb (active or passive).

Quotatives Verbs that appear in directly quoted speech have a greater likelihood of occurring in the imperative and losing the surface subject, e.g. *"Call the police!"*. We therefore include this as a standalone feature.

Additional Chunk based features A number of additional chunk based features are also used to capture the phrase level localized syntactic and collocation patterns from the context to the right of the target verb within a window of between 3 and 10 chunks. For example, using a window of 7, for the verb *kick* in the sentence: *[I/PRP]$_{NP}$ [***kicked***/VBD]$_{VP}$ [him/PRP]$_{NP}$ [out/IN of/IN]$_{PP}$ [the/DT door/NN]$_{NP}$ [through/IN]$_{PP}$ [which/WDT]$_{NP}$ [he/PRP]$_{NP}$ [came/VBD]$_{VP}$*, the first 7 chunks after the chunk that contains **kick** will be used for feature extraction. These additional chunk based features are:

Chunk-type-sequence The concatenation of all the relevant chunk types. For example, using the above *kick* example, this feature will be **NP_PP_NP_PP_NP_NP_VP**.

Regular expression (RE) representation of the chunk types The consecutive identical chunk types in the **Chunk-type-sequence** feature merged into a single symbol. For example, the chunk-type-sequence of **NP_PP_NP_PP_NP_NP_VP** will be represented as **NP_PP_NP_PP_NP+_VP**.

First word of each chunk with the chunk type The list that contains the first word of each chunk will be treated as a separate binary feature. With the *kick* example, this feature would be **him_out_the_through_which_he_came**.

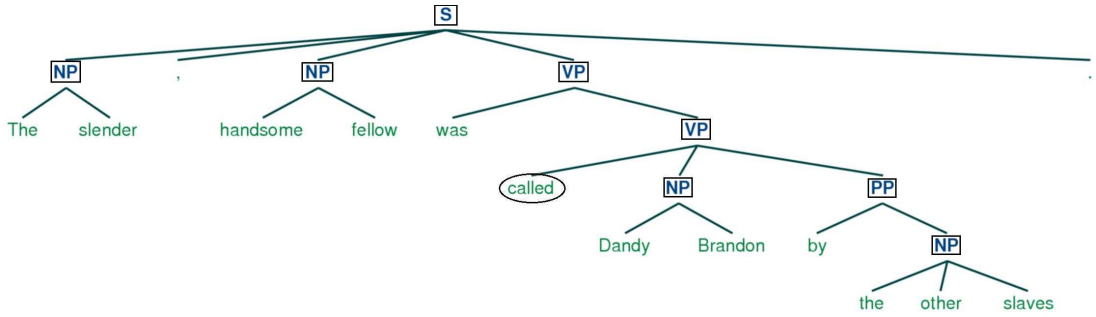

Figure 1: Basic parse tree feature example

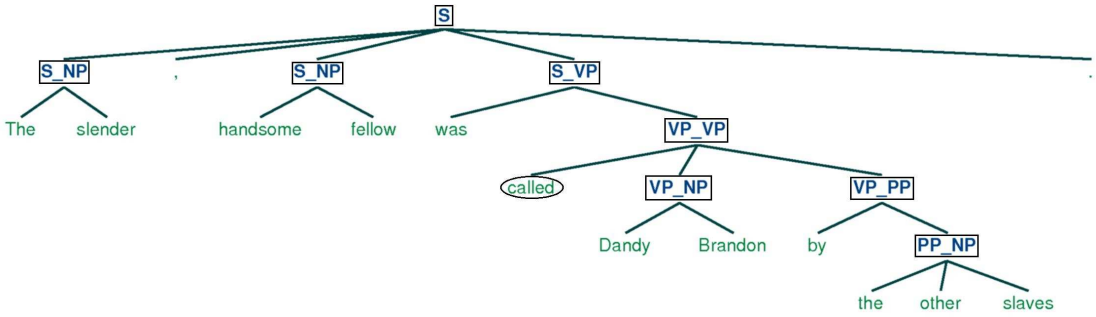

Figure 2: One level parent annotated parse tree

Last word of each chunk with the chunk type The list that contains the last word of each chunk will be treated as a separate feature. With the *kick* example, this feature would be **him_of_door_through_which_he_came**.

First POS tag of each chunk with the chunk type The list that contains the first POS tag of each chunk will be treated as a separate feature. With the *kick* example, this feature would be **PRP_IN_DT_IN_WDT_PRP_VBD**.

Last POS tag of each chunk with the chunk type The list that contains the last POS tag of each chunk will be treated as a separate feature. With the *kick* example, this feature would be **PRP_IN_NN_IN_WDT_PRP_VBD**.

Chunk-type-sensitive combinations This feature is created by merging the chunk types and some additional information associated with the chunks. If a chunk is a PP, then the head preposition will also be part of the feature; if a chunk is an NP and the head word is a question word (*who, what, when, how, where* or *why*), the head word itself will be part of the feature, but if the head word is not a question word, its POS tag will be part of the feature; if a chunk is a VP, then the POS tag of the head verb will be part of the feature. Using the above *kick* example, this feature will be: **(NP, PRP)_(PP, out)_(NP, NN)_(PP, through)_(NP, which)_(NP, PRP)_(VP, VBD)**.

3.2 Selectional Preference Based Features

We used the ASSERT[5] system (Hacioglu et al., 2004) to extract the semantic roles from the target sentences. The following selectional preference based features have been experimented with:

WordNet synsets of the head nouns of the SRs For each semantic role, the WordNet synsets of its head noun, paired with the corresponding semantic role.

Semantic role's relative positions These features are designed to capture the syntactic patterns of the target verb and its semantic roles. The relative position is set up such that the first semantic role to the left of the target verb will be given the position of -1, and the first one to the right will be given the position of $+1$.

Lemmatized head noun of each semantic role Similar to the synset features, each semantic role is also paired with its head noun.

[5]We used version 1.4b of this system which can be downloaded from `http://oak.colorado.edu/assert/`

Preposition of the adjunct semantic role If there is an adjunct semantic role which is also a prepositional phrase, the preposition is also paired with the semantic role.

4 Evaluation Datasets

We used two datasets based on distinct text genres to examine the effectiveness of the multi-semantic-role based selectional preferences: the verb dataset from the English lexical-sample subtask of SENSEVAL-2, and a manually sense tagged movie script corpus (MSC).[6] The SENSEVAL-2 data contains 28 polysemous verbs, and 3564 training instances. The movie script corpus contains 8 sense labelled verbs and 538 training instances. Table 1 outlines the composition of the movie script corpus dataset.

The sense tagged movie script corpus is important because it is an integral part of a broader NLP project which aims to generate computer animation by interpreting movie scripts. Since most of the actions in movie scripts are described by verbs, we believe it is necessary to investigate whether knowing the senses of the verbs can improve the accuracy of the animation generation.

The movie script corpus was hand-tagged by two annotators according to WordNet 2.0 senses, and the differences in the annotation were arbitrated by a third judge. Compared with the SENSEVAL-2 data, which comes from the Brown Corpus, the sentences in the movie script corpus tend to be shorter because they describe certain actions to be performed in the movie. Example sentences from this corpus include the following:

1. A rubber dart *hits* the glass and drops into a trash can next to the door .
2. Neo slowly sets down his duffel bag and *throws* open his coat, revealing an arsenal of guns, knives, and grenades slung from a climbing harness.
3. Morpheus tries to *look* not sad.

5 Feature Selection Algorithms

It has been observed that combining certain features together can lead to a decrease in classification accuracy, hence a feature selection process is deemed necessary. Due to the high numbers of individual binary features and feature types, it would be impractical to generate all possible combinations of the individual features or feature types. Therefore, we propose two automatic feature selection algorithms here: the individual feature ranking algorithm and feature type coverage algorithm.

5.1 Individual Feature Ranking Algorithm

This algorithm is based on the work of Baldwin and Bond (2003) and works by first calculating the information gain (IG), gain ratio (GR) and Chi-square statistics (Chi) for each binary feature as 3 separate scores. Then, each score is separately used to rank the features in a way such that the greater the score, the higher the rank. Features which have the same value for a particular score are given the same rank. Once individual ranks have been determined for each feature, the ranks themselves are summed up and used as a new score which is then used to re-rank all the features one last time. This final ranking will be used to perform the feature selection.

Once the final ranking of the features has been calculated, we then generate separate feature sets using the top N percent ranked features, where N ranges from 0 to 100 with an increment of 5.[7] We evaluate these feature sets in Section 2.

5.1.1 Feature Type Coverage Algorithm

The aim of this algorithm is to use the minimal number of feature **types** to generate the best performance. It works in the following way. First, we assign a unique ID to every training instance in the original training set. We then create a separate dataset for each individual feature type (e.g. **Bag Of Words**, **Left Words**, etc.) and evaluate them as per Section 2. Since the single-feature-typed datasets are all created from the same original training set, we can propagate the IDs of the original training instances to the testing instances in the held-out sets of the single-feature-typed datasets. Furthermore, as the CVs are stratified, we can calculate the accuracy of each feature type with respect to each training instance. For example, suppose the $10th$ training instance for the verb *hit* was correctly classified in the held-out set by the classifier trained with only the **verb-argument structure** features, then the accuracy

[6] The entire MSC dataset contains more than 250 movie scripts, but due to limited resources, only 3 scripts were sense tagged, and only 8 high frequency and highly polysemous verbs were chosen for this study.

[7] The top 0% feature set is replaced by the majority-class heuristic

Verb	Freq. in Sense Tagged Corpus	Freq. in entire corpus	No. of Senses in Sense Tagged Corpus	Majority Class Ratio	Inter Annotator Agreement
hit	32	1770	6	.438	.828
lift	31	1051	4	.548	.807
look	164	19480	5	.884	.963
pull	79	5734	7	.410	.744
stop	43	3025	4	.524	.917
take	54	8277	14	.370	.679
throw	29	1940	6	.379	.724
turn	106	8982	10	.736	.953

Table 1: Movie script corpus details

of **verb-argument structure** feature type on this particular training instance would be 1.0. With these training-instance-specific accuracies, we define the **coverage** of a feature type as a mapping from training instance IDs to their individual accuracies. We also use the sum of the accuracies of the individual data instance to assess the overall coverage of particular feature type with respect to a training set.

In order to assess the coverage of combined feature types, we define an additional procedure called *combine* for merging two coverages together to produce a new coverage. The details of this algorithm are shown in Algorithm 1.

On top of the coverages, we also calculate the **applicability** of each feature type as the percentage of held-out instances for which one or more features of that particular type can be extracted. The final phase of the feature selection algorithm is a process of greedily combining the coverages of feature types together until the coverage plateaus to a local maximum. This process is described in Algorithm 2.

6 Results and Discussion

For the SENSEVAL-2 data, the feature selection algorithms were applied on the 10-fold cross validated training set and the chosen features type and the Gaussian smoothing parameters were applied to the test set. For the movie script corpus, the feature selection algorithms were applied to 5-fold nested cross validation of the entire dataset. The final cross validation results are reported here.[8] Furthermore, in order to measure the usefulness of the feature selection algorithm, we have also included results obtained using "Oracled" feature sets and Gaussian prior values which were tuned with respect to the test data. More specifically, we collected the following information for our evaluation:

Fully Oracled Using both oracled feature sets and oracled Gaussian prior values.

FS Oracled Using oracled feature sets but automatically selected Gaussian prior values.

Maxent Oracled Using automatically selected feature sets and oracled Gaussian prior values.

Fully Auto. Using both automatically selected feature sets and Gaussian prior values.

All Features Including all features and using automatically selected Gaussian prior values.

Tables 2 and 3 respectively summarize the evaluation results on the datasets with and without SRL features.[9]

It can be observed that the impact of the feature selection algorithms on the SENSEVAL-2 dataset is similar to that on the MSC dataset. The feature ranking algorithm seems to perform noticeably worse than having no feature selection at all, but the coverage algorithm seems perform mostly better. This shows that feature selection can be a useful process irrespective of the corpus.

The disappointing performance of the feature ranking algorithm on both datasets is caused by the mismatch between the training and the testing data. Recall that this algorithm works by selecting the top N percent of features in terms of their disambiguation power. Since the feature selection is only performed on the training set, the chosen features could be absent from the test set or have different distributions with respect to the verb senses. Verb-by-verb analysis for this algorithm revealed

[8] In nested cross validation, the training set of each fold is further divided into a sub-training and sub-held-out set, and the feature selection and Gaussian smoothing parameters for the proper held-out sets are tuned on a fold-by-fold basis.

[9] The majority class baseline for the MSC dataset is gathered from the primary held-out sets of the nested CV, therefore it is potentially different to the majority class of the entire MSC dataset.

Algorithm 1 The algorithm of *combine*, which merges two coverages to produce a new coverage.

Let I be the set of IDs of the training instances
Let $Coverage_1$ and $Coverage_2$ be the two coverages to be merged
Let *NewCoverage* be the combined coverage of $Coverage_1$ and $Coverage_2$
for $i \in I$ **do**
    $NewCoverage[i] = \max(Coverage_1[i], Coverage_2[i])$
end for
Return *NewCoverage*

Algorithm 2 The incremental process of determining the final set of feature types

Let I be the set of IDs of the training instances
$CurrentCoverage = \{(i \rightarrow 0.0) | i \in I\}$
Let F be the set of feature types
$Combine(coverage_1, coverage_2) = \{(i \rightarrow max(coverage_1[i], coverage_2[i])) | i \in I\}$
while True **do**
    $NCs \leftarrow \{\}$ ▷ Initialize a temporary list to hold the new combined coverages
    for $f_i \in F$ **do**
        $NewCoverage_i \leftarrow Combine(CurrentCoverage, Coverage_{f_i})$
        Add $NewCoverage_i$ to *NCs*
    end for
    Let $NewCoverage^\star \in NCs$ be the one with highest overall coverage. ▷ Break tie with the lowest applicability.
    if *CurrentCoverage* has the same overall coverage as $NewCoverage^\star$ **then**
        Break
    end if
    $CurrentCoverage \leftarrow NewCoverage^\star$
end while

Dataset	Individual Feature ranking					Feat. Coverage		Majority
	Fully Oracled	FS Oracled	Maxent Oracled	Fully Auto.	All Features	Maxent Oracled	Fully Auto.	Class Baseline
SENSEVAL-2	.623	.615	.556	.540	.574	.588	.583	.396
MSC	.774	.743	.602	.577	.690	.712	.712*	.617

Table 2: Disambiguation accuracies with SRL features (* indicates significantly higher performance than the all features baseline (paired t-test, $p > 90\%$))

Dataset	Individual Feature ranking					Feat. Coverage		Majority
	Fully Oracled	FS Oracled	Maxent Oracled	Fully Auto.	All Features	Maxent Oracled	Fully Auto.	Class Baseline
SENSEVAL-2	.606	.595	.544	.529	.558	.583	.576	.396
MSC	.780	.747	.554	.532	.714	.721	.693	.617

Table 3: Disambiguation accuracies without SRL features

that for most verbs, there was very little correlation between the training data and the test data in terms of the value of N verses the disambiguation performance. This is why this algorithm consistently selected suboptimal feature sets which resulted in the poor performance. This mismatching problem is especially severe for the MSC corpus which contains many verb senses that occur only once.

On the other hand, the performance of the coverage based algorithm tends to be similar to or better than that of using all the features. Recall that this algorithm selects feature types in such a way that the coverage of the final feature set is maximized. As a result, even feature types which performed poorly on the training set may be selected as long as they performed well on some of the training instances that other features performed poorly on. Therefore, the features chosen by this algorithm are less likely to overfit the training data.

Overall, it can be observed that for both datasets, most of our automatic classifiers outperform the majority class baseline, which is very encouraging. For the SENSEVAL-2 dataset, the classifiers with SRL features consistently outperform those without. However, for the MSC dataset, the results of the nested cross validation showed that the performance of the automatic classifiers with the SRL features does not consistently outperform those without the SRL features; the oracled classifiers constructed without the SRL features actually consistently outperformed those with the SRL features.

The differences between the results obtained from the two datasets make it difficult to conclude categorically whether SRL can indeed help VSD. However, a closer examination of the datasets reveals that errors in the output of the semantic role labeler, the intransitivity of the verbs, unresolved anaphors, and verbal multiword expressions (verbal MWEs) are the main reasons for the lack of positive contribution from the SRL features.

Most of the verbs on which SRL features performed poorly are intransitive, which means that only one argument type semantic role is available – the subject or the agent role, which mostly occurs to the left of the verb. However, the feature ranking algorithm showed that most of the useful features occur to the **right** of the verb, which is why SRL features tend to perform poorly on intransitive verbs.

The unresolved anaphors also limited the effectiveness of SRL features because they carry almost no disambiguation information, no matter which semantic role they take, and they occur in a significant number of sentences. The anaphor problem is slightly more pronounced in the movie script corpus, because its texts tend to describe consecutive actions performed by the same actor(s) and involving the same objects in the scene, and therefore anaphors tend to occur more frequently.

Verbal MWEs such as *take off* are not detected as a single lexical item, and the verbs themselves tend to have no suitable sense as far as WordNet is concerned. However, they often occur more than once in the data, and since the annotators were forced always to pick at least one sense, these expressions tend to end up as noise in the data.

Finally, it is also possible that the lack of training data in the MSC corpus contributed to the poor performance, since almost every verb in the MSC corpus contains two or more senses which occur less than twice.

7 Conclusions and Future Work

In this paper, we have presented our research on using multi-semantic-role based selectional preferences obtained from a state-of-the-art semantic role labeler. We have shown that this particular type of selectional preferences can indeed improve the performance of verb sense disambiguation. However, this improvement still depends on the performance of the semantic role labeler, the transitivity of the verbs, the resolution of anaphors, and the identification of verbal MWEs.

In future research, we hope to focus on integrating more competent anaphora resolution systems and verbal MWE detectors into our existing VSD framework, and investigating how to mitigate the errors in the semantic role labeler.

Acknowledgements

We would like to thank the three anonymous reviewers for their valuable input on this research. The research in this paper has been supported by the Australian Research Council through Discovery Project grant number DP0663879, and also NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation.

References

Timothy Baldwin and Francis Bond. 2003. Learning the countability of english nouns from corpus data. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 73–80, Sapporo, Japan.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Eugene Charniak. 2000. Maximum-entropy-inspired parser. In *Proceedings of the First Conference on North American Chapter of the ACL*, pages 132–139, San Francisco, USA.

Trevor Cohn, Andrew Smith, and Miles Osborne. 2005. Scaling conditional random fields using error-correcting codes. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 10–17, Ann Arbor, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Jess Gimnez and Llus Mrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46, Lisbon, Portugal.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proc. of the 8th Conference on Natural Language Learning (CoNLL-2004)*, pages 110–113, Boston, USA.

Anna Korhonen and Judita Preiss. 2003. Improving subcategorization acquisition using word sense disambiguation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 48–55, Sapporo, Japan.

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

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654, December.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics, Why, What and How?*, pages 52–57, Washington, USA.

Erik Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99: Ninth Conference of the European Chapter of the ACL*, pages 173–179, Bergen, Norway.

David Yarowsky, 2000. *Handbook of Natural Language Processing*, chapter 26. Marcel Dekker.

This Phrase-Based SMT System is Out of Order: Generalised Word Reordering in 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

Many natural language processes have some degree of preprocessing of data: tokenisation, stemming and so on. In the domain of Statistical Machine Translation it has been shown that word reordering as a preprocessing step can help the translation process.

Recently, hand-written rules for reordering in German–English translation have shown good results, but this is clearly a labour-intensive and language pair-specific approach. Two possible sources of the observed improvement are that (1) the reordering explicitly matches the syntax of the source language more closely to that of the target language, or that (2) it fits the data better to the mechanisms of phrasal SMT; but it is not clear which. In this paper, we apply a general principle based on dependency distance minimisation to produce reorderings. Our language-independent approach achieves half of the improvement of a reimplementation of the hand-crafted approach, and suggests that reason (2) is a possible explanation for why that reordering approach works.

Help you I can, yes.
Jedi Master Yoda

1 Introduction

Preprocessing is an essential step in Natural Language applications. Reordering of words on a sentence level as a more extensive step for preprocessing has succeeded in improving results in Statistical Machine Translation (SMT). Here, both in the training and in the decoding phase, sentences in the source language are reordered before being processed.

This reordering can be done based on rules over word alignment learnt statistically; (Costa-Juss and Donollosa, 2006), for example, describe such a system. 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 syntactical knowledge of the language is used.

Xia and McCord (2004) do use syntactical knowledge; they use pattern learning in their reordering system. In their work in the training phase they parse and align sentences 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 (that is, decoding while roughly keeping the same order in the target language as in the source language — reordering done within phrases, for example, is an exception).

Syntactically motivated rules are also used in reordering models. In Collins et al. (2005) six hand-written rules for reordering source sentences are defined. These rules operate on the output of an automatic parser. The reordering rules however are language-pair (German-English) specific and hand-written.

We want to extend this idea of word reordering as preprocessing by investigating whether we can find a general underlying principle for reordering, to avoid either thousands of patterns, or arguably arbitrary hand-written rules to do this. To do this, we note that a common characteristic of the Collins et al. (2005) rules is that they reduce the distances of a certain class of long-distance

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 149–156.

dependencies in German with respect to English. We note that minimising of dependency distances is a general principle appearing in a number of guises in psycholinguistics, for example the work of Hawkins (1990). In this paper we exploit this idea to develop one general syntactically motivated reordering rule subsuming those of Collins et al. (2005).

This approach also helps us to tease out the source of translation improvement: whether it is because the reordering matches more closely the syntax of the target language, or because fits the data better to the mechanisms of phrasal SMT.

The article is structured as follows: in section 2 we give some background on previous work of word reordering as preprocessing, on general word ordering in languages and on how we can combine those. In section 3 we describe the general rule for reordering and our algorithm. Section 4 describes our experimental setup and section 5 presents the results of our idea; this leads to discussion in section 6 and a conclusion in section 7.

2 Reordering Motivation

It has been shown that reordering as a preprocessing step can lead to improved results. In this section we look in a little more detail at an existing reordering algorithm. We then look at some general characteristics in word ordering in the field of psycholinguistics and propose an idea for using that information for word reordering.

2.1 Clause Restructuring

Collins et al. (2005) describe reordering based on a dependency parse of the source sentence. In their approach they have defined six hand-written rules for reordering German sentences. In brief, German sentences often have the tensed verb in second position; infinitives, participles and separable verb particles occur at the end of the sentence. These six reordering rules are applied sequentially to the German sentence, which is their source language. Three of their rules reorder verbs in the German language, and one rule reorders verb particles. The other two rules reorder the subject and put the German word used in negation in a more English position. All their rules are designed to match English word ordering as much as possible. Their approach shows that adding knowledge about syntactic structure can significantly improve the performance of an existing state-of-the-art statistical machine translation system.

2.2 Word Order Tendencies in Languages

In the field of psycholinguistics Hawkins (1990) argues that the word order in languages is based on certain rules, imposed by the human parsing mechanism. Therefore languages tend to favour some word orderings over others. He uses this to explain, for example, the universal occurrence of postposed sentential direct objects in Verb-Object-Subject (VOS) languages.

In his work, he argues that one of the main reasons for having certain word orders is that we as humans try to minimise certain distances between words, so that the sentence is easier to process. In particular the distance between a head and its dependents is important. An example of this is the English Heavy Noun Phrase shift. Take the following two sentence variants:

1. I give <*NP*> back
2. I give back <*NP*>

Whether sentence 1 or 2 is favourable, or even acceptable, depends on the size (heaviness) of the NP. If the NP is *it* only 1 is acceptable. When the NP is medium-sized, like *the book*, both are fine, but the longer the NP gets the more favourable 2 gets, until native speakers will say 1 is not acceptable anymore. Hawkins explains this by using head-dependent distances. In this example *give* is the head in the sentence; if the NP is short, both the NP and *back* are closely positioned to the head. The longer the NP gets the further away *back* is pushed. The theory is that languages tend to minimise the distance, so if the NP gets too long, we prefer 2 over 1, because we want to have *back* close to its head *give*.

2.3 Reordering Based on Minimising Dependency Distances

Regarding the work of Collins et al., we suggest two possible sources for the improvement obtained.

Match target language word order Although most decoders are capable of generating words in a different order than the source language, usually only simple models are used for this reordering. In Pharaoh (Koehn, 2004), for example, every word reordering between languages is penalised and only the language model can encourage a different order. If we can match the word order of the target language to a certain degree, we might expect an increase in translation quality, because we already have more explicitly used information of what the new word ordering should be.

Fitting phrase length The achievement of Phrase-Based SMT (PSMT) (Koehn et al., 2003) was to combine different words into one phrase and treat them as one unit. Yet PSMT only manages to do this if the words all fit together in the same phrase-window. If in a language a pair of words having a dependency relation are further apart, PSMT fails to pick this up: for example, verb particles in German which are distant from the governing verb. If we can identify these long distance dependencies and move these words together into the span of one phrase, PSMT can actually pick up on this and treat it as one unit. This means also that sentences not reordered can have a better translation, because the phrases present in that sentence might have been seen (more) before.

The idea in this paper is to reorder based on a general principle of bringing dependencies closer to their heads. If this approach works, in not explicitly matching the word order of the target language, it suggests that fitting the phrase window is a contributor to the improvement shown by reordering. The approach also has the following attractive properties.

Generalisation To our knowledge previous reordering algorithms are not capable of reordering based on a general rule (unlike 'arbitrary' hand written language-pair specific rules (Collins et al., 2005) or thousands of different transformations (Xia and McCord, 2004)). If one is able to show that one general syntactically informed rule can lead to translation quality this is evidence in favour of the theory used explaining how languages themselves operate.

Explicitly using syntactic knowledge Although in the Machine Translation (MT) community it is still a controversial point, syntactical information of languages seems to be able to help in MT when applied correctly. Another example of this is the work of Quirk et al. (2005) where a dependency parser was used to learn certain translation phrases, in their work on "treelets". When we can show that reordering based on an elegant rule using syntactical language information can enhance translation quality, it is another small piece of evidence supporting the idea that syntactical information is useful for MT.

Starting point in search space Finally, most (P)SMT approaches are based on a huge search space which cannot be fully investigated. Usually hill climbing techniques are used to handle this large search space. Since hill climbing does not guarantee reaching global minima (error) or maxima (probability scoring) but rather probably gets 'stuck' in a local optimum, it is important to find a good starting point. Picking different starting points in the search space, by preprocessing the source language, in a way that fits the phrasal MT, can have an impact on overall quality.[1]

3 Minimal Dependency Reordering

Hawkins (1990) uses the distance in dependency relations to explain why certain word orderings are more favourable than others. If we want to make use of this information we need to define what these dependencies are and how we will reorder based on this information.

3.1 The Basic Algorithm

As in Collins et al. (2005), the reordering algorithm takes a dependency tree of the source sentence as input. For every node in this tree the linear distance, counted in tokens, between the node and its parent is stored. The distance for a node is defined as the closest distance to the head of that node or its children.

[1]This idea was suggested by Daniel Marcu in his invited talk at ACL2006, *Argmax Search in Natural Language Processing*, where he argues the importance of selecting a favourable starting point for search problems like these.

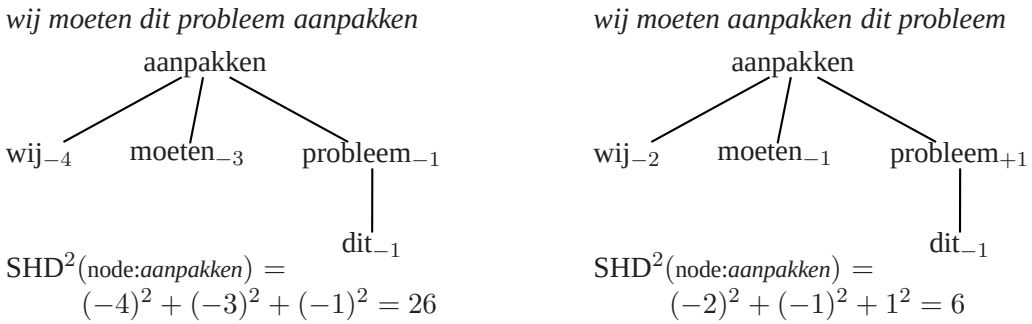

Figure 1: Reordering based on the sum of the head distances

To illustrate the algorithm of this section we present the following two examples:

1. Verb initial in VP:
normal order:
wij moeten dit probleem aanpakken
we must this problem tackle
reordered:
wij moeten aanpakken dit probleem
we must tackle this problem
reference:
we must tackle this particular problem

2. Verb Particle reordering:
normal order:
het parlement neemt de resolutie aan
the parliament takes the resolution over
reordered:
het parlement neemt aan de resolutie
the parliament takes over the resolution
reference:
parliament adopted the resolution

As an example of the calculation of distances, the left tree of Figure 1 is the dependency tree for the normal order for example 1; nodes are annotated with the distance from the word to its governor. Note that in example 1 *probleem* gets a value of 1, although the word itself is 2 away from its head; we are measuring the distance from this complete constituent and not this particular word.

Based on the distance of the different child nodes we want to define an optimal reordering and pick that one. This means we have to score all the different reorderings. We want a scoring measure to do this that ignores the sign of distances and gives higher weight to longer dependency distances. Thus, similar to various statistical optimisation algorithms such as Least Mean Squares, we calculate the square of the Sum of the Head Distances (SHD^2) for each node n, defined as:

$$\text{SHD}^2(n) = \sum_{c \in \textit{children}(n)} \textit{Distance}(c, n)^2$$

Every different ordering of children and head has a SHD^2 value; we are interested in minimising this value. We give an SHD^2 example in Figure 1.

We then reorder the children so that the SHD^2 score of a node is minimised. The righthand tree of Figure 1 gives an example. In example 1 we can see how the Dutch verb *aanpakken* is moved to the beginning of the verb phrase. In this example we match English word order, even though this is not an explicit goal of the metric. The second example does not match English word order as such, but in Dutch the verb *aannemen* was split into *aan* and *neemt* in the sentence. Our reordering places these two words together so that PSMT can pickup that this is actually one single unit. Note that the two examples also demonstrate two of Collins et al. (2005) hand-written rules. In fact, this principle subsumes all the examples given in that paper in the prototypical cases.

3.2 Selecting Minimal Reorderings

In implementing the algorithm, for each node with children we calculate the SHD^2 for all permutations (note that this is not computationally expensive as each node has only a small number of children). We select the collection of sibling orders with a minimal SHD^2. This is indeed a collection because different orderings can still have the same

SHD^2 value. In these cases we try to match the original sentence order much as possible.

There are many different way to calculate which permutation's order is closer to the original. We first count all the constituents which are on the other side of the head compared to the original. For all permutations from the list with the same minimal SHD^2 we count these jumps and keep the collection with the minimal number of jumps. Then we count the breaks where the new order deviates from an ascending ordering, when the constituents are labelled with their original position. For every constituent ordering c this Break Count (BC) is defined as:

$$\text{BC}(c) = \sum_{n=2}^{N} max(0, Pos_o(n-1) - Pos_o(n) + 1)$$

Here $Pos_o(x)$ is the original position of the xth constituent in the new order, and N is the length of the constituent c. As an example: if we have the five constituents $1, 2, 3, 4, 5$ which in a new order y are ordered $2, 1, 3, 4, 5$ we have one break from monotonicity, where from 2 we go to 1. We add the number of breaks to the size of the break. In this case, $\text{BC}(y) = 2$. The original constituent order always gets value 0. From the remaining collection of orderings we can now select that one with the lowest value. This always results in a unique ordering.

3.3 Source Language Parser

To derive the dependency trees we used the Alpino (Bouma et al., 2000) parser.[2] Because this grammar comes with dependency information we closely follow their definition of head-dependent relations, deviating from this in only one respect. The Alpino parser marks auxiliary verbs as being the head of a complete sentence, while we took the main verb as the head of sentence, transforming the parse trees accordingly (thus moving closer to semantic dependencies).

The Alpino parser does not always produce projective parses. Reading off parse trees of Alpino in some cases already changes the word order.

[2]We would like to thank van Noord from the University of Groningen for kindly providing us the parses made by Alpino for most of the Dutch sections for the relevant data.

3.4 Four Reordering Models

We investigate four models.

The first, the 'Alpino model', is to measure the impact of the parser used, as Alpino does some reordering that is intended to be linguistically helpful. We want to know what the result is of using a dependency parser which does not generate projective parses i.e. there are parses which do not result into the original sentence if we read off the tree for this parse. In this model we parse the sentences with our parser, and we simply read off the tree. If the tree is projective this results in the original tree. If this is not the case we keep for every node the original order of its children.

The second, the 'full model', chooses the reordering as described in sections 3.1 and 3.2. We hypothesised the algorithm may be too 'enthusiastic' in reordering. For example, when we encounter a ditransitive verb the algorithm usually would put either the direct or the indirect object in front of the subject. Longer constituents were moved to completely different positions in the sentence. This kind of reordering could be problematic for languages, like English, which heavily rely on sentence position to mark the different grammatical functions of the constituents.

We therefore defined the 'limited model', a restriction on the previous model where only single tokens can be reordered. When analysing previous syntactically motivated reordering (Collins et al., 2005) we realised that in most cases constituents consisting of one token only were repositioned in the sentence. Furthermore since sentence ordering is so important we decided only to reorder if 'substantial' parts were changed. To do this we introduced a threshold R and only accepted a new ordering if the new SHD^2 has a reduction of at least R in regard to the original sentence ordering. If it is not possible to reduce SHD^2 that far we would keep the original ordering. Varying R between 0 and 1, in this preliminary work we determined the value $R = 0.9$.

Finally we reimplemented the six rules in Collins et al. (2005) as closely as possible given our language pair and our parser, the Alpino parser. The 'Collins Model' will show us the im-

	$n=1$	$n=2$	$n=3$	$n=4$	BP[a]	**BLEU**
Baseline	0.519	0.269	0.155	0.0914	0.981	0.207
Alpino	0.515	0.264	0.149	0.085	0.972	0.198
Full	0.518	0.262	0.146	0.083	0.973	0.196
Limited	0.521	0.271	0.155	0.0901	0.964	0.203
Collins	0.521	0.276	0.161	0.0966	0.958	0.208

[a]brevity penalty of BLEU

Table 1: Automatic Evaluation Metrics for the different Models

pact of our parser and our choice of language pair.

4 Experimental Setup

For our experiments we used the decoder Pharaoh (Koehn, 2004). For the phrase extraction we used our implementation of the algorithm which is described in the manual of Pharaoh. As a language model we used the SRILM (Stolcke, 2002) toolkit. We used a trigram model with interpolated Kneser-Ney discounting.

For a baseline we used the Pharaoh translation made with a normal GIZA++ (Och and Ney, 2003) training on unchanged text, and the same phrase extractor we used for our other four models.

As an automated scoring metric we used the BLEU (Papineni et al., 2002) and the F-Measure (Turian et al., 2003)[3] method.

For our training data we used the Dutch and English portions of most of the Europarl Corpus (Koehn, 2003). Because one section of the Europarl corpus was not available in a parsed form, this was left out. After sentence aligning the Dutch and the English part we divided the corpus into a training and a testing part. From the original available Dutch parses we selected every 200th sentence for testing, until we had 1500 sentences. We have a little over half a million sentences in our training section.

[3]In this article, we used our own implementations of the BLEU and the F-Measure score available from `http://www.ics.mq.edu.au/~szwarts/Downloads.php`

5 Results

For the three models and the baseline, results are given in Table 1. The first interesting result is the impact of the parser used. In the Europarl corpus 29% of the sentences have a different word order when just reading off the Alpino parse compared to the original word order. It turns out that our results for the Alpino model do not improve on the baseline.

In the original Collins et al. work, the improvement over the baseline was from 0.252 to 0.268 (BLEU) which was statistically significant. Here, the starting point for the Collins reordering is the read-off from the Alpino tree; the appropriate baseline for measuring the improvement made by the Collins reordering is thus the Alpino model, and the Collins model improvement is (a comparable) 0.01 BLEU point.

The Full Reordering model in fact does worse than the Alpino model. However, in our Limited Reordering model, our scores show a limited improvement in both BLEU and F-Measure above the Alpino model score.

In this model only half of the sentences (49%) are reordered compared to the original source sentences. But as mentioned in section 2 not-reordered sentences can also be translated differently because we hope to have a better phrase table. When we compare sentence orders from this model against the sentence ordering from the direct read-off from the Alpino parser 46% of the sentences have a different order, so our method does much more than changing the 29% changed sentences of the Alpino read-off up to 49%.

In Table 2 we present some examples where we actually produce better translations than the baseline,

		Limited Reordering success
1	B	the democratisation process is web of launched
	L	the democratisation process has been under way
	R	the process of democratisation has only just begun
2	B	the cap should be revised for farmers to support
	L	the cap should be revised to support farmers
	R	the cap must be reorganised to encourage farmers
3	B	but unfortunately i have to my old stand
	L	but unfortunately i must stand by my old position
	R	but unfortunately i shall have to adhere to my old point of view
4	B	how easy it is
	L	as simple as that
	R	it is as simple as that
5	B	it is creatures with an sentient beings
	L	there are creatures with an awareness
	R	they are sentient creatures
		Limited Reordering failure
6	B	today we can you with satisfaction a compromise proposal put
	L	today we can you and i am pleased submitting a compromise proposal
	R	i am pleased to see that we have today arrived at a compromise motion
7	B	this is a common policy
	L	is this we need a common policy
	R	a common policy is required in this area

Table 2: Examples of translation, B: Baseline, L: Our Limited Reordering model, R: Human Reference

and below that some examples where the baseline beats our model on translation quality. Example 3 takes advantage of a moved verb; the original Dutch sentence here ends with a verb indicating that the situation is unchanged. Example 2 also takes advantage of a moved final verb. In example 4, the baseline gets confused by the verb-final behaviour.

6 Discussion

The Full Reordering model, without the limitation of moving only one token constituent and the R threshold, reorders most of the sentences: 90% of the Dutch sentences get reordered. As can be seen from Table 1 our scores drop even further than using only the Alpino model. Getting too close to the ideal of limiting dependency distance, we actually move large clauses around so much, for a language which depends on word order to mark grammatical function, that the sentences gets scrambled and lose too much information. Manually judging the sentence we can find examples where the sentence locally improved in quality, but overall most translations are worse than the baseline.

In addition, the phrase table for the Fully Reordered model is much smaller than the phrase table for the non-reordered model. At first we thought this was due to the new model generalising better. For example we find the verb particle more often next to the governing verb than in other contexts. However a better explanation for this in light of the negative results for this model is based on the GIZA++ training. Eventually the phrases are derived from the output of a GIZA++ training which iteratively tries to build IBM model 4 (Brown et al., 1994) alignments on the sentence pairs. When the source sentences are extremely reordered (e.g. an object moved before the subject) the distortion model of model 4 makes it harder to link these words, so eventually we would extract fewer phrases.

Regarding the results of the Limited model compared to original Collins et al. results, we used the default settings for Pharaoh, while Collins et al. probably did not. This could explain the difference in baseline scores (0.207 vs 0.252) for languages with similar syntactic features.

Comparing the results of the Limited model to the reimplementation of the Collins rules in this work, we see that we have achieved half of the improvement without using any language-specific rules. That the approach works by bringing related words closer, in a way that can be taken advantage of by the phrase mechanisms of SMT without ex-

plicitly matching the syntax of the target language, suggests that this is a source of the improvement obtained by the reordering approach of Collins et al.

In future work we would like to implement the proposed reordering technique after correcting for the parser distortions, hopefully confirming the Collins results over the standard baseline for this language pair, and also confirming the relative improvement of the approach of this paper.

7 Conclusion

Previous work has demonstrated that reordering of the text in the source language can lead to an improvement in machine translation quality. Earlier methods either tried to learn appropriate rules for reordering, or have used hand-coded rules that take account of specific differences between language pairs. In this work, we have explored how a claimed universal property of language — that there is a tendency to minimise the distance between a head and its dependents — can be adapted to automatically reorder constituents in the source language. This leads to an improvement in translation quality when the source language, Dutch, is one where this tendency is less present than in the target language English. We demonstrate that, in the Dutch-English case, unrestricted application of the head-dependent distance minimisation strategy is not optimal, and that a restricted version of the strategy does best; we show that this can achieve half of the improvement of the handcrafted rules by using only one language-independent principle, and suggests that what is contributing to the improvement obtained in the reordering is the collapsing of elements into the phrasal window.

References

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2000. Alpino: Wide Coverage Computational Analysis of Dutch. In *Computational Linguistics in the Netherlands (CLIN)*.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The Mathematic of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

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-Juss and Jos A. R. Donollosa. 2006. Statistical Machine Reordering. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 70–76.

J. A. Hawkins. 1990. A parsing theory of word order universals. *Linguistic Inquiry*, 21:223–261.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

Philipp Koehn. 2003. Europarl: A Multilingual Corpus for Evaluation of Machine Translation Philipp Koehn, Draft, Unpublished.

Philip Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models, Philipp Koehn, AMTA.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40^{th} Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Jul.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan, June. Association for Computational Linguistics.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. 2003. Evaluation of machine translation and its evaluation. In *Proceedings of MT Summit IX*.

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.

Analysis and Prediction of User Behaviour in a Museum Environment

Karl Grieser, Timothy Baldwin and **Steven Bird**
Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
{kgrieser,tim,sb}@csse.unimelb.edu.au

Virtual environments such as internet web sites, virtual maps, and even video game landscapes provide digital representations of conceptual or real spaces. In many cases these virtual maps are best understood by relating them to a corresponding physical environment. Moving around these virtual spaces gives users a feeling of moving through a simulated physical environment.

The way in which a virtual space can describe its physical counterpart allows us to use the information that is easily accessible in the virtual environment to give added meaning to elements of the physical environment. This enables a person in an information-rich physical environment such as a city, town, shop or museum to gain access to this otherwise hidden layer of content through the use of portable technology.

This study aims to find accurate methods of predicting how a user will act in an information-rich space. The space focused on in this research is a museum. The predictions take the form of which exhibits a visitor will visit given a history of previously visited exhibits. These predictions can be used to give the person within the environment recommendations on what locations they may wish to visit in the future, or to relate information to the person that is more relevant to them. A core contribution of this study is its focus on the relative import of heterogeneous information sources a user makes use of in selecting the next exhibit to visit.

Building a Recommender System based on contextual information such as those given in Resnick and Varian (1997) is the major goal here. However the environment in this circumstance is physical, and the actions of visitors are expected to vary within such a space. The effectiveness of statistical information to predict user paths is detailed and summarised in Zukerman and Albrecht (2001). Additionally, the relationship between the nature of the data and the context of the problem in which it is used for prediction is acknowledged.

The relationships exhibits have with one another is key in determining how visitors think about them. An exhibit in a museum may be many kinds of things, and most exhibits will differ in presentation and content. Any conceptual representation of an exhibit must contain the elements which identify it, be they physical attributes, such as size, colour and shape, or detailed descriptions about the content of the exhibit. Similarly it is necessary for any prediction system to have knowledge of how to treat relationships between exhibits.

The domain in which all experimentation takes place is the Australia Gallery of the Melbourne Museum. We categorised each exhibit by way of its **physical attributes** (e.g. size) and taxonomic information about the **exhibit content** (e.g. clothing or animal). We also described each exhibit by way of its **physical location** within the Australia Gallery, relative to a floorplan of the Gallery.

A conceptual model of the exhibition space is created by visitors with a specific task in mind. Interpretation of this conceptual model is key to creating accurate recommendations.

The representation of these intrinsically dynamic models is directly related to the task the visitor has in mind. Hence multiple exhibit similarity measures are necessary.

The models of exhibit representation we examine in this research are exhibit proximity, semantic relatedness and exhibit sequentiality (based on the path data of previous visitors), as well as combinations of the three. For our present purposes,

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 157–158.

semantic content is described by the set of keywords associated with each exhibit, although we hope to extend this in future research to look at full document representations based on associated web documents. Semantic content is represented by way of a term vector, describing the attributes of the exhibit. The three distinct conceptual similarity measures provide insight into how visitors perceive their own paths.

The methods proposed above were tested over a representative sample of three exhibit sequences. These exhibit sequences are designed to test a selection of different conceptual models that may be adopted by museum visitors.

1. v_1: A visitor interested in the history of the Melbourne CBD, a total of 6 exhibits visited.

2. v_2: A visitor interested in aboriginal art and tools, a total of 6 exhibits visited.

3. v_3: A visitor wandering, approaching things that catch his/her eye, a total of 17 exhibits visited.

The goal of testing the predictive methods against the multiple visitors is to infer which form of conceptual model each user is adhering to.

Representations based on physical proximity take into account little of how a visitor conceptualises a museum space. They do however describe the fact that closer exhibits are more visible to visitors, and are hence more likely to be visited. Proximity can be used as an augmentation to conceptual models designed to be used within a physical space.

Our preliminary experiments show that visitors entering the museum with a preconceived conceptual model relate strongly to the semantic information associated within exhibits. Visitors entering the museum with no such model tend to follow the paths of other users, exhibiting behaviour that relates to a collaborative predictive approach augmented by individual perceptions of the semantic space.

References

Paul Resnick and Hal R. Varian. 1997. Recommender systems. *Commun. ACM*, 40(3):56–58.

Ingrid Zukerman and David W. Albrecht. 2001. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11:5–18.

Using Dialogue Acts to Suggest Responses in Support Services via Instant Messaging

Edward Ivanovic
Department of Computer Science and Software Engineering
University of Melbourne
edwardi@csse.unimelb.edu.au

Abstract

Instant messaging dialogue is real-time, text-based computer-mediated communication conducted over the Internet. Messages sent over instant messaging can be encoded We propose a method of using dialogue acts to predict utterances in task-oriented dialogue. Dialogue acts provide a semantic representation of utterances in a dialogue. An evaluation using a dialogue simulation program shows that our proposed method of predicting responses provides useful suggestions for almost all response types.

1 Introduction

Support services in many domains have traditionally been provided over the telephone: when customers have queries, they dial a support number and speak to a support representative. Recent years have seen an increasing trend in support services provided over the Internet. Many companies have web sites with Frequently Asked Questions (FAQs), and also offer e-mail support. More recently, real-time support via *online chat sessions* is being offered where customers and support representatives type short messages to each other.

Chat sessions are conducted over a network, such as the Internet, where textual messages can be sent and received between interlocutors in real-time. These chat sessions are commonly referred to as *instant messaging*.

Support services that are conducted via instant messaging vary from being person-person dialogue, similar to traditional call centres, through to being entirely automated where customers engage in dialogue with a computer program. Commercial software is available to partially automated online support by suggesting responses to a human agent, which may then be accepted or overwritten.

Speaker	Message
Agent	[Hello Jim]$^{\text{CONVENTIONAL-OPENING}}$, [thank you for contacting MSN Shopping]$^{\text{THANKING}}$. [This is Sanders and I look forward to assisting you today]$^{\text{STATEMENT}}$
Agent	[How are you doing today?]$^{\text{OPEN-QUESTION}}$
Customer	[good]$^{\text{STATEMENT}}$, [thanks]$^{\text{THANKING}}$
Agent	[How may I help you today?]$^{\text{OPEN-QUESTION}}$

Table 1: An example of the beginning of a dialogue in our corpus showing utterance boundaries and dialogue-act tags in superscript.

The research presented in this paper aims to provide a degree of natural language understanding to assist in automating task-oriented dialogue, such as support services, by suggesting utterances during the dialogue. We apply various probabilistic methods to improve discourse modelling in the support services domain.

In previous work, we collected a small corpus of task-oriented dialogues between customers and support representatives from the MSN Shopping online support service (Ivanovic, 2005b). The service is designed to assist potential customers with finding various items for sale on the MSN Shopping web site. A sample from one of the dialogues in this corpus is shown in Table 1.

The research presented here advances previous work which examined various models and tech-

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 159–160.

niques to predict dialogue acts in task-oriented instant messaging. In Ivanovic (2005b), the MSN Shopping corpus was collected and a gold standard produced by labelling the utterances with dialogue acts. Probabilistic models were then trained to predict dialogue acts given a sequence of utterances. Ivanovic (2005a) examined probabilistic and linguistic methods to automatically segment messages from the same corpus into utterances. The present paper concludes this work by applying the models to a dialogue simulation program which suggests utterance responses during a dialogue. The performance of the suggested utterances is then evaluated.

2 Background

Our dialogue act tag set contains 12 dialogue acts, which are intended to represent the illocutionary force of an utterance. The tags were derived in Ivanovic (2005b) by manually labelling the MSN Shopping corpus using the tags that seemed appropriate from a list of 42 tags in Stolcke et al. (2000).

The MSN Shopping corpus we use comprises approximately 550 utterances and 6,500 words. Ivanovic (2005b) describes the manual process of segmenting the messages into utterances and labelling the utterances with dialogue act tags to produce a gold standard version of the data. Kappa analysis on both the labelling and segmentation tasks was conducted with results showing high inter-annotator agreement (Ivanovic, 2005a).

3 Evaluation and Results

As part of a high-level, end-to-end evaluation of dialogue act prediction and their usefulness in semi-automated dialogue systems, we developed a program that simulates a live conversation while suggesting responses. The suggested utterances are ranked by their respective probabilities given the dialogue history.

We use cross-validation by training the system on all but one dialogue in our corpus. Following training, a customer support scenario is played out using the one dialogue that was not used for training, known as the *target dialogue*. The aim is to replicate substantially all of the utterances in the target dialogue. The process is repeated for each dialogue in our corpus.

Our interface displays a ranked list of suggested dialogue acts and utterances. The dialogue acts are ranked from highest to lowest probability as determined by the naive Bayes model. The utterances within the dialogue acts are ranked by their frequency count during training. However, many utterances are only seen once, in which case the ordering is assumed random as their frequencies are equal. Our evaluation is only focussed on the dialogue-act rankings, not the utterance rankings. When a dialogue act is selected in the "Suggestions" list, the list of utterances is updated to show the relevant utterances for that dialogue act.

Our support dialogue simulation program showed that it is possible to accurately predict many utterances using dialogue acts; 61% of utterances were correctly predicated within the top three ranked dialogues: 22% were in the first rank and 27% in the second.

References

Edward Ivanovic. 2005a. Automatic utterance segmentation in instant messaging dialogue. In *Proceedings of the Australasian Language Technology Workshop*, pages 241–249, Sydney, NSW, Australia, December. Australasian Language Technology Association.

Edward Ivanovic. 2005b. Dialogue act tagging for instant messaging chat sessions. In *Proceedings of the ACL Student Research Workshop*, pages 79–84, Ann Arbor, Michigan, USA, June. Association for Computational Linguistics.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Probabilities improve stress-prediction in a CFG of Hawaiian phonology

ʻŌiwi Parker Jones
Phonetics Laboratory
41 Wellington Square
Oxford, OX1 2JF
United Kingdom
oiwi.parkerjones@ling-phil.ox.ac.uk

1 Introduction

Should probability play a role in linguistics? Whereas Chomsky (1957: 16–17) influentially rejected probability in syntactic theory, "[i]n phonological theory, probability has not so much been rejected as disregarded" (Coleman, 2003: 89). Disregard has, however, given way to a growing literature on the use of probability across the various linguistic sub-disciplines (see, e.g., Bod et al., 2003; Coleman, 2003).

This paper is a case-study of probability in phonology, both as it applies to an improved description of Hawaiian stress-assignment, and as this description, in turn, reflects back on the probability question, above.

2 Grammars

By formalizing two strongly equivalent analyses, where one is a non-probabilistic Context-Free Grammar (CFG) and the other is a Stochastic Context-Free Grammar (SCFG) (Booth, 1969; Suppes, 1970), we can put the probability question to a test. For a given data set, if the SCFG does not outperform its strongly equivalent CFG, then parsimony should compel us to reject, rather than disregard, the added probabilities.

On the other hand, should the SCFG outperform its strongly equivalent, non-probabilistic CFG, then we ought, at least, to accept some role for probability in phonology; this would support the growing literature mentioned above.

Let our data set be Hawaiian. Schütz (1978, 1981) argues that Hawaiian stress-assignment is not 100% predictable, based on words like /ma.ˌku.a.ˈhi.ne/ 'mother' and /ˌʔe.le.ma.ˈku.le/ 'old man'. It might help to illustrate this argument by developing Schütz's analysis into a CFG.

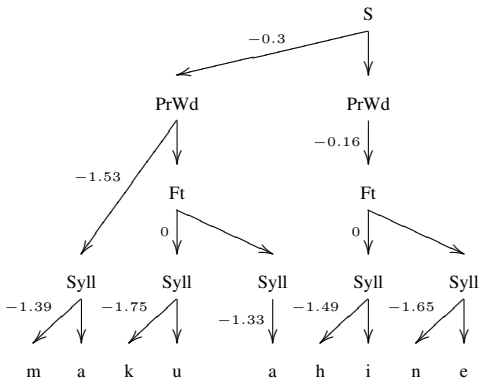

Figure 1: /ma.ˌku.a.ˈhi.ne/ parse-tree.

The crucial parse-trees for 'mother' and 'old man' are in Figures 1–4. Note that the terminal symbols are phonemes. The non-terminal symbols are syllables (Syll), metrical-feet (Ft), prosodic words (PrWd), and the start symbol (S). Also note that the leftmost syllable in each metrical-foot is stressed. The rightmost stress in a word is primary. Finally, let us ignore the labeled branches for the moment, as they do not apply to the non-probabilistic CFG.

The argument follows. The correct parse for 'mother' (Figure 1) is paralleled by an incorrect parse for 'old man' (Figure 2); except for their terminal expansions, these parse-trees have the same structure. Thus, the correct parse for 'mother' implies an incorrect parse for 'old man'. Moreover, the correct parse for 'old man' (Figure 3) implies an incorrect parse for 'mother' (Figure 4). Therefore, no matter how we order the CFG rules, a procedural interpretation will get either 'mother' or 'old man' wrong. Hence, Hawaiian stress-assignment is not 100% predictable.

Although this conclusion might be true (and

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 161–162.

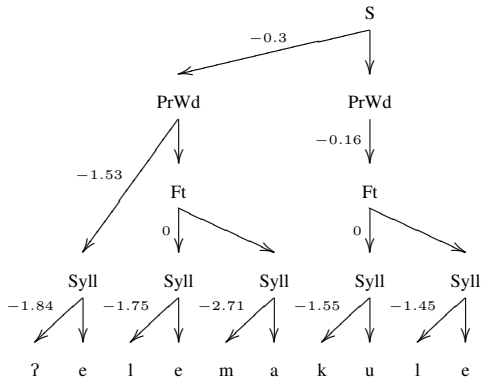

Figure 2: */ʔeˌleˈmaku.le/ parse-tree.

there is nothing here to disprove it), the SCFG turns out to be better than its strongly equivalent CFG at predicting stress-assignment in Hawaiian.

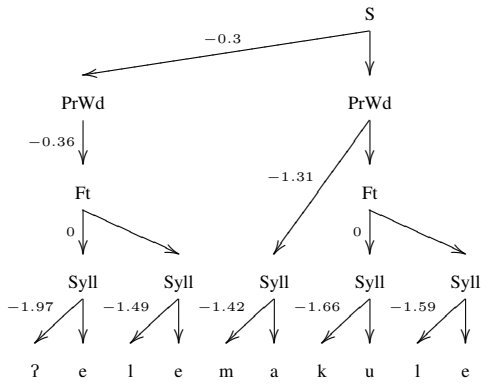

Figure 3: /ˌʔeˈleˈmaˈkule/ parse-tree.

In Figures 1–4, each labeled branch expresses the base-10 log probability for some SCFG rule, where the probabilities were obtained by training the grammar on data from a Hawaiian dictionary (Pūkuʻi and Elbert, 1986). The probability of a parse-tree is just the sum probability of its rules, so Figure 2's probability is -11.29. By contrast, Figure 3's probability is -10.09. The SCFG correctly picks /ˌʔeˈleˈmaˈkule/ over */ʔeˌleˈmaˈkule/, since a log probability of -10.09 is higher than a log probability of -11.29. Moreover, the SCFG correctly picks /maˌkuaˈhine/ over */ˌmakuaˈhine/, since the log probability of -9.59 is higher than that of -9.95. In both examples, the SCFG correctly disambiguates the parses.

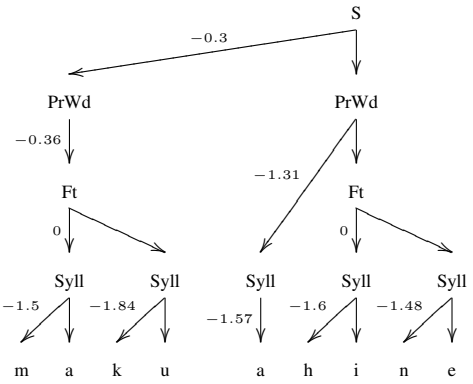

Figure 4: */ˌmaku.aˈhine/ parse-tree.

3 Evaluation

In a computational evaluation of 16,900 Hawaiian words, the CFG correctly parsed 84.6%. However, the SCFG correctly parsed 97.38%. These results demonstrate that probabilities improve stress-prediction in a CFG of Hawaiian phonology; there is a role for probability in phonology.

Acknowledgments

Thanks to John Coleman for supervision, Keola Donaghy for the electronic Hawaiian dictionary, Greg Kochanski for discussions on probability, and Lamakū for financial support.

References

R. Bod, J. Hay and S. Jannedy, eds. 2003. *Probabilistic Linguistics*. MIT Press, Cambridge, MA.

T. L. Booth. 1969. Probabilistic representation of formal languages. *Tenth Annual IEEE Symposium on Switching and Automata Theory*. Pp. 74–81.

N. Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

J. S. Coleman. 2003. Commentary: probability, detail and experience. In J. Local, R. Ogden, and R. Temple, eds. *Phonetic Interpretation: Papers in Laboratory Phonology VI*. Cambridge University Press, Cambridge. Pp. 88–100.

M. K. Pūkuʻi and S. H. Elbert. 1986. *Hawaiian Dictionary*. University of Hawaiʻi Press, Honolulu.

A. J. Schütz. 1978. Accent in two Oceanic languages. *Anthropological Linguistics*, 20(4): 141–149.

A. J. Schütz. 1981. A reanalysis of the Hawaiian vowel system. *Oceanic Linguistics*, 20(1): 1–43.

P. Suppes. 1970. Probabilistic grammars for natural languages. *Synthese*, 22: 95–116.

Towards Cognitive Optimisation of a Search Engine Interface

Kenneth Treharne
School of Informatics and Engineering
Flinders University of South Australia
PO Box 2100 Adelaide, 5001
treh0003@flinders.edu.au

Darius Pfitzner
School of Informatics and Engineering
Flinders University of South Australia
PO Box 2100 Adelaide, 5001
pfit0022@flinders.edu.au

David M W Powers
School of Informatics and Engineering
Flinders University of South Australia
PO Box 2100 Adelaide, 5001
David.Powers@flinders.edu.au

Abstract

Search engine interfaces come in a range of variations from the familiar text-based approach to the more experimental graphical systems. It is rare however that psychological or human factors research is undertaken to properly evaluate or optimize the systems, and to the extent this has been done the results have tended to contradict some of the assumptions that have driven search engine design. Our research is focussed on a model in which at least 100 hits are selected from a corpus of documents based on a set of query words and displayed graphically.

Matrix manipulation techniques in the SVD/LSA family are used to identify significant dimensions and display documents according to a subset of these dimensions. The research questions we are investigating in this context relate to the computational methods (how to rescale the data), the linguistic information (how to characterize a document), and the visual attributes (which linguistic dimensions to display using which attributes).

1 Introduction

Any search engine must make two kinds of fundamental decisions: how to use keywords/query words and what documents to show and how. Similarly every search engine user must decide what query words to use and then be able to interact with and vet the displayed documents. Again every web page author or designer makes decisions about what keywords, headings and link descriptions to use to describe documents or sections of documents. This paper presents one set of experiments targeted at the choice of keywords as descriptive query words (nWords[1]) and a second set of experiments targeted at explaining the effectiveness of the graphical options available to display and interact with the search results.

2 Term relevance and Cognitive Load

There is considerable literature on visualisation techniques and a number of experimental and deployed search engines that offer a visualisation interface e.g. kartoo.com and clusty.com. However, there is little research to establish the effectiveness of such techniques or to evaluate or optimise the interface. This is surprising given the many theories and studies that target memory and processing limitations and information channel capacity, including many that build on and extend the empirical results summarized in George Miller's famous Magical Number Seven paper (1956). It seems likely that for any search task visualization to realize optimal use of channel capacity, it should permit users to draw on their powerful and innate ability of pattern recognition.

Another important aspect that has never been properly evaluated relates to the question of "which words do people use to describe a document?" Techniques like TFIDF are used in an attempt to automatically weight words according to how important they are in characterizing a document, but their cognitive relevance remains unexplored.

[1] nWords is available at http://dweb.infoeng.flinders.edu.au

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 163–164.

Our work will enhance the document search process by improving the quality of the data the user filters while reducing machine processing overheads and the time required to complete a search task. It will also provide fundamental insight into the way humans summarise and compress information. The primary objective of this part of our research is to quantify the number of words a broad spectrum of people use to describe different blocks of text and hence the number of dimensions needed later to present this information visually. Our secondary objective is to enhance our understanding of the approaches taking by users during completion of search tasks. A third objective is to understand the choices a user makes in selecting keywords or phrases to describe or search for a document.

2.1 nWords Results

Results for the nWords experiments explaining use of terms as descriptors or queries using a web based scenario are presented in Table 1 (with standard deviations). It is not only instructive to compare the results across task conditions (with/without access to the text, with/without restricting terms to occur in text), but the difference between the sub conditions where subjects were asked for keywords that described the document (D) versus search query terms (Q) they would use.

Descriptor & Query Term Usage	Task1	Task2	Task3
Number of D's Used	3.79 ± 3.34	4.02 ± 2.39	4.98 ± 3.35
Total Distinct Stems Used in Ds	8.22 ± 7.40	7.27 ± 4.71	11.80 ± 10.63
Average Distinct Stems per D	2.40 ± 1.83	2.14 ± 1.64	3.02 ± 3.46
Distinct D Stems in Top 10 TFIDF	1.79 ± 1.62	1.85 ± 1.52	2.62 ± 1.91
Total Distinct Q Stems Used in Qs	3.28 ± 1.78	3.81 ± 1.99	3.85 ± 1.89
Distinct Q Stems in Top 10 TFIDF	0.98 ± 0.95	1.19 ± 1.02	1.38 ± 0.99
Q Stem / D Stem Intersections	1.83 ± 1.58	2.52 ± 1.69	2.70 ± 1.59

Table 1: Statistics for nWords survey tasks (± standard deviation). Descriptors (D), query terms (Q). Task 1 Access to text, Task 2 No access, Task 3 Term must be in text.

From the results of Table 1 we can see that TFIDF is poor at ranking keywords and query words. For the full data pool, only 2.11 ± 1.74 of the top ten TFIDF terms are used in description which best describes a text, whilst only 1.19 ± 0.99 are used in the query building task.

3 Graphical Representations

Little research on perceptual discrimination of dynamic encodings can be found, but a few empirical studies have investigated the human visual system's ability to detect movement. Motion is a visual feature that is processed pre-attentively. Motion cues can be detected easily and quickly. Overall dynamic encodings outperform static encodings for attracting attention.

3.1 Preliminary Results

Our Miller inspired experiments using web based applets varied properties of icons either statically or dynamically as show in figure 1. In general our preliminary results indicate that dynamic icons yield slower performances in relation to static vs. dynamic variation of any attribute other than size (see error bars in figure 1). Intuition tells us that some aspect of the performance slow down is due to the fact that a dynamically encoded feature requires longer identification time since the encoding is time based. However, we also report that a high degree of volatility or variance was observed in all dynamic conditions. Significant differences were observed between static encodings and their dynamic equivalents in all cases except feature size (and hue - we do not report a flashing hue condition currently).

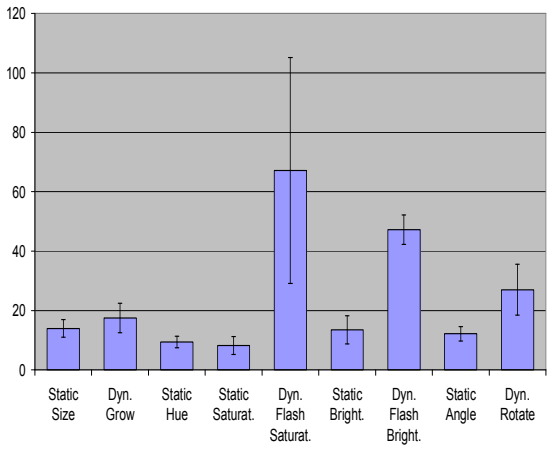

Figure 1 Aggregated subject response times (seconds) across 9 static and dynamic conditions.

An analysis over repeated sessions did not reveal any learning trend. However, we do note that significant outliers were regularly encountered at the beginning of all experiments indicating that first timers took a while to learn how to use the experiment software. Possible outliers in trials midway through the duration of some experiments may indicate that the subject was distracted during the search task. Experiment subjects. This is possible given that subjects administered the test in their own time.

Natural Language Processing and XML Retrieval

Alan Woodley
Faculty of Information Technology
Queensland University of Technology
a.woodley@qut.edu.au

Xavier Tannier
Dept. Networks, Information, Multi-media
Ecole des Mines Saint-Etienne
tannier@emse.fr

Marcus Hassler
Department of Informatics
Universitdat Klagenfurt
Marcus.Hassler@uni-klu.ac.at

Shlomo Geva
Faculty of Information Technology
Queensland University of Technology
s.geva@qut.edu.au

Abstract

XML information retrieval (XML-IR) systems respond to user queries with results more specific than documents. XML-IR queries contain both content and structural requirements traditionally expressed in a formal language. However, an intuitive alternative is natural language queries (NLQs). Here, we discuss three approaches for handling NLQs in an XML-IR system that are comparable to, and even outperform formal language queries.

1 Introduction

Information retrieval (IR) systems respond to user queries with a ranked list of relevant documents, even though only parts of the documents are relevant. In contrast, XML-IR systems are able to exploit the separation of structure and content in XML documents by returning relevant *portions* of documents. To interact with XML-IR systems users must specify both their content and structural requirements in structured queries. Currently, formal languages are used to specify structured queries, however, they have proven problematic since they are too difficult to use and are too tightly bound to the collection.

A promising alternative to formal queries are natural language queries (NLQs). Here, we present justifications for NLQs in XML-IR, and describe three approaches that translate NLQs to an existing formal language (NEXI). When used in with an XML-IR system the approaches perform strongly, at times outperforming a baseline consisting of manually constructed NEXI expressions. These results show that NLQs are potentially a viable alternative to XML-IR systems.

2 Motivation

There are two major problems with formal query languages for XML-IR that could be rectified with NLQs. First, expressing a structural information need in a formal language is too difficult for many users. O'Keefe and Trotman (2004) investigated five structured query languages and concluded that all of them were too complicated to use. In practice, 63% of the expert-built queries queries in the 2003 INEX campaign had major semantic or syntactic errors, requiring up to 12 rounds of corrections. In contrast, users should be able to express their need in NLQs intuitively.

Second, formal query languages require an intimate knowledge of a document's structure. So, in order to retrieve information from abstracts, sections or bibliographic items, users need to know their corresponding tags. While this information is contained in the DTD/ Schema, it may not be publicly available, and is too much information to remember (INEX, for instance has 192 nodes). The problem extrapolates in a heterogenous collection since a single retrieval unit could be expressed in multiple tags. In contrast, since structures in NLQs are formulated at the conceptional level users do not have to know their actual tag names.

3 The Approaches

Here, we present three techniques used to translate NLQs to NEXI in INEX 2004 and 2005. The three approaches are called *Hassler*, *Tannier* (Tannier, 2005) and *Woodley* (Woodley and Geva, 2005) after their authors. While each of the approaches is different, they all contain four main stages.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 165–166.

Detecting Structural and Content Constraints. The first stage is to detect a query's structural and content constraints. *Hassler* uses template matching based on words and parts-of-speech. Links between structure and content are not linguistically motivated, and it is assumed that content is the last element. *Woodley* adds shallow syntactic parsing before applying the same kind of template matching. *Tannier* uses deep syntactic analysis, complemented by some specific semantic rules concerning query structure.

Structure Analysis. The second stage is to map structural constraints to corresponding XML tags. This requires lexical knowledge about the documents' structure, since the tags in the XML documents are rarely "real" words or phrases, but abbreviations, acronyms or an amalgamation of two. Furthermore, a single tag can be referred to by different names. *Tannier* uses grammatical knowledge to recognise some frequent linguistic constructions that imply structure.

Content Analysis. The third stage is to derive users' content requirements, as either terms or phrases. Noun phrases are particularly useful in information retrieval. They are identified as specific sequences of parts-of-speech. *Tannier* is also able to use content terms to set up a contextual search along the entire structure of the documents.

NEXI Query Formulation. The final stage of translation is the formulation of NEXI queries. Following NEXI format, content terms are delimitated by spaces, with phrases surrounded by quotation marks.

4 Results

Here, we present the ep-gr scores from the 2005 INEX NLQ2NEXI Track. The results correspond to different relevance quantisation and interpretations of structural constraints - a thorough description of which is provided in (Kazai and Lalmas, 2005). The results compare the retrieval performance of a XML-IR system (Geva, 2005) when the 3 natural language approaches and a fourth "baseline" system, which used manually constructed NEXIs queries, were used as input. The results show that the NLP approaches perform comparably - and even outperform - the baseline.

	Baseline	*Hassler*	*Tannier*	*Woodley*
Strict	**0.0770**	0.0740	0.0775	0.0755
Gen	**0.1324**	0.1531	0.1064	0.1051

Table 1: SSCAS ep-gr scores

	Baseline	*Hassler*	*Tannier*	*Woodley*
Strict	**0.0274**	0.0267	0.0304	0.0267
Gen	**0.0272**	0.0287	0.0298	0.0311

Table 2: SVCAS ep-gr scores

	Baseline	*Hassler*	*Tannier*	*Woodley*
Strict	**0.0383**	0.0338	0.0363	0.0340
Gen	**0.0608**	0.0641	0.0682	0.0632

Table 3: VSCAS ep-gr scores

	Baseline	*Hassler*	*Tannier*	*Woodley*
Strict	**0.0454**	0.0372	0.0418	0.0483
Gen	**0.0694**	0.0740	0.0799	0.0742

Table 4: VVCAS ep-gr scores

5 Conclusion

While the application of NLP XML-IR is in its infancy, it has already produced promising results. But if it is to process to an operational environment it requires an intuitive interface. Here, we describe and presented the performance of three approaches for handling NLQs. The results show that NLQs are potentially a viable alternative to formal query languages and the integration of NLP and XML-IR can be mutually beneficial.

References

Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltàn Szlàvik, editors. 2006. *INEX 2005 Proceedings, Schloss Dagstuhl, Germany, November 28–30, 2005.*

Shlomo Geva. 2005. GPX - Gardens Point XML Information Retrieval at INEX 2005. In Fuhr et al. (Fuhr et al., 2006), pages 211–223.

Gabriella Kazai and Mounia Lalmas. 2005. INEX 2005 Evaluation Metrics. http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf.

Richard A. O'Keefe and Andrew Trotman. 2004. The Simplest Query Language That Could Possibly Work. In *Proceedings of the second Workshop of the INEX, Schloss Dagstuhl, Germany.*

Xavier Tannier. 2005. From Natural Language to NEXI, an interface for INEX 2005 queries. In INEX05 (Fuhr et al., 2006), pages 289–313.

Alan Woodley and Shlomo Geva. 2005. NLPX at INEX 2005. In INEX05 (Fuhr et al., 2006), pages 274–288.

Extracting Patient Clinical Profiles from Case Reports

Yitao Zhang and **Jon Patrick**
School of Information Technology
University of Sydney
NSW 2006, Australia
{yitao, jonpat}@it.usyd.edu.au

Abstract

This research aims to extract detailed clinical profiles, such as signs and symptoms, and important laboratory test results of the patient from descriptions of the diagnostic and treatment procedures in journal articles. This paper proposes a novel mark-up tag set to cover a wide variety of semantics in the description of clinical case studies in the clinical literature. A manually annotated corpus which consists of 75 clinical reports with 5,117 sentences has been created and a sentence classification system is reported as the preliminary attempt to exploit the fast growing online repositories of clinical case reports.

1 Corpus and Mark-up Tags

This paper proposes a mark-up scheme aimed at recovering key semantics of clinical case reports in journal articles. The development of this mark-up tag set is the result of analysing information needs of clinicians for building a better health information system. During the development of this tag set, domain experts were constantly consulted for their input and advice.

1.1 The Mark-up Tag Set

- **Sign** is a signal that indicates the existence or nonexistence of a disease as observed by clinicians during the diagnostic and treatment procedure. Typical signs of a patient include the appearance of the patient, readings or analytical results of laboratory tests, or responses to a medical treatment.
- **Symptom** is also an indication of disorder or disease but is noted by patients rather than by clinicians. For instance, a patient can experience weakness, fatigue, or pain during the illness.
- **Medical test** is a specific type of sign in which a quantifiable or specific value has been identified by a medical testing procedure, such as blood pressure or white blood cell count.
- **Diagnostic test** gives analytical results for diagnosis purposes as observed by clinicians in a medical testing procedure. It differs from a medical test in that it generally returns no quantifiable value or reading as its result. The expertise of clinicians is required to read and analyse the result of a diagnostic test, such as interpreting an X-ray image.
- **Diagnosis** identifies conditions that are diagnosed by clinicians.
- **Treatment** is the therapy or medication that patients received.
- **Referral** specifies another unit or department to which patients are referred for further examination or treatment.
- **Patient health profile** identifies characteristics of patient health histories, including social behaviors.
- **Patient demographics** outlines the details and backgrounds of a patient.
- **Causation** is a speculation about the cause of a particular abnormal condition, circumstance or case.
- **Exceptionality** states the importance and merits of the reported case.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 167–168.

Total articles	75
Total sentences	5,117
Total sentences with tag	2,319
Total tokens	112,382
Total tokens with tag	48,394

Table 1: Statistics of the Corpus

- **Case recommendations** marks the advice for clinicians or other readers of the report.

- **Exclusion** rules out a particular causation or phenomenon in a report.

1.2 The Corpus

The corpus described in this paper is a collection of recent research articles that report clinical findings by medical researchers. To make the data representative of the clinical domain, a wide variety of topics have been covered in the corpus, such as cancers, gene-related diseases, viral and bacteria infections, and sports injuries. The articles were randomly selected and downloaded from BioMed Central[1]. During the selection stage, those reports that describe a group of patients are removed. As a result, this corpus is confined to clinical reports on individual patients. A single human annotator (first author) has manually tagged all the articles in the corpus. The statistical profile of the corpus is shown in Table 1.

2 The Sentence Classification Task

The patient case studies corpus provides a promising source for automatically extracting knowledge from clinical records. As a preliminary experiment, an information extraction task has been conducted to assign each sentence in the corpus with appropriate tags. Among the total of 2,319 sentences that have tags, there are 544 (23.5%) sentences assigned more than one tag. This overlapping feature of the tag assignment makes a single multi-class classifier approach not appropriate for the task. Instead, each tag has been given a separate machine-learned classifier capable of assigning a binary 'Yes' or 'No' label for a sentence according to whether or not the sentence includes the targeted information as defined by the tag set. Meanwhile, a supervised-learning approach was adopted in this experiment.

[1]http://www.biomedcentral.com/

Tag	Precision	Recall	F_1
Diagnostic test	66.6	46.8	55.0
Medical test	80.4	51.6	62.9
Treatment	67.6	44.7	53.8
Diagnosis	62.5	33.8	43.8
Sign	61.2	50.5	55.4
Symptom	67.8	45.8	54.7
Patient demographics	91.6	73.1	81.3
Patient health profile	53.0	24.1	33.2

Table 2: Sentence Classification Result for Some Semantic Tags

A Maximum Entropy (MaxEnt) classifier[2] and a SVM classifier (SVM-light) with tree kernel (Moschitti, 2004; Joachims, 1999) were used in the experiment. The SVM classifier used two different kernels in the experiment: a linear kernel (SVM t=1), and a combination of sub-tree kernel and linear kernel (SVM t=6). The introduction of the tree kernel was an attempt to evaluate the effectiveness of incorporating syntactic clues for the task. The feature set used in the experiment consists of unigrams, bigrams, and title of the current section. The experiment results for selected markup tags are shown in Table 2.

Acknowledgments

We wish to thank Prof Deborah Saltman for defining the tag categories and Joel Nothman for refining their use on texts.

References

Thorsten Joachims. 1999. Making Large-scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods-Support Vector Learning*.

Alessandro Moschitti. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. In *Proceedings of the 42-th Conference on Association for Computational Linguistic*.

[2]Zhang Le http://homepages.inf.ed.ac.uk/s0450736/