Proceedings of the
Australasian Language Technology Workshop 2004

December 8th, 2004

Macquarie University
Sydney, Australia

Editors:

Ash Asudeh, Cécile Paris, Stephen Wan

Sponsors:

Australian Government
Department of Defence
Defence Science and
Technology Organisation

Proceedings of the Australasian Language Technology Workshop 2004

Website: http://www.alta.asn.au/events/altw2004/publication/index.html

Published by:

The Australian Speech Science and Technology Association (ASSTA)
Email: secretary@assta.org
or
G.P.O. Box 143, Canberra City, ACT, 2601.

Contact for additional copies:

ALTA Workshop 2004
Email: workshop@alta.asn.au

ISBN: 0 9581946 1 0

Australasian Language Technology Workshop 2004

Foreword

ALTW 2004 is the second workshop organised by the recently founded Australasian Language Technology Association (ALTA), following previous Australasian Natural Language Processing Workshops. The first Australasian Natural Language Processing Workshop was held at Macquarie University, in 2001 and the second in Canberra in 2002. The first ALTW was held in Melbourne in 2003, in conjunction with a Summer School in Language Technology (ALTSS). ALTW is now ALTA's annual meeting. This year, ALTW and the Summer School in Language Technology are being held in conjunction with the Tenth Australian International Conference on Speech Science & Technology (SST 2004).

Every year, the number of submissions (including submissions from overseas) increases significantly, indicating both the growth of the research area in Australasia and the higher international visibility of the workshop. As a result, every year the standard of the workshop also increases. This year, as in previous years, each submission was a full paper and was reviewed by at least two members of an international programme committee of leading researchers in language technology, listed on the next page. We would like to thank the programme committee for their dedicated and careful review of the papers. The call for papers for ALTW solicited papers presenting substantial, original and unpublished research on all aspects of language technology. We have selected 22 papers for the workshop, which is organised in two parallel sessions.

We would like to thank the local organisers of ALTW/ALTSS and SST for their help in organising this workshop, in particular Diego Molla and Steve Cassidy. We would also like to thank the ALTA executive committee for assistance in various respects. Finally, we would like to thank the sponsors (DSTO, Appen, and VeCommerce) for their generous help in supporting the workshop, and Dominique Estival as the sponsorship coordinator.

As a final word, we hope you will all enjoy the workshop. We are very excited to see `that the language technology community in Australia and the region is vibrant and growing. We hope the workshop will once again foster interactions amongst researchers in language technology research.

Ash Asudeh, Cécile Paris and Stephen Wan
ALTW 2004 programme chairs
November 2004

ISBN: 0 9581946 1 0

Program Committee

Ash Asudeh, University of Canterbury (NZ) (Co-chair)
Cecile Paris, CSIRO (AU) (Co-chair)
Stephen Wan, CSIRO and Macquarie University (AU) (Student Chair)
Steven Bird, University of Melbourne (AU)
Steve Cassidy, Macquarie University (AU)
Nathalie Colineau, CSIRO (AU)
Dominique Estival, DSTO (AU)
James Curran, Sydney University (AU)
Alistair Knott, University of Otago (NZ)
Mirella Lapata, University of Sheffield (UK)
Corrin Lakeland, University of Otago (NZ)
Nadine Ozkan, Scansoft (Canada)
Hiroshi Masuichi, Fujixerox (JP)
Daniel Midgley, University of Western Australia (AU)
Harold Somers, UMIST (UK)

Table of Contents:

Complex, Corpus-Driven, Syntactic Features for Word Sense Disambiguation

Ari Chanen and Jon Patrick
Sydney Language Technology Research Group
School of Information Technologies
University of Sydney
Sydney, Australia, 2006
{ari,jonpat}@it.usyd.edu.au

Abstract

Although syntactic features offer more specific information about the context surrounding a target word in a Word Sense Disambiguation (WSD) task, in general, they have not distinguished themselves much above positional features such as bag-of-words. In this paper we offer two methods for increasing the recall rate when using syntactic features on the WSD task by: 1) using an algorithm for discovering in the corpus every possible syntactic feature involving a target word, and 2) using wildcards in place of the lemmas in the templates of the syntactic features. In the best experimental results on the SENSEVAL-2 data we achieved an F-measure of 53.1% which is well above the mean F-measure performance of official SENSEVAL-2 entries, of 44.2%. These results are encouraging considering that only one kind of feature is used and only a simple Support Vector Machine (SVM) running with the defaults is used for the machine learning.

1 Introduction: Syntactic Features

The best features for machine learning classification are the ones that have the most discriminatory power, for the task at hand. This paper will be discussing the use of **syntactic features** (SF's) in **word sense disambiguation** (WSD.) In WSD, the task is to choose (or classify) the correct sense of the **target word** (the word whose sense is to be disambiguated) given the surrounding text. One type of feature that is commonly used in WSD classification systems is called **bag-of-words**. Bag-of-words features are rather information poor, only specifying the presence or absence of words in the target word's context. SF's, on the other hand, are much richer in information. Not only do syntactic features have information on the presence of words in the context but they also include information about the syntactic relationships that hold between the target word and context words in the same sentence as the target.

In order to use SF's, a syntactic parser is needed that produces a parse tree for every training corpus sentence. The parse tree gives the syntactic relationships between words in each sentence. Connexor parser (Järvinen and Tapanainen, 1997) was used to annotate the data with syntactic relationships in the research presented here. In this research, a SF is defined as a connected group of words from a parse tree that must include the target word. The SF includes information on each of its words, the syntactic relationships between them, and information on how each word relates to the others in the tree hierarchy. The SF word information includes the word, lemma, and **part-of-speech** (POS) for each word.

The use of syntactic features in WSD might seem to be a more effective discriminatory feature compared to a information poor feature like bag-of-words because of the potential for SF's to offer more specific information about how the sense of

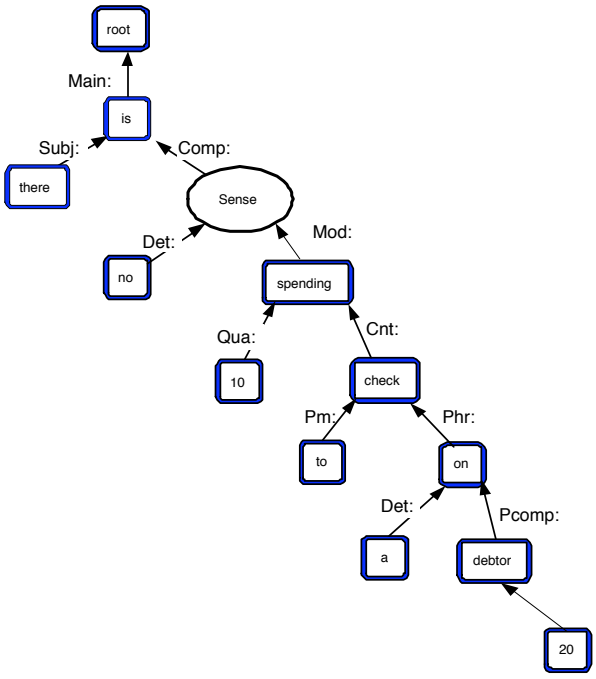


Figure 1: Conexor parser tree for "**There is** no **sense** spending 10 to check on a 20 debtor."

a target word relies not only on the words around it, but also on the information about the syntactic relationships that hold between words. Alas, the fact that SF's contain more detail about the set of words in a specific SF makes it less likely for a given SF to occurred as frequently as the corresponding set of bag-of-words features, thus leading to a general problem with SF's of having lower recall. The issue is data sparseness not whether or not syntactic features have potential as a powerful NLP feature. Rather, the question is how can the strength of syntactic features be boosted. In this paper, we explore two ways to help syntactic features live up to their promise by:

1. Developing an algorithm for finding syntactic features in the sentence that surrounds the target-word. More specifically, the algorithm identifies **all** syntactic features that 1) involve the target word 2) contain a number of syntactic links that is less than or equal to a fixed maximum.
2. Allowing for abstract features. Syntactic features are made up of member elements of various types. The term "abstract" here is being used in the sense of being opposite of concrete. One type of member is the lemma element. A method has been devised to exhaustively enumerate all possible features where one or two wildcards have replaced original lemma elements. Any feature where a lemma has been abstracted to a wildcard is defined as abstract.

Both of these methods are shown experimentally to be effective in boosting recall. The specifics of the methods will be discussed in section 4 and the experimental results will be discussed in section 5.

2 The WSD Task

2.1 General Description

One of the reasons that human language is far from trivial to process is that many words in the lexicon hold different meanings depending on their context. For instance, the word "sense" has five senses as a noun and four senses as a verb according to WordNet 2.0. Two of the five noun senses are exemplified in the following two sentences:

1. There is a pleasing **sense** of justice about the observation. (Here "sense" means "general conscious awareness" – WordNet 2.0)
2. There is no conceivable **sense** in going to the opposite extreme. (Here "sense" means "sound practical judgement" – WordNet 2.0)

The sense of a particular instance of a word in a text can only be determined by the surrounding context.

In the SENSEVAL competitions, teams of researchers build word sense classifiers. The teams are all given the same training examples of the same set of words. The task is to build one classifier for each word that classifies an instance of that word in context as one of its possible senses. The training examples consist of the target-word, along with the surrounding context which is typically several sentences. SENSEVAL is a supervised learning task so all of the training examples supplied by the SENSEVAL organisers come with a label in the form of a sense-tag that nails down the sense of the target-word to one of the senses listed in WordNet.

Different research groups try to outperform the other groups by using different and hopefully superior methods. There are, at least, five basic areas in which the groups may differ 1) training data used 2) enrichment of the training data, if any 3) kinds of features extracted for the machine-learning process 4) method of selecting the best features 5) machine-learning algorithm or combination of algorithms used.

2.2 Data Enhancements by Parsing

The SENSEVAL-2 and SEMCOR sense-tagged corpora were used as the training data in this research. This data was enriched by extracting syntactic information using the Connexor parser. It is difficult to extract reliable syntactic information without first processing the data with a good parser. Connexor is a dependency parser, as opposed to a constituency parser. Any type of syntactic parser that produces a hierarchical sentence tree could be used with the syntactic feature extraction methods used in this work.

2.3 Feature Selection Method

The method used for feature selection follows (Ng and Lee, 1996). Three steps are used on all the features found by a given feature-extractor to filter them down to the selected, final set of features used in the machine learner. According to this paper, each feature must meet the following conditions to be selected:

1. The feature f must have a feature count of at least FC_{min}.
2. The conditional probability for some sense i given the feature f must be greater than a predetermined probability. $P(i \mid f) \geq CP_{min}$.

Condition one is enhanced by allowing the minimum count to depend on the abstraction level of the feature where the **abstraction level** is defined as the number of wildcards in a syntactic feature.

2.4 Motivation

The SENSEVAL-2 papers indicate that no one came up with a single "magic bullet" idea that put them out in front of the crowd, rather the teams that did best were better able to combine known ideas and better able to make small adjustments in the application of these ideas. This is one of reasons this study concentrates on understanding the problems in-depth and improving a single type of feature rather than combining many features and using many different machine learners.

2.5 WSD Learning Infrastructure

Our WSD system is built on an extensible framework for feature extraction and feature vector construction. All of the experiments reported on here were done using this in-house system. Due to limited space, the system will not be described here but in (Bell and Patrick 2004.)

3 Syntactic Features and WSD

3.1 Dekan Lin

(Lin, 2000) describes the only other WSD system we are aware of that makes use of syntactic features alone. Lin's system discovered all syntactic features in the corpus which inspired the current systems principle of only using syntactic features discoved automatically in the corpus. Lin's syntactic features are less inclusive, and less complex then those described in this paper. See section for further comparisons. He used a nearest neighbour (NN) algorithm to choose the best sense of the word. Despite having simpler features, his system showed better performance on the same task. In the conclusion to this paper , we will speculate as to why.

3.2 David Yarowsky et al.

(Yarowsky et al., 2001) describes the system that did best among all competing supervised-learning systems at SENSEVAL-2. This system is not directly comparable because they used five types of features and a more complex, voting scheme machine-learner. Nevertheless, it is instructive to contrast the Yarowsky's system syntactic features with those being described here. Their system identifies a closed set of syntactic feature types first (e.g. verb/obj) and then can only extract those types from the corpus.

3.3 David Fernandez-Amoros

(Férnandez-Amorós, 2004) is also not directly comparable because he uses unsupervised learning. Again though, a comparison can be made be-

tween his syntactic features and those being described here. He first parsed all the WordNet glosses. He looked for parts of the parse trees that contained WSD target-words and used these subtrees as patterns for that target-word. He also used wildcards in place of pronouns and content words, like the current research. He uses transformations on these sub-tree patterns in a further attempts to increase recall. In spirit, his research and that described here are similar however he was not able to achieve the same amount of automation in both identifying syntactic tree topologies and in generating wildcard features. Also, his base-syntactic patterns were limited to those found in the WordNet glosses for a given target-word.

4 Methods

The SF's, described here, were originally inspired by (Lin, 2000), however, a single one of Lin's SF's is not capable of capturing all the different topologies of subtrees involving the target word. Lin chose a limited yet easy to calculate set of syntactic features that involve the target word. Specifically, he extracted features that always started with the target word and included all the dependency links and words that would be touched on the way to any word in the sentence. Because his features never branch but rather are a string of words connected by dependency relationships we call them **linear syntactic features** (LSF's.) In his syntactic features he included the lemma and POS of the words in the feature and the dependency relationships that are on the links in-between the words. We have followed his lead in including this information in the syntactic features.

Figure 1 shows the Connexor dependency parse of a random (and somewhat awkward) sentence from the SENSEVAL-2 data.

An example of a two link feature in this sentence would be one that started at the target word "sense" and then goes to the word "spending" by following the "mod" link and then finally on to "check" by following the "cnt" link.

From Lin's description of his features, even some linear features would not be extracted. For instance, features where the target word is in the middle of a linear path from one word to another in this sentence would not be extracted because his features must start with the target word.

links	ATSF	running total	LSF	running total
1	3	3	3	3
2	6	9	3	6
3	10	19	2	8
4	15	34	1	9
5	21	55	1	10
6	27	82	0	10
7	30	112	0	10
8	26	138	0	10
9	16	154	0	10
10	6	160	0	10
11	1	161	0	10

Table 3: Comparing the number of LSF's vs. ATSF's found in the sentence: "There is no sense spending 10 to check on a 20 debtor." (from the SENSEVAL-2 corpus)

4.1 All-topologies Syntactic Features (ATSF's)

A version of Lin's LSF's was first implemented. Those types of SF's were automatically extracted from the enhanced corpus documents. It was obvious when looking at a dependency parse tree that while many features were identified there were many more potential features that the LSF feature-extractor was not able to catch. Thus, a more all-inclusive class of syntactic feature which we have named **all-topology syntactic features** (ATSF's), was developed.

The major motivating factor behind seeking to extract ATSF's was that it seemed they would be more abundant than LSF's. The basic idea is that **any** subtree of a sentence parse tree with up to and including a maximum number of dependency links could be potentially useful as a feature. Referring to figure 1, one example of a feature that is not a LSF would be one that involves the words **(is, no, sense, spending)**. The links involved in that feature could not be placed in a line. To give an idea of how many more ATSF's there might be compared to LSF's, table 3 shows the feature counts in the example parse tree. The feature count is broken down into groups of features

that have the same number of dependency links in them. In each of our experiments, a parameter sets the maximum number of links that a feature can have in that experiment. The best performance to date comes from a maximum of three links. Table 3 shows that there are five times as many three link ATSF's as three link LSF's. In fact, in this example sentence, there are no LSF's that have more than five links and this sentence is typical.

4.1.1 Canonical Form and Representation

The same subtree could be represented in many different ways so a canonical form needs to be defined. ATSF's are defined as nested elements were each element has the basic form:

{**[***DRWP***]::***lemma***=***POS [children]*}

Where *DRWP* stands for *dependency relationship with parent*. Out of the words in a feature, the word that is topmost in the parse tree being represented is placed first in the feature set. The *DRWP* of that top-most word is deleted. The rest of the dependency relationships further down the tree are represented in the children element of the top tree item. The children of a feature or sub-feature are sorted in alphabetical order by first the POS, then DRWP and finally by lemma.

For example, the feature involving the words **(is, no, sense, spending)** is rendered as:

{::be=v{comp::sense=n{det::no=det}{mod::spending=ing}}}

4.1.2 Algorithm for Identifying ATSF's

In (Férnandez-Amorós, 2004), the author called the problem of systematically identifying all syntactic features "challenging" and said that for lack of time he was not able to come up with a solution yet. We also found it challenging but were able to come up with a divide-and-conquer/dynamic programming solution which is presented in outline form here.

The basic idea is to define a recursive function whose job it is to identify all possible parse tree topologies that can be formed with a constant number of links where all topologies must involve a target tree node and may involve any or all of a group of neighbour nodes and their children. Let us call the function **gen-all-topologies**. It returns a list of features of all topologies. Its arguments are:

target-ID The unique identifier of the target node. This would usually be the ID of the token node for a word.

links The function returns only syntactic features with this many links.

neighbour-IDS The neighbours of the target node which can be used to form the features. Notice, this is usually not all of the neighbours of the **target-ID**

Inside **gen-all-topologies** there is a loop that assigns a variable **links-to-first-neighbour** values from zero to the value of the argument **links**. For each iteration in this loop we try different splits of the **links** between the first neighbour in the list and the target node[1] Here are the two sub-recursive calls:

```
feature-list-1 =
gen-all-topologies(
   first(neighbour-IDS),
   links-to-first-neighbour,
   neighbours*(first(neighbour-IDS))

feature-list-2 =
gen-all-topologies(
   target-ID,
   links - links-to-first-neighbour,
   rest(neighbour-IDS))
```

first and **rest** get the first element and the rest of the elements of a list, respectively. **neighbours*** gets all of the neighbours of a node except for the target node. Once these two sub-recursive calls have returned we do a cross product of the two lists meaning that each member of a list must be combined with each of the features on the other list yielding a number features equal to the product of the sizes of the two lists of features. Determing how to combine two features into a bigger feature has a straightforward solution.

gen-all-topologies is called n times, where the **links** arguments ranges from 1 to n which will obtain all features for a sentence with from 1 to n links in the features.

The implementation of **gen-all-topologies** makes use of dynamic programming techniques,

[1]Only on the top-level call is the target node actually the target word for the WSD problem.

as some of these sub-recursive calls will be called more than once with the same arguments. Therefore, the returned features from each call are saved and simply used again if a call to **gen-all-topologies** with the same arguments is repeated. In one test, 35% of the calls were able to get the results from the dynamic programming results table. In practice, it seems that the feature extraction algorithm is fairly fast, even when extracting features with as many as five links.

4.1.3 Adding Abstraction to Improve Recall

Experiments show that the WSD system using ATSF's outperform the mean F-measure of 44.2% (see the results table 2, second to last row.) The best recall is 50.1%. The syntactic feature-extractor was extended to first extract the same features as before and, in addition, derive additional abstract features where a "*" or wildcard might take the place of a (non-targetword) lemma. It is important to do the bookkeeping that keeps track of how many literal features make up an abstract feature so when it comes time for feature selection we know the count and the conditional probability with which that abstract feature supports given senses. Table 1 and its caption give details of a real example from the training data.

The addition of wildcard features can make an especially noticeable difference when a sense of a word goes from having zero features when wildcards are not used to having one or more features with wildcard use. Table 1 shows such an example.

4.1.4 Minimum Abstraction Support

Table 1 shows an example of an abstract feature that has five literal features mapped to it. However, many abstract features are only spawned by a single literal feature from the training data (single-support abstract features.) At best, such abstraction features do not add new information and at worst they may add noise. Therefore, by default, abstract features with only one supporting literal feature have been removed. This aspect of the system is called over **abstraction protection** (OAP.)

4.1.5 Feature Selection Strategies Based on Abstraction Level

It could be argued, that constructing abstract features comes with the risk of overgeneralization. One way to control this risk is by use of OAP. Another way to control this risk is in the feature selection process. Section 2.3 specifies that one of the conditions that a feature must meet to be selected is that its count must be greater than FC_{min}. With the addition of abstract features, the system now allows for different values of FC_{min} based on the level of abstraction in a feature. If the feature has n wildcards then that feature must have at least a minimum count of $FC[\mathrm{n}]_{min}$ to not be eliminated.

5 Experiments and Results

All results discussed below are listed in table 2.

Abstract Features: The experimental results back the importance of abstraction. The results table is divided into three horizontal sections based on the number of wildcards (0, 1, or 2) used in the experiments. The F-measure of every 2-wildcard experiment is greater than the the F-measure of every 1-wildcard experiment just as the 1-wildcard's are greater than the 0-wildcard's. This seems like strong evidence that abstraction is invaluable tool for increasing both precision and recall.

OAP: The use of OAP is supported experimentally as the system does slightly worse when single-support abstract features are not removed. Experiment number 13 has the exact same parameters as the best performing experiment 14, but in 13 OAP is off while in 14 it is on. Experiment 13 does slightly worse than 14 probably because of the extra noise.

Basing FC_{Min} on abstraction level: Experiments 12 and 14 have all parameters exactly the same except $FC[1]_{min}$ and they come up with different results lending weight to the proposition that such control could be useful. Further experiments need to be run to determine the scope of variation in results as a result of different settings of $FC[\mathrm{n}]_{min}$ for different values of n.

Best performance: Experiment 14 performed best.

Again, Lin's system is one of the few SENSEVAL systems that only uses syntactic features and thus should be quite comparable with our system. Lin only gave his results in the course-grained scale. The scores in table 2 are all in terms of the fine-grained scale. Therefore, Lin's results are not included in table 2. His most comparable experiment achieved a coarse-grained F-score of 67%. The best coarse-grained F-score, of the system described here, was 61.2%.

6 Conclusion

The Yarowsky et al. WSD system achieved the highest official score with an F-measure of 64.2%. Their system used six types of features and a voting-scheme machine-learner that used five base machine-learners. Given that the system described here is using only a single type of feature, syntactic, and a single type of machine-learner, SVM, coming within 11.1% of the top score is quite respectable.

Lin's system, that used LSF's, performed better then the ATSF's despite our expectations to the contrary. One reason that ATSF's might not have outperformed Lin's features could be because Lin is using a nearest neighbor (NN) learner and Lin may be able to compose many simpler features to build up a similar picture to a fewer number of the more complex ATSF's. If this is the case, then ATSF's would not seem to offer any advantages over LSF's. The fact that Lin's system did significantly better then this system might say something about the use of nearest-neighbor , compared to SVM's. Lin's system built up a case library and thus did not forget any data quirks. This might be important in an area like WSD , where there is not a lot of supervised training data available, at this point.

There are some advantages to the ATSF's representation of the data. If one thinks of a feature as representing properties of the data then ATSF's can represent such properties more compactly. Several of Lin's features might be required to represent the same data property as one ATSF. Especially where it is important for humans to interpret the features culled from the data, the ATSF representation might be more efficient for humans to deal with.

Acknowledgements

The word sense disambiguation architecture was jointly constructed with David Bell. We would like to thank the Capital Markets CRC and the University of Sydney for financial supported and everyone in the Sydney Language Technology Research Group for their support.

References

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

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

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

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

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 40–47, San Francisco. Morgan Kaufmann Publishers.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

David Yarowsky, Silviu Cucerzan, Radu Florian, Charles Schafer, and Richard Wicentowski. 2001. The johns hopkins SENSEVAL2 system descriptions.

D. Yarowsky. 2000. Hierarchical decision lists for word sense disambiguation.

Feature Type	**ATSF**	**example phrase**	**sense of "blind"**
literal	{::bit=n {attr::blind=a} {mod::of=prep}}	blind bit of	[unassignable]
literal	{::fear=n {attr::blind=a} {mod::of=prep}}	blind fear of	irrational
literal	{::force=n {attr::blind=a} {mod::of=prep}}	blind force of	irrational
literal	{::hatred=n {attr::blind=a} {mod::of=prep}}	blind hatred of	irrational
literal	{::pursuit=n {attr::blind=a} {mod::of=prep}}	blind pursuit of	irrational
abstract	{::*=n {attr::blind=a} {mod::of=prep}}	blind * of	irrational

Table 1: The first five rows above hold literal features from the training data for the word "blind." These 5 literal features did form one abstract feature, shown in the last row. The first example was not observed with the same sense as the other literal features. The occurrence count of the abstract feature that was formed from the literal features is five and since four out of five of the senses of the literal feature, are of the same sense (the sense of "blind" as being *irrational*), the conditional probability that this abstract feature supports that sense is 0.80. Under the feature selection parameter settings of the experiment that had the best performance, the minimum count for a one-wildcard feature was 4 and the conditional probability cut-off was 66%. Therefore, the abstract feature shown above would have been selected.

	Parameters							**Results**		
Experiment#	**links**	***'s**	$FC[\mathbf{n}]_{min}$			CP_{min}	**OAP**	**Precision**	**Recall**	**F-measure**
			0	**1**	**2**					
1	1	0	3	-	-	75	-	0.505	0.482	0.493
2	2	0	3	-	-	75	-	0.516	0.501	0.508
3	3	0	3	-	-	75	-	0.515	0.500	0.507
4	4	0	3	-	-	75	-	0.515	0.500	0.507
5	4	0	3	-	-	80	-	0.511	0.492	0.501
6	5	0	3	-	-	75	-	0.515	0.500	0.507
7	2	1	3	4	-	75	yes	0.530	0.514	0.522
8	3	1	3	4	-	75	yes	0.527	0.511	0.519
9	4	1	3	4	-	75	yes	0.528	0.512	0.520
10	5	1	3	4	-	75	yes	0.528	0.512	0.520
11	3	2	3	4	4	51	yes	0.533	0.517	0.525
12	3	2	3	3	4	66	yes	0.538	0.521	0.529
13	3	2	3	4	4	66	no	0.539	0.522	0.530
14	3	2	3	4	4	66	yes	0.539	0.523	**0.531**
15	3	2	3	4	4	75	yes	0.536	0.520	0.528
16	4	2	4	4	4	75	yes	0.533	0.517	0.525
17	4	2	3	4	4	75	yes	0.534	0.518	0.526
18	5	2	3	4	4	75	yes	0.534	0.518	0.526
SENSEVAL-2 competition baseline:								0.476	0.476	0.476
SENSEVAL-2 competition mean:								0.459	0.425	0.442
SENSEVAL-2 competition best:								0.642	0.642	0.642

Table 2: Experimental results

AnswerFinder — Question Answering by Combining Lexical, Syntactic and Semantic Information

Diego Mollá and **Mary Gardiner**
Centre for Language Technology
Division of Information and Communication Sciences
Macquarie University
Sydney, Australia
diego@ics.mq.edu.au and gardiner@ics.mq.edu.au

Abstract

We present a question answering system that combines information at the lexical, syntactic, and semantic levels, in the process to find and rank the candidate answer sentences. The candidate exact answers are extracted from the candidate answer sentences by means of a combination of information-extraction techniques (named entity recognition) and patterns based on logical forms. The system participated in the question answering track of TREC 2004.

1 Introduction

Question answering is an area that is becoming increasingly active in research and is currently being deployed into practical applications. Research in question answering has recently been fostered by large-scale programs like AQUAINT[1] and evaluation frameworks like TREC[2], NTCIR[3], and CLEF[4]. Such research and the current need to cope with large volumes of text has led various companies to produce practical question answering systems. For example, research groups from Microsoft, IBM, NTT, Oracle, and Sun have participated in the question answering track of TREC. In addition, there are several attempts to provide question-answering extensions to the current Web search engines, with demos available by MIT[5], LCC[6], and BrainBoost[7], among others.

AnswerFinder is an open-domain question answering system that combines information at the lexical, syntactic, and semantic levels in various stages to find the exact answer to the user question. This paper describes the AnswerFinder system as it stood at the time of the TREC 2004 question answering track. Section 2 introduces TREC and the question answering track. Section 3 describes the architecture of the system. Section 4 details the function of each module within the AnswerFinder system. Section 5 gives the system performance on the TREC 2003 question set. Section 6 mentions related work and Section 7 lists problems with the AnswerFinder system that should be addressed in the near future.

2 The TREC 2004 Question Answering Track

The Text REtrieval Conference (TREC) started in 1992 as part of the TIPSTER text program. A fundamental goal of the conference is to provide an evaluation framework for the comparison of the results of independent information retrieval systems. The concept of information retrieval is to be understood in a broad sense, and this conference has developed various tracks that focus on specific areas of information retrieval, such as ad-hoc (the name given to document retrieval), routing, speech, cross-language, web, video, and very large corpora (Voorhees, 2003).

The question answering track started in 1999 and ever since its creation it has been the most popular track. Every year the complexity and difficulty of the task increases. Thus, in 1999 the competing systems were asked to retrieve small snippets of text containing the answer. The questions were designed by the participants and the answer was guaranteed to be in the text corpus. In the 2004 competition, in contrast, the questions were extracted from logs of real questions, the answer is not guaranteed to be in the corpus, and the systems were asked to find the exact answers of factoid questions and list questions. The questions were grouped into targets, each target containing fact-based questions and list questions (explicitly marked as such), plus a question asking to find any other

[1] www.ic-arda.org/InfoExploit/aquaint/index.html
[2] trec.nist.gov
[3] research.nii.ac.jp/ntcir/index-en.html
[4] clef.iei.pi.cnr.it
[5] www.ai.mit.edu/projects/infolab/
[6] www.languagecomputer.com/
[7] www.brainboost.com/

information relevant to the target. The questions were encoded in XML as shown in Figure 1. In this example, the target is *Fred Durst*, so question with ID number 2.2 in the figure is asking *What record company is Fred Durst with?*.

```
<target id = "2" text = "Fred Durst">

<qa>
 <q id = "2.1" type="FACTOID">
  What is the name of Durst's group?
 </q>
</qa>

<qa>
 <q id = "2.2" type="FACTOID">
  What record company is he with?
 </q>
</qa>

<qa>
 <q id = "2.3" type="LIST">
  What are titles of the group's releases?
 </q>
</qa>

<qa>
 <q id = "2.4" type="FACTOID">
  Where was Durst born?
 </q>
</qa>

<qa>
 <q id = "2.5" type="OTHER">
  Other
 </q>
</qa>

</target>
```

Figure 1: A hand-made example of a group of questions using the TREC 2004 format

The corpus of supporting text was the AQUAINT corpus, which comprises over 1 million news articles taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. This corpus is not large in comparison with the terabites of text available via the Internet, but it is still large enough to require the need to resort to shallow-processing preselection methods before performing a real attempt to find the answer.

3 System Overview

The question answering procedure used by AnswerFinder follows a pipeline structure that is typical of rule-based question answering systems. The process is outlined in Figure 2 and is as follows:

1. All questions are normalised, so that *What record company is he with?* in Figure 1 becomes *What record company is Fred Durst with?*
2. The questions are classified into types based upon their expected answer. So the question *How far is it from Mars to Earth?* would be classified as a "Number" question as it expects a numeric value in response.
3. 100 candidate answer sentences are extracted from the corpus.
4. The 100 sentences are re-scored based upon their word overlap, grammatical relations overlap, and flat logical form overlap with the question text.
5. Exact answers —fragments like *416 million miles*— are extracted from the candidate answer sentences.
6. The exact answer list is sorted, re-scored and filtered for duplicate exact answers.
7. A number of exact answers from the top of the list are selected, depending on the question type.

AnswerFinder uses the following knowledge sources to analyse the question and to select from among possible answers:

Named entity data generated by the GATE system (Gaizauskas et al., 1996), marking pieces of text in the AQUAINT corpus as one of the types *Date*, *Location*, *Money*, *Organization*, and *Person*. These data are generated off-line before any question is processed. GATE's analysis was extended with a simple set of regular expressions that detect numbers as well.

The list of preselected documents provided by the US National Institute of Standards and Technology (NIST), containing for each target entity, the 1,000 top scoring documents for that entity. NIST co-sponsors the TREC conferences and it obtained the list of preselected documents by running the target query through the PRISE (Harman and Candela, 1990) document retrieval system.

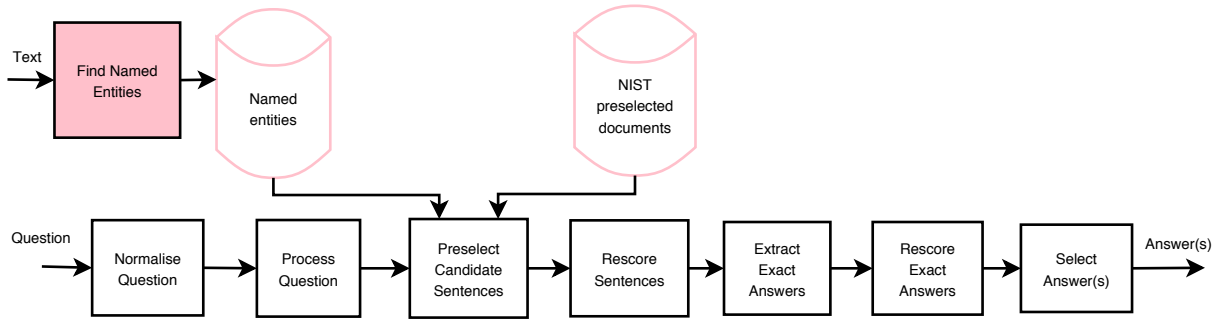


Figure 2: System overview

4 Modules

4.1 Question Normalisation

AnswerFinder relies heavily on the common information found between a question and the candidate answer sentence. Therefore questions like *What record company is he with?* need to undertake an anaphora resolution process to determine that *he* in fact refers to *Fred Durst*.

Questions in the TREC 2004 competition co-referred with previous questions or with their target in a number of ways.

Questions might co-refer with their target pronominally:

> **Target**: *Fred Durst*
>
> **Q**: *What record company is he with?*

Questions might co-refer with their target using a definite noun phrase:

> **Target**: *Club Med*
>
> **Q**: *How many Club Med vacation spots are there worldwide?*

Questions might co-refer with another question:

> **Target**: *Fred Durst*
>
> $\mathbf{Q}_{2.1}$: *What is the name of Durst's group?*
>
> $\mathbf{Q}_{2.3}$: *What are titles of the group's releases?*

Finally, questions may relate to their target associatively, that is, there may not be a direct co-reference:

> **Target**: *Heaven's Gate*
>
> **Q**: *When did the mass suicide occur?*

AnswerFinder normalises questions in the first case, where the question co-refers with the target pronominally. It performs a simple replacing of pronouns in the question with the target text, forming a regular plural and possessive where necessary, as shown in Table 1.

Finally, "other" type questions, which were of the generic form *other*, were transformed into *What is TARGET?* so that question 2.5 in Figure 1 is transformed into *What is Fred Durst?* This was a crude attempt at doing something useful with the "other" type questions. Clearly a more detailed processing of these questions is required.

4.2 Question Classification

Particular question words signal particular named entity types required as a response. The example below requires a person's name in response to the question:

> *Who founded the Black Panthers organization?*

AnswerFinder uses a set of 29 regular expressions to determine what named entity type a question requires in its response from the list *person, date, location, money, number, city, organization, percent, country, state, river, name, unknown*. The regular expressions were developed with the question set from TREC 2002, and they produced an accuracy of 78.6% correct classifications. This figure is lower than the one reported by other systems like the ones by Paşca and Harabagiu (2001) or Zhang and Lee (2003), each of which reported an accuracy of 90% or over. The question classification module clearly needs further refinement, but an evaluation with the question set from TREC 2003 showed an accuracy of 77%, thus indicating that the regular expressions generalise well.

*What record company is **he** with?*	$\longrightarrow$	*What record company is **Fred Durst** with?*
*How many of **its** members committed suicide?*	$\longrightarrow$	*How many of **Heaven's Gate's** members committed suicide?*
*In what countries are **they** found?*	$\longrightarrow$	*In what countries are **agoutis** found?*

Table 1: Examples of pronoun resolution performed by AnswerFinder

4.3 Candidate Sentence Extraction

Given the set of AQUAINT documents preselected by the NIST document retrieval system, AnswerFinder selects 100 sentences from these documents as candidate answer sentences.

Candidate sentences are selected in the following way:

1. The 1,000 preselected documents provided by NIST for each target are split into sentences by means of a simple sentence splitting process.
2. Each sentence is assigned a numeric score: 1 point for each distinct non-stopword overlapping with the question string, and 10 points for the presence of one or more named entities of the right type. This way we reward heavily the presence of a string of the expected answer type.
3. The 100 top scoring sentences are returned as candidate answer sentences.

As an example of the scoring mechanism, consider this question/sentence pair:

> **Q**: *How far is it from **Mars** to **Earth**?*
>
> **A**: *According to evidence from the SNC meteorite, which fell from **Mars** to **Earth** in ancient times, the water concentration in Martian mantle is estimated to be **40 ppm**, far less than the terrestrial equivalents.*

The question and sentence have 2 shared non-stopwords: *Mars* and *Earth*. Further, this sentence has a named entity of the required type (Number): *40 ppm*, making the total score for this sentence 12 points.

4.4 Sentence Re-Scoring

The goal of all the above modules is to reduce the corpus of text to a list of the 100 sentences with highest likelihood to contain an answer. The sentence re-scoring module uses a combination of lexical, syntactic, and semantic information to perform a more detailed analysis of these sentences:

lexical: The combined word overlap and named entity score.

syntactic: The grammatical relation overlap score.

semantic: Overlaps with flat logical form patterns.

We have seen the use of lexical information in Section 4.3. Below we will see the use of grammatical relations and flat logical form patterns, and the final combinations used in TREC 2004.

4.4.1 Grammatical Relation Overlap Score

The grammatical relations were initially devised by Carroll et al. (1998) as a means to normalise the output of parsers for their comparative evaluation. The set of grammatical relations represent some of the common relations that exist between the words in a sentence, a selection of which is shown in Table 2. To build the grammatical relations of questions and answer candidate sentences, AnswerFinder processes the output of the Connexor Dependency Functional Grammar, which is a dependency-based robust parser with a wide-coverage grammar of English (Tapanainen and Järvinen, 1997). Below is an example of the grammatical relations of a question and an answer candidate sentence.

> **Q**: *How far is it from Mars to Earth?*
> **(subj be it _)**
> (xcomp from be mars)
> (ncmod _ be far)
> (ncmod _ far how)
> **(ncmod earth from to)**
>
> **A**: *It is 416 million miles from Mars to Earth.*
> **(ncmod earth from to)**
> **(subj be it _)**
> (ncmod from be mars)

Relation	*Description*
CONJ(type,head+)	Conjunction
MOD(type,head,dependent)	Modifier
CMOD(type,head,dependent)	Clausal modifier
NCMOD(type,head,dependent)	Non-clausal modifier
DETMOD(type,head,dependent)	Determiner
SUBJ(head,dependent,initial_gr)	Subject
OBJ(head,dependent,initial_gr)	Object
DOBJ(head,dependent,initial_gr)	Direct object
XCOMP(head,dependent)	Clausal complement without an overt subject

Table 2: Grammatical relations used in this paper

(xcomp _ be mile)
(ncmod _ million 416)
(ncmod _ mile million)

The score is the number of relations shared between question and sentence. In the example above, the overlap between the grammatical relations of question and candidate sentence is 2, corresponding to the two grammatical relations marked in boldface.

4.4.2 Flat Logical Form Patterns

In previous research we have developed a flat notation for the logical forms of sentences and a method to produce the logical forms from arbitrary sentences by traversing their syntactic structures (Mollá, 2001; Mollá and Hutchinson, 2002). These flat logical forms have been used to determine the likelihood that a sentence contains the answer by checking the semantic similarity of the question with the sentence. In a similar fashion to grammatical relations, the semantic similarity of two sentences is the number of logical terms shared between them. Thus if we have the following logical forms:

Q: *What is the population of Iceland?*
object(iceland, o6, [x6])
object(population, o4, [x1])
object(what, o1, [x1])
prop(of, p5, [x1, x6])
A: *Iceland has a population of 270000*
dep(270000, d6, [x6])
object(population,o4,[x4])
object(iceland,o1,[x1])
evt(have,e2,[x1,x4])
prop(of,p5,[x4,x6])

The semantic similarity between the two sentences is 2, as the number of overlaps between the logical form of question and answer is 2 (overlap shown in boldface). Note that the computation of the overlap is complicated by the fact that logical terms include variables and it is necessary to keep the relation between the variables in the overlapping terms. Thus, in the example above, the variable **x1** in the question terms corresponds with **x4** in the answer candidate sentence and therefore whenever **x1** is used in the question, **x4** must be used in the answer. A simple process of Prolog unification suffices to match the variables of the question terms with those of the sentence terms, by converting the question term variables into real Prolog variables.

Since there are several ways to answer a question, for TREC 2004 we have developed a set of patterns to capture the expected logical form of sentences that contain the answer to questions. Below is the matching pattern associated with the template that we labelled as "what2" and one of its replacement patterns:

Template "what2" :
Pattern:
object(ObjX,VobjX,[VeX]),
object(what,_,[VeWHAT]),
object(ObjY,VobjY,[VeWHAT]),
prop(of,_,[VexistWHAT,VeX])
Replacement 1:
dep(ANSWER,ANSW,[VeANSW]),
prop(of,_,[VeY,VeANSW]),
object(ObjX,VobjX,[VeX]),
evt(have,_,[VeX,VeWHAT]),
object(ObjY,VobjY,[VeY])

Borrowing the notation of Prolog variables, the above template uses forms in uppercase or "_" to express the slots that can unify with logical form components. As the logical form of *What is the population of Iceland?* matches the

pattern above (we use standard Prolog unification to perform the matching), then its logical form is transformed into:

Q: *What is the population of Iceland?*
dep(ANSWER,ANSW,[VeANSW]),
prop(of,_,[VeY,VeANSW]),
object(iceland,o6,[x6]),
evt(have,_,[x6,x1]),
object(population,o4,[VeY])

The semantic similarity between this logical form and the one of *Iceland has a population of 270000* is now 5, since all five terms of the modified question logical form can be found in the logical form of the answer and all variables unify.

In addition to returning the overlap between a candidate sentence and a matching answer pattern, AnswerFinder uses the instantiation of the ANSWER variable to determine the answer: "270000" in the case of our example.

The introduction of flat logical form patterns parallels the use of patterns based on regular expressions, but using the logical level of a sentence instead of the surface level. This way it is hoped that less patterns are required to cover a broader range of sentences. In practice, however, the difficulty to read logical forms by humans slows down the production of patterns and replacements. As a result, a small set of 10 patterns were developed for our experiments in TREC 2004. As we can see in Table 3, most of the questions from the TREC 2004 test set were covered by only 4 template patterns and there was an important number of questions that did not trigger any pattern.

Template ID	*Num.*	Template ID	*Num.*
howmany1	0	how1	1
howmany2	0	who_generic	39
what2	3	what_generic	116
what3	0	what_noun	69
what6	1	*no match*	78
when1	47		

Table 3: Number of questions triggering each template; a question may trigger several templates

The patterns that were triggered most frequently were generic patterns that were introduced to maximise the coverage of the pattern set. For example, the most frequent pattern, "what_generic", is defined so as to allow any noun to replace the word *what*:

Template "what_generic" :
Pattern:
object('what',_,[XWho])
Replacement:
object(_,ANSWER,[XWho])

4.5 Exact Answer Extraction, Filtering and Scoring

Having selected and re-ranked the 100 top-scoring candidate sentences, AnswerFinder then selects exact answer strings from within them. AnswerFinder combines the use of named entities with that of logical form patterns:

1. For each candidate sentence, extract all named entities that match the question classification.
2. For each candidate sentence, extract ANSWER values from any matching flat logical form pattern.

Exact answers are scored as follows:

1. If the exact answer is a named entity, its score is the score of the candidate sentence it is found in.
2. If the exact answer is an ANSWER value from a flat logical form pattern, its score is the score of the candidate sentence it is found in.
3. If the exact answer is both a named entity and an ANSWER value from a flat logical form answer pattern, its score is twice the score of the candidate sentence it is found in.

If the same string is extracted from two answer sentences the score becomes the sum of the scores of the duplicate answers. This way answer redundancy is rewarded.

4.6 Exact Answer Selection

AnswerFinder selects the answers depending on the type of question:

Factoid questions requiring exactly one answer: return the top scoring answer; or if there are no answers with a score more than 0, return "NIL" indicating that there were no answers.

List questions and "other" questions requiring a number of answers: return all exact answers within a threshold difference in score with the top score. If there are no exact answers with a score of more than 0, return the top scoring candidate sentence.

5 Performance

After testing several combinations of lexical, syntactic, and semantic information (see the work by Mollá (2003) for the sort of analysis that we performed), AnswerFinder used two combinations of scores for the runs submitted to TREC 2004:

3gro+lfo 3 times the grammatical relation overlap score added to the flat logical form pattern overlap score. This combination was chosen because it gave the best results in our preliminary experiments with question sets taken from past TREC QA conferences.

lfo The flat logical form pattern overlap score.

Although not explicitly expressed in the above combinations, lexical information is used implicitly because the scoring is based on the output of a preselection module that did use solely lexical information (word overlap and named entities), as we have seen in Section 4.3.

In a preliminary analysis of the system we used the answer patterns provided by Ken Litkowsky via NIST. These answer patterns cover all the answers found by the systems participating in TREC 2003. We tested our system with the TREC 2003 questions and checked the output of the sentence re-scoring module and the final output of the system. We found that the re-scoring module gave the highest score to a sentence containing the answer about 20% of times. In contrast, the final system returned a correct and exact answer about 5% of times.

The results of our participation in TREC 2004 are significantly better. In all runs, the accuracy of the factoid questions is 10%, the F-score of the list questions is 0.08, and the F-score of the "other" questions is 0.09. Since the system was not fine-tuned for the list or "other" questions only the results of the factoid questions need to be considered. We believe that the reason for the better results of the factoid question with respect to our preliminary analysis is that Litkowsky's patterns that we used did not cover all cases of good answers. In fact, we tried the patterns on the answers submitted to TREC 2004 and we obtained an accuracy of 8.59%, which is between the accuracy given by our preliminary experiments and the one given by the TREC human assessors. Comparatively with the other systems participating in TREC 2004, the results are below the median of the results returned by all the systems (which was 17%). Also, surprisingly, all of our runs had virtually the same results. These unusual results led us to suspect that a chain of bugs may have made the system ignore the information provided by the logical form patterns. Currently we are analysing the results.

6 Related Work

AnswerFinder as it stood in TREC 2004 differs from previous versions in several aspects. First of all, now AnswerFinder uses the Named Entity data that has been pre-calculated on the entire AQUAINT corpus. This way the system does not need to spend precious time during the on-line stage when the user is waiting for the answer of the question. Also, in contrast with AnswerFinder's participation in TREC 2003 where it focused on the extraction of passages containing the answer, now AnswerFinder extracts exact answers and attempts to answer list and definition questions. In the process, AnswerFinder uses a set of templates based on patterns of logical forms.

The overall architecture of AnswerFinder is similar to that of other question answering systems. The aim is to gradually reduce the amount of text to process through several levels of increasing complexity. We use an information retrieval system to preselect the documents and information from named entities and the expected answer type obtained from the question to reward the sentences that may contain the answer. One difference that sets our system apart from the majority is the use of logical forms in the process to further scope the sentences that are most likely to contain the answer. Other question answering systems use logical forms (Harabagiu et al., 2001, for example) that were developed independently from our research.

But the main difference with respect to other systems is the use of patterns derived from logical forms to determine the exact answer. A baseline method that would return the text tagged by the named entity recogniser has been used by various systems. Adding further com-

plexity, systems like the one developed by Echihabi et al. (2004) use patterns based on named entities and parts of speech. However, we are not aware of any other system besides AnswerFinder that tries to use logical form patterns.

7 Conclusions and Further Work

AnswerFinder is a question answering system that uses a combination of lexical, syntactic, and semantic information to find the answer to the user question. An early version of this system participated in the passages section of the 2003 TREC question answering track. After the equivalent of only 55 person-hours work, the system ranked above the median of the seven participating systems. For TREC 2004 we have included a named entity recogniser and a process to find exact answers that uses a combination of patterns based on logical forms and named entities.

Current and future work focuses on the refining of the candidate sentence scoring, exact answer scoring, and pattern development. We also plan to work on a more detailed processing of list and definition questions.

For the refining of the sentence scoring, we are exploring the use of weighted measures for different types of terms in the flat logical forms. We are also exploring the integration of graph-based methods such as the ones developed by (Montes-y-Gómez et al., 2001).

For the exact answer scoring, we are developing further logical form patterns to increase their coverage. We will also explore fuzzy matching methods so that every question will match at least one pattern.

To facilitate the discovery and development of logical form patterns, we are studying methods to increase the readability of the flat logical forms by converting them into graph structures.

References

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

Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy, Daniel Marcu, Eric Melz, and Deepak Ravichandran. 2004. How to select an answer string? In Tomek Strzalkowski and Sanda Harabagiu, editors, *Advances in Textual Question Answering*. Kluwer.

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.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, and Răzvan Bunescu. 2001. Answering complex, list and context questions with LCC's question-answering server. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC 2001*, number 500-250 in NIST Special Publication. NIST.

Donna K. Harman and Gerald Candela. 1990. Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science*, 41(8):581–589.

Diego Mollá and Ben Hutchinson. 2002. Dependency-based semantic interpretation for answer extraction. In *Proc. 2002 Australasian NLP Workshop*.

Diego Mollá. 2001. Ontologically promiscuous flat logical forms for NLP. In Harry Bunt, Ielka van der Sluis, and Elias Thijsse, editors, *Proceedings of IWCS-4*, pages 249–265. Tilburg University.

Diego Mollá. 2003. Towards semantic-based overlap measures for question answering. In *Proc. ALTW03*, pages 130–137, Melbourne.

Manuel Montes-y-Gómez, Alexander Gelbukh, and Ricardo Baeza-Yates. 2001. Flexible comparison of conceptual graphs. In *Proc. DEXA-2001*, number 2113 in Lecture Notes in Computer Science, pages 102–111. Springer-Verlag.

Marius A. Paşca and Sanda M. Harabagiu. 2001. High performance question answering. In *Proc. SIGIR'01*, New Orleans, Luisiana, USA. ACM.

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

Ellen M. Voorhees and Lori P. Buckland, editors. 2003. *The Twelfth Text REtrieval Conference (TREC 2003)*, number 500-255 in NIST Special Publication. NIST.

Ellen M. Voorhees. 2003. Overview of TREC 2003. In Voorhees and Buckland (Voorhees and Buckland, 2003).

Dell Zhang and See Sun Lee. 2003. Question classification using support vector machines. In *Proc. SIGIR 03*. ACM.

Using WordNet Domains In A Supervised Learning Word Sense Disambiguation System

David Bell and Jon Patrick
Sydney Language Technology Research Group
School of Information Technologies
University of Sydney
Sydney, Australia
{dbell,jonpat}@it.usyd.edu.au

Abstract

This paper describes the impact of introducing domain information, obtained from context words, into the process of supervised learning for word sense disambiguation. A word sense disambiguation system is described in which many features can be combined to create a rich feature extraction system. Two Wordnet Domain feature extractors are created for this system and are tested with a combination of other context features. A comparison is made between real and binary valued domain feature vectors. Support vector machines are used as a machine learning tool and two different kernels with various parameters are compared in order to find the ideal learning model for this task and data. Results obtained over the Senseval 2 test data suggest that Wordnet domains perform relatively well as single features but when combined with other context features provide little benefit. It is argued other authors obtain superior results as their methods are tuned to the data set.

1 Introduction

Word sense disambiguation (WSD) is the task of assigning the most appropriate meaning, or sense, to a polysemous word in a particular context. The fields of machine translation, document classification, knowledge acquisition and many others all benefit from having information about the meaning of a particular word, and word sense disambiguation greatly increases the accuracy of the process of associating meaning with these words.

The best results in WSD, particularly over a limited sample of words, have been achieved using supervised learning techniques. These techniques involve the extraction of features from the context of the target word in order to build a feature vector that is then used to create a machine learning classifier. Supervised learning WSD systems usually build a collection of "Word Experts" that specialise in classifying one particular word into its different sense classes.

Supervised learning systems require the construction of a feature vector for each example to be classified. This vector represents the chosen features for that particular example. This vector is then compared with those of training examples to decide on the classification according to a learnt model. The method of this comparison depends on the particular machine learning technique.

Wordnet Domains were first introduced for word sense disambiguation by (Magnini et al., 2002) as part of the Senseval 2 word sense disambiguation task. The system used a technique in which a series of domain vectors were built of length equal to the number of domains to be considered. In this case all 43 domains in the WordNet domains database were considered. The series of vectors consisted of one text vector and several sense vectors. The text vector is a domain vector extracted from the context window around a targeted ambiguous word. Given a set of domains $D = \{D_1, D_2, ..., D_n\}$, a text T and a word at position p, the text vector T_p will be the n-dimensional vector such that the component i is the relevance of D_i for T at the position p. A sense vector s is a domain vector extracted from a word sense. Its length represents the frequency of the occurrence of that sense, and its direction represents the 'mean' vector of the texts where the sense usually occurs. A sense vector is built as the sum of the text vectors of the contexts in the training set containing the sense. The disambiguation procedure of a word occurrence T_p consists of a simple comparison between the text vector T_p and the sense vectors of the word itself. This is done using a dot product between T_p and each sense vector. The result is a ranked list of sense vectors for T_p, a sense is then selected by applying a cutoff. This method has the advantage of

setting the cutoff to decide when a good match occurs. This allows this system to be adjusted to boost precision at the cost of lower recall. The drawback of this system is that it does not allow for any other semantic or syntactic information to be included in the disambiguation decision. Also this system does not allow for other more sophisticated machine learning techniques to be applied such as support vector machines, or maximum entropy. This paper demonstrates a system in which a feature vector is created for each example of a word, which holds domain information for the context of that word and can also hold other information about that example. This feature vector can then be used for many machine learning techniques and allows for experimentation into combining domain information with other features for the purpose of word sense disambiguation.

2 Wordnet Domains

Wordnet Domains is an extension to Wordnet by (Magnini et al., 2002) widening the coverage of domain labels in the existing lexicon. In Wordnet Domains, Wordnet synsets have been annotated with one or more domain labels. A domain may include synsets of different syntactic categories and may also include senses from different sub-hierachies. Additionally domains can group senses of the same word into thematic categories which helps to reduce ambiguity.For example the common WSD example word, **bank**, has ten different senses in WordNet. These senses are shown in Table 2 with corresponding domains.

The WordNet Domains database is broken into a hierachical structure a sample of which is given in Table 1. Table 2 demonstrates the grouping of senses into thematic groups by domain. i.e. senses #2 and #7 which have similar meanings are grouped by the domains GEOGRAPHY and GEOLOGY.

2.1 One domain per discourse

The theory behind using WordNet domains for WSD is based upon the idea of one domain per discourse (ODD). This phenomena is in turn based upon the idea of one sense per discourse (Gale et al., 1992) and was verified by (Magnini et al., 2002). The ODD hypothesis claims that multiple uses of a word in a coherent portion of text tend to share the same domain. Hence using the ODD a word could be disambiguated by

Table 1: A sample of the WordNet Domains Hierarchy

doctrines	archaeology astrology history linguistics literature philosophy psychology art religion
art	dance drawing music photography plastic arts theatre
drawing	painting philately

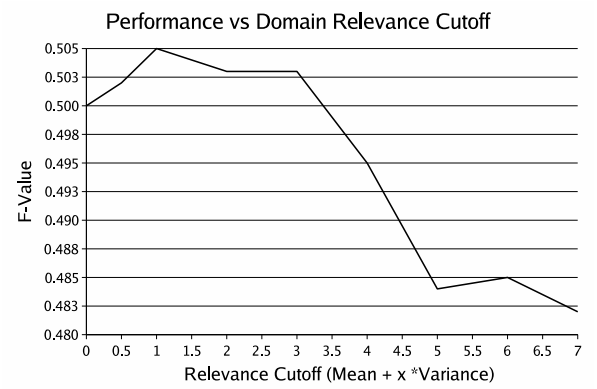


Figure 1: Performance of wsd system with respect to domain relevance cutoff

referring to the dominant domain of the context surrounding the word.

2.2 Domain relevance

(Magnini et al., 2002) introduces the notion of domain relevance. The relevance of a domain with respect to a text is represented as a positive real number. It finds that the frequency of a domain in a text does not imply its relevance in a text. They hypothesise that a domain is relevant for a text only if its frequency is significantly higher than in the texts unrelated with that domain.
In order to calculate the relation between frequency and relevance we measure the frequency

Table 2: An example of WordNet Domain entries for the senses of the word **bank**

Sense	Gloss	Domains
1	a financial institution ...	ECONOMY
2	sloping land	GEOGRAPHY, GEOLOGY
3	a supply or stock held in reserve	ECONOMY
4	a building	ARCHITECTURE, ECONOMY
5	an arrangement of similar objects	FACTOTUM
6	a container .... coin bank	ECONOMY
7	a long ridge or pile	GEOGRAPHY, GEOLOGY
8	the funds held by a gambling house	ECONOMY, PLAY
9	a slope in the turn of a road	ARCHITECTURE
10	a flight manoeuver	TRANSPORT

of each domain in the Semcor 2 corpus. Then using a normal distribution theorem it is said that if the frequency of a domain D computed on a text T is significantly higher than the mean frequency D on that corpus then D is relevant with respect to T. Our method varies from (Magnini et al., 2002) in that they calculated the domain statistics using the LOB corpus which unlike the Semcor corpus is not tagged for word sense. They included all domains of all senses of each word in the corpus in the statistical calculations where as we, due to the Semcor sense tagging, only needed to include the domains of the tagged sense of each word in the corpus. This lead to lower mean values and higher standard deviations for each domain over the corpus, and caused us to change our interpretation of *significantly higher than the mean frequency*, from exceeding the mean by two standard deviations (Magnini et al., 2002) to exceeding the mean by a single multiple of the variance. This value was found via experimentation with varying relevance cutoffs. (See Figure 1)
If a domain is not relevant in a particular text, its value is removed from the domain vector, see section 3.1. Domain relevance is only used in this manner with the real valued Wordnet Domains feature extractor, not the binary valued or selected wordnet domains feature extractors.(See Section 4)

3 WSD Infrastructure

Our word sense disambiguation system involves an extensible framework for feature extraction and feature vector construction. The framework is a 3 tiered construction:

Tier 1 - Feature Extractors This tier consists of the actual feature extractors. A feature extractor takes a context window and extracts features from it and builds a list of numerical attributes from them. A single feature extractor reserves a variable length section of the feature vector into which it inserts the values for the features it extracts. Each feature extractor extracts a class of features such as part of speech information, morphological information or domain information.

Tier 2 - Word Expert A word expert is a method that is responsible for the feature vector construction for a particular word. It organises the indivdual feature extractors input into a single feature vector. The word expert tier consists of a group of feature extractors for each word in the vocabulary.

Tier 3 - Word Expert Parliament The word expert parliament gathers parsed example data and sends examples to the word experts for processing of a particular word. The feature vectors returned from the experts are then appended with a class label (if in training stage) and the resulting feature vectors are then grouped into training or testing files and sent to the machine learner. The word expert parliament is also responsible for the construction of the disambiguation system for each target word, in that it creates the feature extractors and assigns them to a word expert according to a configuration file.

The framework is built and customised via an xml configuration file which specifies which feature extractors, and which parameters for those feature extractors, are used with each word in the vocabulary. The xml file also specifies the parameters for the machine learning. This sys-

tem can potentially be used for tagging tasks beyond word sense disambiguation e.g. POS tagging. It is essentially a feature vector construction system and by simply changing the class label on the training feature vectors, the system could be orientated to another task.

4 Features and Feature Extractors

This system has been tested with a number of feature extractors. Below is a description of the feature extractors used in this experiment:

4.1 Domain Features

The Wordnet Domains feature extractor is based on the idea that word sense is often dependant on the domain of the discourse. For example, if I am talking about computers, then the word "mouse" is more likely to refer to a computer mouse than a rodent. The domain of a discourse is established by finding the most prevalent domain amongst the word in that discourse. The domain of a particular word is found using the Wordnet Domains database (Magnini et al., 2002).

4.1.1 WordNet Domain Feature Extractor

Two versions of this feature extractor were created. The first builds a feature vector of length equal to the amount of domains at the selected depth of the WordNet Domains hierarchy and places a score in each position of this vector based upon the presence of that domain in the context window. This approach is similar to the approach taken by (Magnini et al., 2002) except that when used in our WSD infrastructure it can be combined with other features to provide a richer feature space.

4.1.2 Selected Domains Feature Extractor

The second selects the domains that will be considered based upon training data. This was done by passing over the Semcor 2.0 and Senseval2 training corpora looking for instances of the word of interest. When an occurrence of the "target" word was found the sense was noted along with the domains of the surrounding context words. This was continued until all occurrences of the word are considered. The domains selected are those that occur more often with one sense of the word than any other. These domains are considered to be indicative of the sense for that word.

In contrast to (Magnini et al., 2002) after this selection step we have an optimal set of domains from which to build our domain vector. It is hoped that this will help to reduce noise in the training.

Both of the domain information feature extractors can be set to produce binary (1 if a domain occurs, or 0 if not) and real valued (a value from 0-1 depending on significance of domain presence) vectors. Both feature extractors work on a context window of 4 sentences to the left and right of the sentence of the target word. When using the non-selected domain feature extractor domain relevance is taken into account and a dimension on the vector may be set to zero if the corresponding domain is deemed irrelevant (See section 2.2).

4.2 General Feature Extractors

For the purpose of this investigation into the a set of "General" feature extractors was constructed to provide some extra semantic and syntactic information for each word sense. These feature extractors form a basic system that has results by itself of 56% F-score for the lexical sample task, which is well above the average F Score for Senseval2 of 46.2%. The constituants of this base set of feature extractors are outlined in the next three sub-sections.

4.2.1 Context words

The context words feature extractor works in a similar way to the Selected Wordnet Domains feature extractor described above. It includes a pre-processing step in which words which are indicative of sense are identified using conditional probability. Note that this differs from the domain feature extractor as it considers the words themselves, not the domain of the word.

4.3 Collocations

The collocations feature extractor follows the basic idea of (Ng and Lee, 1996) where 9 word sequences around the target word are considered as collocations involving the word. These 9 word sequences are found by varying the left offset and right offsets of the context window around the target word. This creates word sequences of up to 4 words starting from up to 3 words to the left and ending up to 3 words to the right of the target word. See table 3.

Table 4: Results of experimentation for Senseval 2 eng-lex-sample using fine grained scoring

Feature Extractors	**Precision**	**Recall**	**F-Value**
LEMMA (baseline)	0.48	0.48	0.48
GENERAL	0.56	0.56	**0.56**
WND Binary	0.45	0.45	0.45
WND Real	0.51	0.51	**0.51**
PSWND Binary	0.47	0.46	0.47
PSWND Real	0.48	0.47	0.48
GENERAL + WND Real	0.56	0.56	**0.56**
GENERAL + PSWND Real	0.53	0.53	0.53
GENERAL + PSWND + WND Real	0.56	0.56	0.56
GENERAL + LEMMA	0.54	0.54	0.54
MAGNINI	0.67	0.25	0.36
WND Real Magnini Set	0.62	0.23	0.34

Table 3: Features for Collocations involving the word "interest" - (Ng and Lee, 1996). Left and Right indicate the position of the left and right boundaries of the context windows, with respect to the target word

Left	Right	Collocation Example
-1	-1	accrued interest
1	1	interest rate
-2	-1	principle and interest
-1	1	national interest in
1	2	interest and dividends
-3	-1	sale of an interest
-2	1	in the interest of
-1	2	an interest in a
1	3	interest on the bonds

A pre-processing step is also done with this feature extractor to identify the collocations that are indicative of sense for each word. The vector generated by this feature extractor has a position for each collocation identified which is set to 1 if the collocation exists in the example or 0 if it doesn't.

4.3.1 Other feature extractors

A number of other less sophisticated feature extractors where used to enhance the performance of the word experts. These included a part of speech identifier which considers the part of speech of the target word and context words, and a morphology feature extractor which considers the morphology of the target word and context words.

5 Method

In order to test the effect of using WordNet Domain information in supervised learning WSD a number of experiments were conducted, using different feature extractors in the framework described in section 3. Firstly the selected and the regular Wordnet domains feature extractors were tested with both binary and real valued feature vectors. Secondly the General features extractors were tested as a group with no input from the WordNet Domains feature extractors. Next, the General feature extractors were used in combination with the Wordnet Domains feature extractor, the Pre-Selected Wordnet Domains feature extractor seperatly and both together. Finally an experiment was run with only the lemma feature extractor, which simply provided the lemma of the target word as a feature. This was used in order to provide a further baseline for comparison.

5.1 Data

The data used for this research consisted of the Semcor2.0 corpus and the training and testing data from the Senseval 2 lexical sample task. Semcor is a completely word sense tagged version of sections of the Brown corpus, and thus provides a rich resource for training.

5.2 Parsing and Processing

All data was parsed using the Conexor parser. This parser tags for part of speech, morphology, as well as syntactic links. Processing involved converting the xml output of the parser to include sense information and then using our WSD infrastructure to build arff files for the Weka machine learning framework, these files consists of the feature vectors extracted by

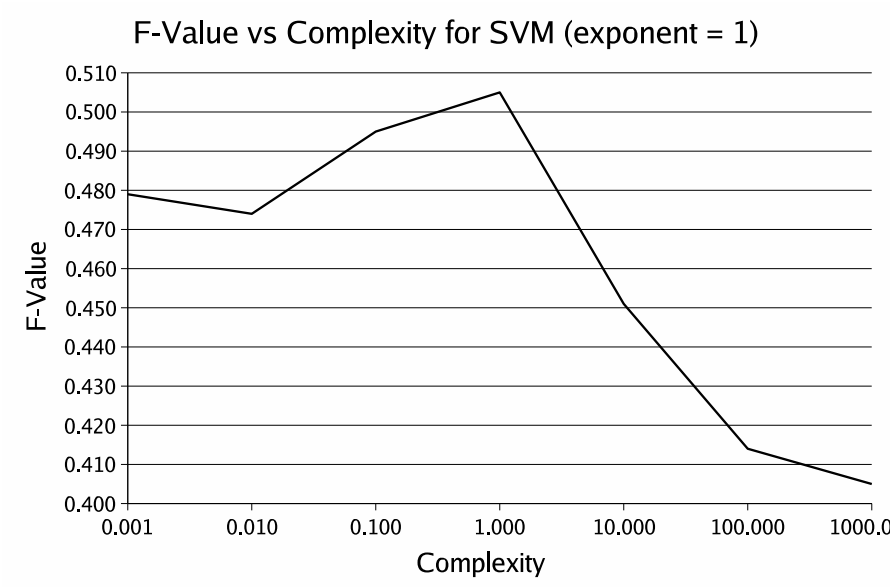


Figure 2: Performance of polynomial svm kernel with varying complexity and exponent of 1

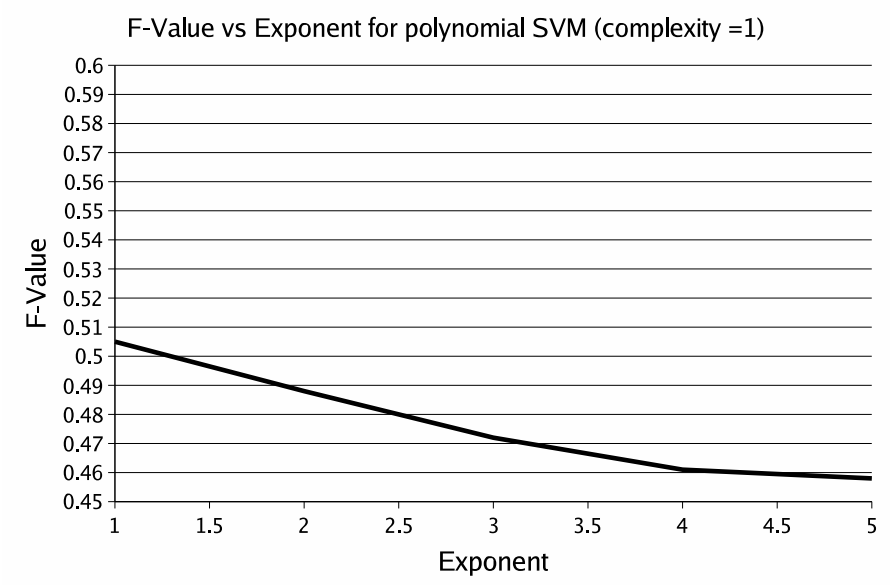


Figure 3: Performance of polynomial svm kernel with varying exponent and complexity of 1

the numerous feature extractors for each training/testing document.

5.3 Machine Learning

The Weka machine learning framework was used for this research. It involves a collection of machine learning algorithms and an infrastructure to aid in processing of data and results. Support Vector Machines were used as the machine learning algorithm and a variety of kernels and variables were tested in order to find the best parameters for the wsd task using domain information. The two kernels tested were a polynomial kernel and the RBF kernel. The polynomial kernel was tested while varying the exponent and complexity values, and the RBF kernel was tested whilst varying the gamma value. The results of these experiments are shown in figures 2,3 and 4. From this experimentation a decision was made to use a polynomial kernel with complexity 1.0 and exponent 1.0.

6 Results

In all 12 experiments were run. Each experiment consisted of building a supervised learning

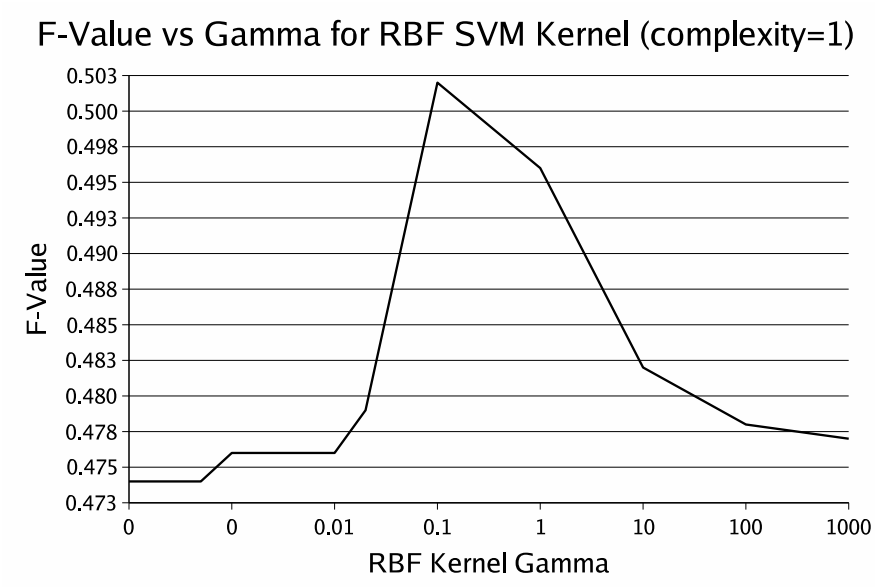


Figure 4: Performance of RBF svm kernel with varying gamma and complexity of 1

model by processing the training data (semcor and senseval) producing the feature vectors and using Weka to construct a model file. Then each model was tested using the Senseval 2 evaluation data for the english lexical sample section. The Senseval 2 scoring software was used to evaluate performance. When the PSWND and WND feature extractors were combined with the BASE feature extractors the real valued version of both these feature extractors was used as these were indicated to be better from earlier experiments. The results of these experiments are shown in table 4 which displays the precision, recall and f-value of the various feature extraction combinations for the Senseval2 lexical sample task. Table 4 also shows the results obtained by (Magnini et al., 2002) for comparison.

7 Discussion

Table 4 shows that the best way to use Wordnet domains is in a real valued non-selected fashion (WND Real). However domain information, when combined with other features has little impact (GENERAL + WND Real). This could be due to the fact that the information, relevant to word sense, provided by Wordnet Domains is already contained in the other features included in the GENERAL set, such as context words and collocations. However this argument can not be assured as it is uncertain if the same errors are being made by the GENERAL set and the Wordnet domains. The performance of the WND Real extractor is superior to that shown in (Magnini et al., 2002), f-value of 0.51 vs f-value of 0.36 and respectable in terms of Senseval 2 results (avg f-score 0.44, max f-score 0.64), however when the same experimentation is run over the subset of the Senseval data that the Magnini system attempted, the performance of our system is slightly lower (2% on f-value). This *Magnini set* is highly tuned to their method as they only make decisions on 37% of the testing corpus. This must almost certainly point to the fact that the Magnini system is a non-optimal solution across a wider data set.
The selected domains technique seems to not do well, and this is probably due to the small number of indicitive domains found during the selection process. This number could be increased by changing the conditional probability thresholds for selection. Real valued features do better than binary features, this is attributed to the richer model that a real valued feature incorporates and shows that the quantity of domain indicators in a context is important to word sense indication.
The diversity in sense determination is equally matched between WND and General features. The latter being four times larger than the former indicates a weakness in selectivity in the Wordnet domains database.

8 Conclusions

We have described a word sense disambiguation framework that can be used to easily combine

many different feature extraction techniques for the context of the word to be disambiguated. This framework was used to investigate the effect of using domain information, in particular the WordNet domain database information, on the word sense disambiguation task. This was done through a series of experiments involving two versions of a WordNet domain feature extractor and a set of General feature extractors. The performance was also measured against a simple feature extractor that took only the lemma of the target word. It was found that while WordNet domains can be used to provide a good feature for word sense disambiguation, when combined with other features such as context words and collocations the add-on effect is minimal.

9 Future Work

Further work could be done to examine the effect of using different levels of the domain hierarchy. This could change greatly the effect of WordNet domains on the WSD task. It would be interesting to see if changing the selection criteria on the Selected Domains feature extractor would improve the performance of this feature. (Magnini et al., 2002) reports better performance of WordNet domains on the Senseval all words task which has not been investigated in this research, thus it would be interesting to extend this system for the all words task. Another improvement might be made by using information such as part of speech to eliminate senses of context words in order to restrict the amount of possible domains for that word. Finally some investigation should be conducted into the errors being made by the classifier based on the GENERAL set of features and the classifier based on Wordnet domains features. This would indicate whether or not the two sets of features overlap, and if this overlap causes the combination of these two feature sets to be unsuccessful.

Acknowledgements

The word sense disambiguation architecture was jointly constructed with Ari Chanen. We would like to thank: The people at The Instituto Trentino di Cultura (ITC) for providing the WordNet Domains database; The Capital Markets CRC and The University of Sydney for financial assistance and everyone from The Sydney Language Technology Research Group for all their support.

References

Christopher J. C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Clara Cabezas, Philip Resnik, and Jessica Stevens. 2001. Supervised sense tagging using support vector machines.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press.

Philip Edmonds. 2000. Designing a task for senseval-2.

Gale, Church, and Yarowsky. 1992. One sense per discourse. In *4th ARPA Workshop on Speech and Natural Language Processing*, pages 233–237, NY. Harriman.

A. Kilgarriff and J. Rosenzweig. 2002. English senseval: Report and results.

Adam Kilgarriff. 1998. SENSEVAL: An exercise in evaluating word sense disambiguation programs. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 581–588, Granada, Spain.

A. Kilgarriff. 2000. English lexical sample task description.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

Rada F. Mihalcea. 2002. Word sense disambiguation with pattern learning and automatic feature selection. *Natural Language Engineering*, 8(4):343–358.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 40–47, San Francisco. Morgan Kaufmann Publishers.

T. Pedersen. 2001. Machine learning with lexical features: The duluth approach to senseval.

Bernhard Schölkopf and Alexander J. Smola. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press.

Using a Trie-based Structure for Question Analysis

Luiz Augusto Sangoi Pizzato
Centre for Language Technology
Macquarie University
2109 Sydney, Australia
pizzato@ics.mq.edu.au
http://www.clt.mq.edu.au

Abstract

This paper presents an approach for question analysis that defines the question subject and its required answer type by building a trie-based structure from a set of question patterns. The question analysis consists of comparing the question tokens with the path of nodes in the trie. A look-ahead process solve the mismatches of unknown words by assigning a entity-type or semantically linking them with other question words. The developed approach is evaluated using different datasets showing that its performance is comparable with state-of-the-art systems.

1 Introduction

When a question is presented to a person, or even to an automatic system, the first task, in order to provide an answer, is to understand the question. The question analysis process may not be very clear for people when answering questions, however for an automatic question answering (QA) system it plays a crucial role.

Acquiring the information embedded in a question is the primary task that allows the system to execute the right commands in order to provide the correct answer to it. According to Moldovan et al. (2003), when the question analysis fails, it is hard or almost impossible for a QA system to perform its task. The importance of the question analysis is very clear in the system of Moldovan et al. (2003) since this task is performed by 5 of the 10 modules that compose their system.

The most common approach for analysing questions is to divide the task into two parts: Finding the question expected answer type, and finding the question focus.

Many systems (Mollá-Aliod, 2003; Chen et al., 2001; Hovy et al., 2000) use a set of handcrafted rules for finding the expected answer type (EAT). Normally the rules are written as regular expressions (RE), while the task of finding the EAT consists of matching questions and REs. Every RE will have an associated EAT that will be assigned to a question if it matches its pattern.

For the task of finding the question focus, the simplest approach is to discard every stopword on the question and to consider the remaining terms as the focus representation.

In the approach described in this paper, the EAT and the question focus are defined using a trie-based structure built from a manually annotated corpus of questions. The structure stores the answer type in every trie node and uses the question words or entity types to link the nodes.

The question analysis method was evaluated over an annotated set of question of an academic domain, over the annotated TREC-2003 questions and over the 6,000 questions of the training/testing set of question of Li and Roth (2002) showing promising results.

This paper addresses a technique used to analyse natural language (NL) questions and its evaluation. Section 2 describes the technique, while Section 3 presents its evaluation. In Section 4 some related work is described. Finally, in Section 5 we present the concluding remarks and some further work.

2 Question Analysis

The developed technique for finding the EAT and the focus of the questions is based on a training set of questions. The questions in the training corpus are marked with their EAT and with their entities and entity types.

A training question is delimited by the tag `Q`. The `Q` tag must contain the attribute `AT` telling the EAT of a question. The question may contain entities, and these entities can be marked to help the learning process. For the purposes of presentation, the entity annotation is done in a way similar to the named entity task of past Message Understanding conferences (Grishman

and Sundheim, 1996) by using the `ENAMEX` tag and its `type` attribute.

(1) `<Q AT='NAME'> Who is the <ENAMEX type="POS">dean</ENAMEX> of <ENAMEX type="ORG">Macquarie University</ENAMEX>?</Q>`

Observe that Example 1 informs that 'dean' is a `POS` (Position) and 'Macquarie University' is an `ORG` (Organization).

Every question in the training file provides one question pattern. For instance, Example 1 informs that a question matching the RE in Example 2 asks for a name.

(2) `Who is the (.+) of (.+)?`

Notice that the RE of Example 2 has two groups of variable terms. If a question matches the RE, it is possible to assume that the words inside the groups match the same entity category as the one defined in the question RE. According to Example 1, the Example 2 categories are `POS` and `ORG`.

In our technique we use the words matching the non-fixed part of the RE as the question focus, while we define the EAT using the answer type of the RE.

2.1 Trie-based Structure

A trie $T(S)$, according to Clément et al. (1998), is a data structure defined by a recursive rule $T(S) = \langle T(S/a_1), T(S/a_2), \ldots, T(S/a_r)\rangle$, where S is an set of strings of the alphabet $A = \{a_j\}_{j=1}^{r}$, and S/a_n is all string of S that starts with a_n stripped their initial letter.

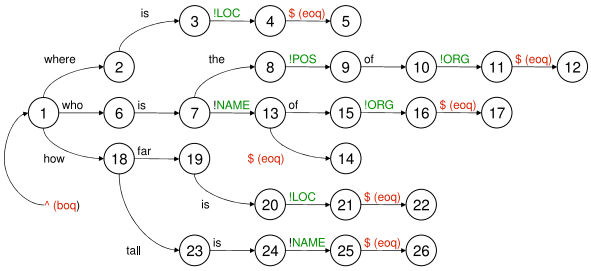
In our question analysis we used a trie-based structure where our 'strings' are the question patterns and our 'alphabet' is the set of question words and entity types.

A question pattern is a representation of the RE where the beginning and the end of question is marked and its non-fixed parts are represented by the entity type. For instance Example 1, would be transformed to:

(3) ^Who is the `!POS` of `!ORG` $

The construction of our question trie is similar to the construction of a dictionary trie. However the information stored, the tokens used, and the structure utilisation are different.

In the trie construction phase, every time a node is visited or created, the information regarding the frequency of its EAT is recorded. Since a node in the trie can be reached from different patterns, it is likely that we have a set



Nodes	Information (EAT,Frequency)
1	(LOC,1),(NAME,1),(DESC,2),(NUMBER,2)
2-5	(LOC,1)
6-7	(NAME,1),(DESC,2)
8-12	(NAME,1)
13	(DESC,2)
14-17	(DESC,1)
18	(NUMBER,2)
19-26	(NUMBER,1)

Figure 1: Trie for the question patterns of Table 1

of frequencies and categories recorded on every node.

Figure 1 shows how the information is structured and recorded in our question trie in case of training the patterns of Table 1. It can be observed that every node in the trie records one or more EAT.

Table 1: Training question/patterns of Figure 1

Question	Pattern	EAT
Where is Chile?	^Where is !LOC$	LOC
Who is the dean of ICS?	^Who is the !POS of !ORG$	NAME
Who is J. Smith?	^Who is !NAME$	DESC
Who is J. Smith of ICS?	^Who is !NAME of !ORG$	DESC
How far is Athens?	^How far is !LOC$	NO
How tall is Sting?	^How tall is !NAME$	NO

2.2 Trie-based Analysis

There are many differences as well as similarities between the utilisation of our trie structure for question analysis and the extraction of indexes from word tries. The first step in the question analysis is to transform the question into the pattern-like format of Example 3. The pattern-like format requires the beginning-of-question and end-of-question marks and if known (by the use of a Gazetteer file) the substitution of some of the question phrases by their entity type.

Using the question' patterns we try to match the first token of the question with the nodes of the trie. If a match is found, then the next token is searched on the nodes linked with the first one. This process continues until there is no more tokens to be examined or the current

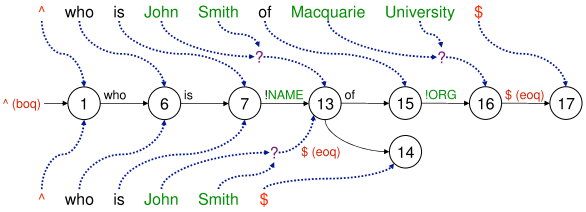


Figure 2: Look-ahead process in the analysis of questions

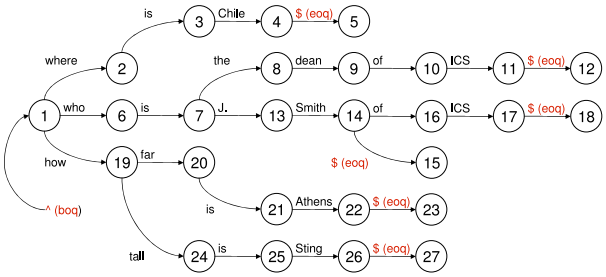


Figure 3: Trie-based structure built without entity information

token can not be matched against the following trie nodes.

This process returns the EAT with the highest frequency of the last visited node. This information will be used as the EAT of the question that was been analysed.

If the current token does not match any following nodes, then a look ahead becomes necessary. In this case the next token is examined over the next nodes of the following nodes. Figure 2 exemplifies the look-ahead process on the analysis of the questions 'Who is John Smith?' and 'Who is John Smith of Macquarie University?' over the trie of Figure 1

The analysis of question 'Who is John Smith?' is done by matching the beginning-of-sentence token and the words 'who' and 'is'. Notice that the words 'John' and 'Smith' and the phrase 'John Smith' were not replaced by their entity type since their condition as names is unknown by the Gazetteer. The word 'John' is not found in the nodes following 'is' (node 13), so the next question word ('Smith') is then searched in those nodes (14 and 15) which are 2 nodes away from the last matched one (node 7). The process continues to search for words in the question in a 2 nodes distance from the last word/node found.

If a match is found, all the words that were not found in previous interaction, are assumed to be of the same type as the node in between the matches. If more than one match is found, the path with the highest frequency will prevail. In this process, the node between the matching words/nodes will define the entity-type of the non-matching phrase on the question pattern.

In the examples of Figure 2, both questions complete the analysis and are assigned a description (DESC) as their EAT. If the process consumes all the tokens of the question and still does not find a match in the nodes, then the last visited node will define the question EAT.

The focus is defined by the entity part of the pattern-like representation of a question. The replacement of some of the question phrases by their entity types can be done before (using the Gazetteer file) or during the utilisation of the trie in the look-ahead process. In both occasions the phrases and their entity types define the question focus. For the questions of Figure 2 the focus would be the 'NAME' 'John Smith' and the 'ORG' 'Macquarie University'.

Our method also considers incomplete matches of question in the trie. If such cases occur, the EAT with the highest frequency of the last visited node will be assigned to the question. For instance, the most frequent EAT of node 6 will be assigned to the question 'Who?' since it is too short to completely traverse the trie. In a similar situation, the question 'Who killed JFK?' cannot be fully matched in the trie and the information of node 6 will define its EAT. Observe that in both cases the last analysed node defines the EAT.

As previous stated, our method requires a training corpus of questions annotated with their EAT and, if possible, with their entities and entity types. The method for finding the EAT does not require the markup of entities. In this case the trie is built only with the information from the words of the questions. Figure 3 shows the question trie constructed from the questions of Table 1 discarding the entity information.

When the entities and entity types are not marked, the analysis of question will still perform the same look-ahead process as demonstrated before. However, in this case, the look-ahead process does not define an entity category but describes an unknown relation between a word in the training questions and another word

or phrase in the question that is been analysed.

To illustrate this situation, consider the question 'Who is the administrative assistant of Macquarie University?'. Since neither 'administrative' nor 'assistant' can be found in the tier of Figure 3, the look-ahead process matches the word 'of' with node 10, assuming that there is a relation between 'administrative assistant' with 'dean'. The same situation will occur with 'Macquarie University' and 'ICS'.

In the current development of our technique, the information about the semantic relations of these words are simply discarded. Further studies are needed to understand where this semantic relations can be used in our QA method.

When the recognition of the entities and their entity types is not possible, the focus is defined by the remaining words in a stopword removing procedure. In some cases this approach finds the same focus words as our entity recognition, however it lacks the information of their entity type.

3 Evaluation of the Question Analyser

Our question analysis technique was intrinsically evaluated using a semi-automatically constructed training set of questions. We did not perform any extrinsic evaluation in the sense of Jones and Galliers (1996). That is to say, we did not perform any evaluation of the question analyser over the results in an embedded application such as the question answering task.

The training set contains 1385 randomly selected questions from a set of approximately 40,000 NL questions. The questions were extracted from the JustAsk search engine logs between February 2000 and April 2004. JustAsk is an information retrieval interface to the Macquarie University web site that encourages its users to present queries as full NL questions.

The questions posed in JustAsk are clearly domain dependent, since the search engine is limited to the university domain. Further studies are needed to evaluate how feasible this training set is in questions of different domains.

For the evaluation, we wanted to determine the impact of the size of the training set. For this, we randomly created a training set of x questions and we used the remaining questions for evaluation. To iron out potential idiosyncrasies of the training test we repeated the evaluation n times (normally $n = 200$ but for practical reasons sometimes we used different values)

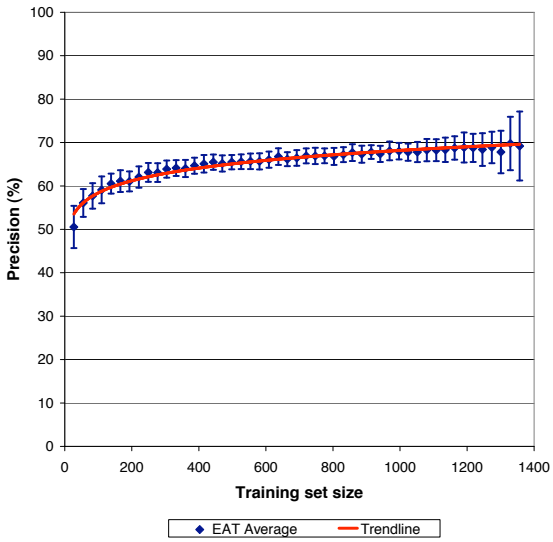


Figure 4: Average results for the EAT

and computed the average of the results, which are shown in Figures 4 and 5

Figure 4 shows a graphical representation of the evaluation of the question analysis over a set of 1385 annotated JustAsk NL questions. It also shows that the EAT precision improves according to the size of the training set. As the size of the training and the verification sets are directly related, it is possible to observe higher standard deviation in the results when few questions are used either for training or for verifying.

We observed that in Figure 4 the precision seems to have a limit in between 70 and 75 percent. In order to measure the hypothetic limit of these measures, we executed a test using the same set of questions for training and for validating the technique. The test showed that the maximum performance when the system was trained and validated with the full set of questions was around 85%.

The test also showed that the maximum performance for finding the EAT degrades when more training questions are provided. This happens because when new questions patterns are introduced, some of them may be similar and present ambiguous information to the overall system. In many cases questions with similar structures require different types of answers. Observe Examples 4 and 5:

(4)
```
<Q AT='NAME'> Who is the
<ENAMEX type='POS'>chair</ENAMEX>
of
<ENAMEX type='EVENT'>ALTW</ENAMEX>
</Q>
```

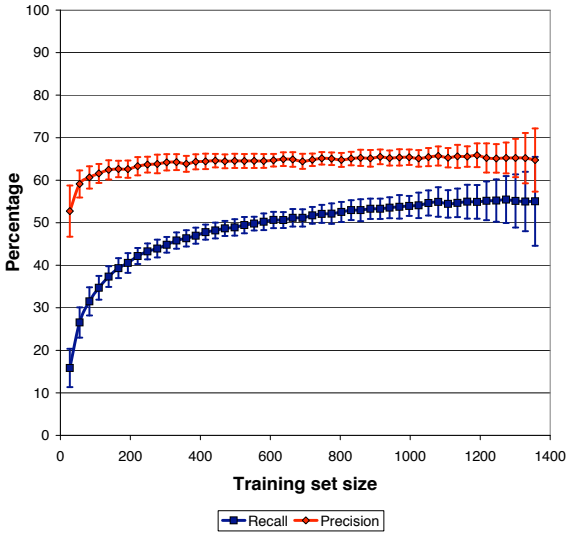


Figure 5: Average results for the question focus

(5) `<Q AT='ORG'> Who is the <ENAMEX type='POS'>sponsor</ENAMEX> of <ENAMEX type='EVENT'>ACL</ENAMEX> </Q>`

Both examples follow the same pattern (`^Who is the !POS of !EVENT$`), however Example 4 asks for a name of a person while Example 5 requires a name of an organization.

Figure 5 shows the evaluation of the question focus using precision and recall measures. Recall represents the percentage of entities in the verification set that were identified as focus by the question analysis, while the precision measure represents the percentage of entities found that actually existed in the original question.

The evaluation of Figure 5 shows that the performance of the focus identification improves for every new data inserted in the training set. The average of the recall measure increases from less than 20% to more than 50% with less than 600 questions. The results also show that after a few training questions the precision of the discovered entities is kept around 60 and 70 percent for all the training section.

The precision score in Figure 5 gives the impression to have a 65% limit while recall appears to have a limit in the region of 55% and 60%. An estimation of the maximum performance for the entities recognition revealed that the precision value could be as high as 80%, while recall value reaches 85% when all questions used for training are used for validating the technique.

The technique used to assign EATs to questions does not require the markup of entities in the training questions. And because of that, we were able to evaluate the technique on the set of TREC 2003 questions that were manually marked with their EAT information.

The results of this evaluation demonstrated that the precision increases as the size of the training set increases, reaching the mark of 70% with less than 150 training questions and approaching 80% on 400 questions.

To understand if the higher precision of the system in TREC 2003 question was achieved due to the lack of entity information, we tested the EAT precision of the system using the JustAsk training questions with and without the annotation of entities. The idea was to comprehend if the presence of the entities improve or worsen the quality of the EAT analysis. We observed that there were no significant differences between the results, therefore the inclusion or not of entities marks in the training set have to be defined exclusively by the goal of the analysis.

It is clear that the inclusion of entities markup will provide important information about the semantic role of the words in the query focus. However, the cost of marking entities in the question set may not be viable when the question analysis is only used for finding the EAT.

4 Related Work

The importance of a good question analysis for QA is clear. The correct EAT identification helps QA process to pinpoint answers by allowing it to focus on a certain answer category. The right question focus provides QA systems with knowledge that helps systems to choose the best sentences to support answers. In this section we discuss some of the techniques used for the task of question analysis.

According to Chen et al. (2001) the EAT recognition falls into two broad groups, those based on lexical categories and those based on answer patterns. The EAT analysis based on lexical categories can be identified by the lexical information present in the questions, while the analysis based on answer patterns are predicted by the recognition of certain question types.

It seems that the most popular approaches for the EAT identification are based on answer patterns. Most works in this group performs the analysis of questions using handcrafted rules

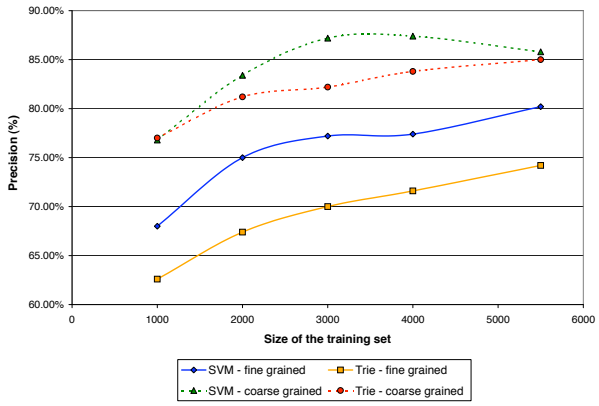


Figure 6: Average results for the trie-based and the SVM approaches

(Mollá-Aliod, 2003; Chen et al., 2001; Hovy et al., 2000).

Hovy et al. (2000) built a QA typology in order to create specific to general EAT. Question patterns were assigned for every answer type, and for those some examples of questions were provided. In a further work Hermjakob (2001) described their intentions of migrating from manual defined rules to automatic ones.

Our system, as described in this paper, uses a rule based approach to automatically build a trie-based question structure. This type of approach has the advantage of being capable of changing domains or even languages by using a different set of training questions.

In order to understand how well our technique performs in comparison to others, we tested our system using the same training/test set of questions used by the LAMP QA system (Zhang and Lee, 2003b).

The LAMP QA system uses a Support Vector Machine (SVM) to classify questions into answer categories. In further work Zhang and Lee (2003a) evaluated their technique using the testing dataset of Li and Roth (2002). Figure 6 compares the results of our trie-based approach with the one using SVM.

The comparison with Zhang and Lee (2003a) technique was made using the same testing dataset and considering the results of Zhang and Lee using bag-of-words features. This comparison shows that SVM provide better results for fine grained answer categories, while for coarse grained answer categories both techniques provide similar results when using the training sets of 1000 questions and 5500 questions.

The comparison shows that our technique provides reasonable result without the need of linguistic resources. And once again we notice that the accuracy of our technique improves when more training data is provided.

With a different approach some systems identify their EAT by using some lexical information of the questions. For instance, the work of Paşca and Harabagiu (2001) uses WordNet (Fellbaum, 1998) to assign a category for its answers. Their system matches questions' keywords with WordNet synsets, and by finding dependencies between synsets, derives an EAT from it.

Paşca and Harabagiu (2001) affirm that their approach for identifying the EAT was successful in 90% of the TREC-9 questions. Their approach for the EAT recognition used the Princeton WordNet along with an answer type taxonomy and a name entity recogniser. Their experiments showed that the use of a large semantic database can help to achieve high quality precision over ambiguous questions stems for finding the questions' EAT.

WordNet has been successfully used in almost every kind of natural language application; undoubtedly it can provide important information to question analysers. For instance, in the QA system of Na et al. (2002) WordNet supports some manually defined questions patterns in the classification of answer categories.

The evaluation of our question analyser shows that we can achieve good results regarding solely in pattern information. We believe that the performance of our system can be boosted by using a hybrid approach, where question patterns are combined with lexical and semantic information.

5 Concluding Remarks

This paper presented a method for question analysis that uses a trie-based structure in order to obtain the focus and the expected answer category of a question. The trie-based question analyser was evaluated by using different sets of annotated questions, demonstrating that the developed technique can be used as an alternative to handcrafted RE, since it is a simple method which provides reasonable quality results.

We observed that by increasing the size of the training set our method gets better results. In spite of the fact that the method shows an upper limit in performance, for either recognition of the EAT and the question focus, the results are

not far from the hypothetic maximum value.

It is observed that the hypothetic maximum performance decreases when the training set increases in size. This, as already stated, is due to implicit characteristics of question patterns; however this decrease in quality may be accentuated when poor or no guidelines are presented on the stage of building the training corpus.

Sometimes the job of defining the questions' EAT and their entities is hard even for human annotators. Some questions may have different interpretation on different occasions making the question analysis a challenging task. It is essential that the same decisions are made by the human annotator when dealing with ambiguous questions. Since this problem was only identified during the annotation of JustAsk training questions, our training set may contain some noisy markups. Some further work is needed to determine how this noise degrades the results of the question analysis.

Manual question markup requires not only well defined guidelines but also a great amount of time. The complexity of manually building a training corpus increases when the annotation of named-entities is required. In future work we intend to use the training questions without the markup of the named-entities. We are planning on using the parts of speech (POS) of the questions words and some semantic information from WordNet to assign the question focus and to find out its semantic role.

The extraction of the question focus has not been totally explored yet. For the question analysis on the Macquarie domain, the results for extracting the focus are promising. However, we believe that the combination of POS and semantic information may increase the precision and recall for either focus and the EAT.

To further ensure the effectiveness of the question analyser, we still need to perform an extrinsic analysis in a working question answering environment. Still, the results shown in this paper provide enough evidence that the our question analysis is feasible to be applied in a QA system.

References

J. Chen, A.R. Diekema, M.D. Taffet, N. McCracken, N. Ercan Ozgencil, O. Yilmazel, and E.D. Liddy. 2001. Question answering: CNLP at the TREC-10 question answering track. In *Proceedings of TREC-2001*, pages 485–494.

J. Clément, P. Flajolet, and B. Vallée. 1998. The analysis of hybrid trie structures. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 531–539, Philadelphia, PA. SIAM Press.

C. Fellbaum. 1998. *WordNet – An electronic lexical database*. MIT Press, Cambridge, Massachusetts.

R. Grishman and B. Sundheim. 1996. Message understanding conference-6: a brief history. In *Proceedings of the Coling'96*, pages 466–471.

U. Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the Workshop on Open-Domain Question Answering at ACL-2001*, Toulouse, France, July.

E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C. Y. Lin. 2000. Question answering in webclopedia. In *Proceedings of TREC-9*, pages 655–654.

K. Sparck Jones and J. R. Galliers. 1996. *Evaluating Natural Language Processing Systems: An Analysis and Review*. Springer-Verlag New York, Inc.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of the COLING-02*, pages 556–562.

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

D. Mollá-Aliod. 2003. Answerfinder in TREC 2003. In *Proceedings of TREC-2003*.

S.H. Na, I.S. Kang, S.Y. Lee, and J.H. Lee. 2002. Using grammatical relations, answer frequencies and the world wide web for TREC question answering. In *Proceedings of TREC-2002*.

M. Paşca and S. Harabagiu. 2001. High performance question/answering. In *Proceedings of SIGIR'01*, New Orleans, Louisiana, USA. ACM.

D. Zhang and W.S. Lee. 2003a. Question classification using support vector machines. In *Proceedings of the SIGIR-03*, pages 26–32. ACM Press.

D. Zhang and W.S. Lee. 2003b. A web-based question answering system. In *Proceedings of the SMA Annual Symposium 2003*, Singapore.

Evaluation of a Query-biased Document Summarisation Approach for the Question Ansnwering Task

Mingfang Wu[1] **Ross Wilkinson**[1] **Cécile Paris**[2]
CSIRO ICT Center,
[1]Gate 7, 71 Normanby Road, Private Bag 10, South Clayton 3169
[2]Locked Bag 17, North Ryde, NSW 1670, Australia
{Mingfang.Wu, Ross.Wilkinson, Cecile.Paris}@csiro.au

Abstract

This paper presents an approach on a query-biased document summarisation and an evaluation of the use of such an approach for question answering tasks. We observed a significant difference in the user's performance by presenting a list of documents customized to the task type, compared with a generic document summarization approach. This indicates that paying attention to the task and the searcher interaction may provide substantial improvement in task performance.

1 Introduction

People are searching information to meet their information needs from their tasks at hand. However, most search engines interact with user in a "one size fits all" fashion and ignore the user's preferences, search context or the task context. The burden is then placed on the user to scan, navigate, and read the retrieved documents to identify what s/he wants. We believe that paying attention to the nature of the information task and the needs of the searcher may provide benefits beyond those available through more accurate matching. As Saracevic [9] pointed out, the key to the future of information systems and searching processes lies not only in increased sophistication of technology, but also in increased understanding of human involvement with information.

In the study presented in this paper, we examine searchers' ability to carry out a question answering task [13]. Unlike the task of the non-interactive TREC question answer track [10], where the question answering focuses on fact-based, short answer questions such as "Who is the first prime minister of Australia". We looked at the type of question answering task that is more complex than the task of finding a single fact. The answer to this type of questions would not generally be available in a single document, but would require facts to be extracted from several documents. For example, an Australian cattle farmer would like an information access system that could tell s/he "which countries are the top ten importers of Australian beef?". An ideal answer should consist of a list of country names together with corresponding beef import data. This answer could be synthesized from scattered information collected from various sources, such as a news article about Japanese meat imports and an analysis report on Australian beef in the European market.

The successful completion of such a task requires an answer to be obtained, citing the relevant source documents. If we assume that we do not have an advanced language engine that can understand such questions and then synthesize answers to them, a searcher will be involved in the process, beyond simply initiating a query and reading a list of answers. Some of the elements that might lead to successful answering might include:

- support for query formulation (and re-formulation)
- effective matching and ordering of candidate documents
- delivery of a useful form of the list of the candidate documents
- support for extraction of answers from documents
- synthesis of the answer

There has been quite a bit of study on how to support query formulation, and the bulk of IR research has been devoted to better matching. Research into question answering technology (for automatic approaches) or text editing (for manual approaches) is needed for the last two activities. In this work, we have concentrated on the task of delivering a useful form of the list of the candidate documents. The research question we investigated is: given a same list of retrieved documents, will

the variation in document summary/surrogate improve searcher's performance on question answering task?

Under the evaluation framework of the TREC (Text REtreieval Conference) interactive track [7], we conducted two experiments that compared two types of candidate lists in two experimental systems. One system (the control system) uses the document title and the first N words of a document as the document's summary, while the other system (the testing system) uses the document title and the best three "answer chunks" extracted from the documents as the document's summary. The second confirming experiment repeated the first experiment, but with different search engine, test collection and subjects. The purpose of the second experiment is to confirm the strong results from the first experiment and to test whether the methodology could be generalized to web data.

The rest of the paper is organized as follows: Section 2 discusses our motivation and approach. Section 3 describes the experimental setup, including experiment design and test collections. Section 4 presents the experiments' results and provides detailed analysis. Section 5 provides the conclusions we have drawn.

2 Motivation and approach

In our previous studies [12], we investigated the use of clustering and classification methods to organize the retrieved documents, we found that while subjects could use the structured delivery format to locate groups of relevant documents, the subjects often either failed to identify a relevant document from the document summary or were unable to locate the answer component present within a relevant document.

We hypothesize that one of the reasons for potential gains from structured delivery not being realized is that in our previous test systems the tools that were provided to differentiate the answer containing documents from non-answer containing documents were inadequate for the task of question answering.

In our previous testing systems, a retrieved document is represented by its title. While a document's title may tell what the document is about, very often an answer component exists within a small chunk of the document, and this small chunk may not be related to the main theme of the document. For example, for the question "Which was the last dynasty of China: Qing or Ming?", the titles of the first two documents presented to a searcher are: "Claim Record Sale For Porcelain Ming Vase" and "Chinese Dish for Calligraphy Brushes Brings Record Price". The themes of the two documents are Ming vases and Chinese dishes respectively, but there are sentences in each document that mention the time span of the Ming Dynasty and of the Qing Dynasty. By reading only the titles, searchers miss a chance to easily and quickly determine the answer, even the answer components are in the top ranked documents.

In this work, we still use document retrieval, but focus on the surrogate or summary of the retrieved documents. Some experiments have evaluated the suitability of taking extracted paragraphs or sentences as a document summary [2], [6], [8]. The produced summary by these methods is purely based on individual document and basically a condensed version of a document - it requires the user less reading time to get to know the gist of the document. There are little studies that have shown whether the use of these summaries is suitable for the interactive question answering task.

In our approach, a document is summarized and represented by its title and the three best answer-indicative sentences (AIS). The three best AIS are dynamically generated after each query search, based on the following criteria:

- An AIS should contain at least one query word.
- The AIS are first ranked according to the number of *unique* query words contained in each AIS. If two AIS have the same number of unique query words, they will be ranked according to their order of appearance in the document.
- Only the top three AIS are selected.

Our hypothesis is that the title and answer-indicative sentences should provide a better indication of whether a document might help answer a given question. This is because documents can easily be completely off the topic of interest to the searcher, but still be relevant because they contain a part of the answer to the question. Therefore, our experiment focused on the comparison and evaluation of two systems using different summaries. The control system First20 uses the title and the first twenty words as the document summary, and the test system AIS3 uses the title and best three answer-indicative sentences as the document summary. Performance will be evaluated in terms of searchers' abilities to locate answer components, searchers' subjective perceptions of the systems, and the efforts required by searchers to determine answers.

3 Experimental setup

Experimental design

The experimental design concerns three major factors: system, question, and searcher, with focus

on the comparison of two experimental systems. Thus, we adopted a factorial, Latin-square experiment design. In this design, each searcher uses each system to search a block of questions; questions are rotated completely within two blocks. For an experiment involving two systems and eight questions, a block of sixteen searchers is needed.

System description

In each experiment, the two experimental systems use the same underlying search engine. The Experiment I used the MG [11] search engine, while the Experiment II used the Padre search engine [4]. In each experiment, the two experimental systems provide natural language querying only. For each query, both systems present a searcher with the summary of the top 100 retrieved documents in five consecutive pages, with each page containing 20 documents. Each system has a main window for showing these summary pages. A document reading window will pop up when a document title is clicked. If a searcher finds an answer component from the document reading window, s/he can click the "Save Answer" button in this window and a save window will pop up for the searcher to record the newly found answer component and modify previously saved answer components.

The difference between the two systems is the form and content of the result presented in the main windows. The main window of the control system (First20) is shown in Figure 1. The main windows of the test systems (AIS3) are shown in Figure 2 and Figure 3.

The AIS3 windows in each experiment are slightly different. In Experiment I (Figure 2), each answer-indicative sentence is linked to the content of the document and the sentence in the document is highlighted and brought to the top of the window. In Experiment II (Figure 3), we remove these links to make the interface closer to the interface of First20 and the three AIS truly the summary. There is a save icon beside each AIS (in Figure 2) or each document title (in Figure3) in AIS3, this icon has the same function as the Save Answer button in the document reading window. If a searcher finds a fact from the following three answer-indicative sentences, s/he can save the fact directly from this (summary) page by clicking the icon.

Document collection

The document collection used by Experiment I contains all newswire articles. Experiment II used a partial collection from the main web track (W10G) [1]. This collection is a snapshot of the WWW; all documents in the collection are web pages. To concentrate on document summaries instead of browsing, we removed all links and images inside a web page - for the purpose of this experiment; each web page was treated as a stand-alone document.

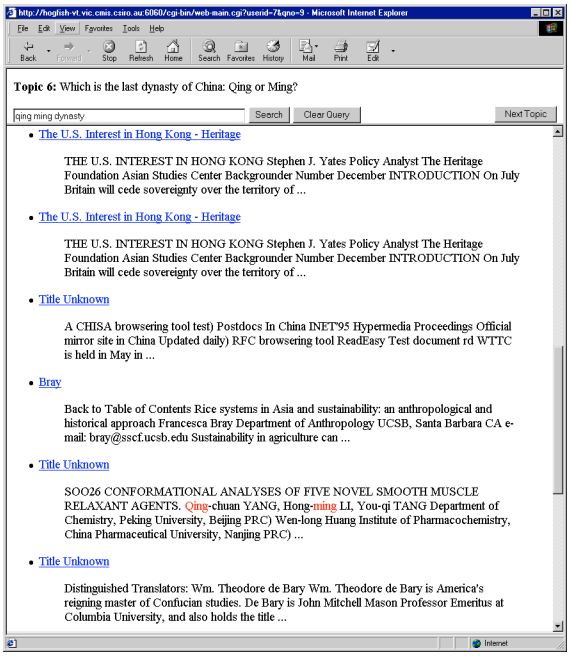


Figure 1. The interface of the First20

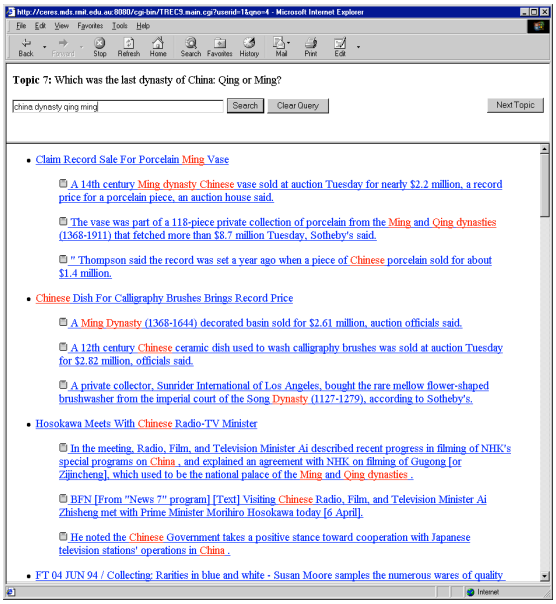


Figure 2. The interface of the AIS3 system in Experiment I

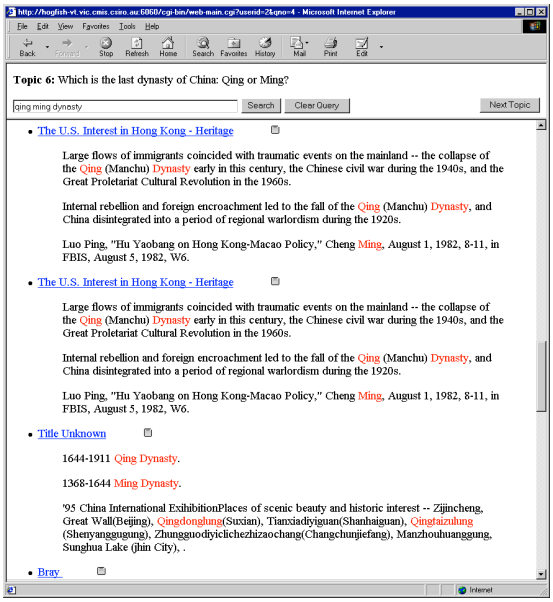


Figure 3. The interface of the AIS3 system in Experiment II

Questions

There are two types of questions in the experiments. The Type 1 questions are of the form <<find any n Xs>>; for example, "Name four films in which Orson Welles appeared.". The Type 2 questions are of the form <<compare two specific Xs>>; for example, "Which was the last dynasty of China: Qing or Ming?". For the Type 1 questions (question 1–4), a complete answer consists of n answer components, plus a list of supporting documents. For the Type 2 questions (question 5–8), two facts are usually needed to make the comparison, plus supporting documents.

Experiment I used a set of eight questions developed by TREC9i participants. To prepare a set of questions for Experiment II, we started with the eight questions from TREC9i. We then removed those questions that could not be fully answered from the document collection used in Experiment II. Additional questions were added either by modifying questions from the main web track, or were developed by an independent volunteer.

Evaluation

A searcher's performance is evaluated in terms of the success rate. For each search question, the saved answers and their supporting documents were collected for judging. There are two levels of judgement: one is whether the searcher finds the required number of answer components (for questions of Type 1) or whether the found facts are enough to infer the answer (for questions of Type 2); another is whether the submitted facts (or answers) are supported by the saved documents. For the success rate, a search session is given a score between 0 and 1: each correctly identified fact supported by a saved document contributes a score of 1/n to the search score, where n is the number of required answer components (or facts) for the question

Experimental procedure

Generally, we followed the procedure recommended by the TREC interactive track [7]. During the experiments, the subjects performed the following tasks:

- Pre-search preparation: consisting of introduction to the experiment, answering a pre-search questionnaire, demonstration of the main functions of each experimental system, and hands-on practice.
- Searching session: each subject attempts four questions on each of the systems, answering a pre-search questionnaire and a post-search questionnaire per question, and a post-system questionnaire per system. Subjects have a maximum of five minutes per question search.
- Answering an exit questionnaire.

Subjects

All searchers were recruited via an internal university newsgroup: all were students from the department of computer science. The average age of searchers was 23, with 4.7 years of online search experience.

Subjects were asked about their familiarity about each question. Overall, subjects claimed low familiarity with all questions (all under 3 on a 5-point Likert scale). In experiment I, the average familiarity of questions from each system is 1.5 (AIS) and 1.58 (First20). In experiment II, the scores are 2.1 (AIS) and 2.0 (First20). No significant correlations are found between familiarity and success

4 EXPERIMENTAL RESULTS

To determine the success of a system at supporting a user performing an information task, it is important to know how well the task is done, how much effort is required, and whether an information system is perceived as helpful. We use independent assessment for performance, system logging for effort, and questionnaires for perception.

4.1 Searcher performance

Experiment I

We aimed to determine whether searchers could answer questions more successfully with the

First20 system or the AIS3 system. Our results show that searchers using AIS3 had a higher success rate than those using First20 for all questions except for Question 5. Overall, by using AIS3, searchers' performance is improved by 38%.

In this experiment, the three variables to consider are the question, the searcher, and the system. Although the Latin-square design should minimize the effect of question and searcher, it is possible that question or searcher effects may still occur. An ANalysis Of Variance (ANOVA) model was used to test the significance of individual factor and the interactions between the factors. Here, the success rate is the dependent variable, and system, question, and searcher are three independent variables. A major advantage of using the ANOVA model is that the effect of each independent variable as well as their interactions are analyzed, whereas for the t-test, we can compare only one independent variable under different treatments. Table 1 shows the result of the three-way ANOVA test on success rates. It tells us that the system effect and question effect are significant, but that the searcher effect and the interaction effects are not.

Table 1. Experiment I: summary of ANOVA model for the success rate

Source	p-value
System	0.041
Question	0.000
Searcher	0.195
System * Question	0.414
Question * Searcher	0.691
System * Searcher	0.050

Experiment II

Experiment II was aimed to confirm the strong result from the experiment I. We planned to repeat the above experiment with a quite different document collection, another set of questions, and different searchers. However, we found that the technique for selecting AIS used in Experiment I could not be applied directly to web documents. Unlike news articles that have coherent text with a well-define discourse structure, web pages are often a chaotic jumble of phrases, links, graphics, and formatting commands. On the other hand, compared with news articles, web documents have more structural information. Although their mark-up is more for presentation effect than to indicate their logical structure, some information between two tags (for example: <LI>…</LI>) can be regarded as a semantically coherent unit and treated as a sentence. Therefore, in addition to the techniques used in Experiment I to segment documents into sentences, we also used some document mark-up as "sentence" indicators.

Table 2 shows the ANOVA test on the experiment II data. The table shows results similar to those in Table 1: only the system and the question have significant effect on the success rate. Overall, AIS3 leads to a performance improvement of 34% over First20.

Based on the searchers' performance in both experiments, our hypothesis that the AIS is a better form of document summary than the first N words for the question answering task is supported.

Table 2. Experiment II: summary of ANOVA model for the success rate

Source	p-value
System	0.020
Question	0.018
Searcher	0.547
System * Question	0.248
Question * Searcher	0.808
System * Searcher	0.525

4.2 Searcher effort

The effort of a searcher in determining answers to a question can be measured by the number of queries sent, the number of summary pages viewed, and the number of documents read.

On average, searchers sent fewer queries, viewed fewer summary pages, and read fewer documents from *AIS3* than from *First20* in both experiments (refer to Table 3).

We note that searchers generally did not use more than one summary page per query, nor did they need to read many documents to carry out the task. Considering the summary page of *AIS3* displays more text than that in *First20*, we may tentatively conclude that searchers read similar amount of text, but *AIS3* provides higher quality information than the *First20* does, since we know searcher performance is better.

Searcher preference

The perception of searchers of the systems is captured by three questions in exit questionnaire. The three questions are

- Q1: Which of the two systems did you find easier to learn to use?
- Q2: Which of the two systems did you find easier to use?

Table 3. Searchers' interactions with two systems

	Experiment I		Experiment II	
	First20 Mean(SD)	*AIS3* Mean(SD)	*First20* Mean(SD)	*AIS3* Mean(SD)
No. of unique queries sent	2.14(0.56)	1.73(0.57)	2.0(1.2)	1.7(1.0)
No. of surrogate pages viewed	2.80(1.64)	1.98(0.97)	2.4(1.4)	2.0(1.3)
No. of documents read	3.42(1.22)	2.66(0.77)	4.2(2.8)	3.2(2.7)

Table 4. Searchers' perceptions of two systems

	Experiment I			Experiment II		
	Q1	Q2	Q3	Q1	Q2	Q3
First20	3	4	5	2	2	2
AIS3	8	11	11	10	12	13
No difference	5	1		4	2	1

- Q3: Which of the two systems did you like the best overall?

The distribution of the searchers' choices is shown in Table 4. Combining the results from the two experiments' questionnaires, for question 1, 15% of subjects selected *First20,* while 56% of subjects selected *AIS3*; for question 2, 19% of subjects selected *First20,* while 71% of subjects selected *AIS3*; for question 3, 22% of subjects preferred *First20*, while 75% preferred *AIS3*.

5 Conclusion

In this paper, we report two user studies on interactive question answering task. By constructing a delivery interface that takes into account the nature of the task, we saw that searchers:

- issued fewer queries
- read fewer documents
- found more answers

We conducted two experiments that would allow us to determine searcher performance, searcher effort and searcher preference. Our results show that searchers' performance when using an AIS3 system is improved over using a First20 system, based on objective assessment; this result is consistent in both experiments. The performance difference between two experimental systems is statistically significant. The data suggests that searchers using AIS3 require less effort, although cognitive load experiments are required to confirm this. Finally, AIS3 is preferred by most searchers. Thus, the experiments support our hypothesis that AIS3 is a better indication of document suitability than First20, for the question answering task.

Different search tasks may require different delivery methods. For example: the clustering of retrieved documents can be used for the task of finding relevant documents [5], and the classification of retrieved documents can be used for the purposing of browsing. However, for the task of question answering, we found that none of these delivery methods performed better than a ranked list [12]. The experiments presented in this paper indicate that a relatively simple document summary can significantly improve the searcher's performance in question answering task.

References

[1] Bailey P., Craswell N. and Hawking D. *Engineering a Multi-purpose Test Collection for Web Retrieval Experiments.* Information Processing and Management. 2001.

[2] Berger A. L. and Mittal V. O. *OCELOT: A System for summarizing web pages.* In Proceedings of the 23rd ACM SIGIR Conference. July 24-28, 2000, Athens, Greece (pp. 144–151).

[3] D'Souza D., Fuller M., *et al. The Melbourne TREC-9 Experiments.* In Proceedings of the Ninth Text Retrieval Conference (TREC-9) (pp. 366-379). November 2000, Gaithersberg, MD, USA.

[4] Hawking D., Craswell N. and Thistlewaite P. *ACSys TREC-8 Experiments.* In Proceeding of Seventh Text Retrieval Conference (TREC-8), November 1999, Gaithersburg, MD, USA.

[5] Hearst M. A. and Pedersen J. O. *Reexamining the cluster hypothesis: Scatter/gather on retrieval results.* In Proceedings of the 19th ACM SIGIR conference, August 1996, Zurich, Switzerland (pp. 76-84).

[6] Kupiec J., Pedersen J. and Chen F. *A Trainable Document Summarizer.* In Proceedings of the ACM SIGIR conference, July 1995, New York, USA (pp. 68–73).

[7] Over P. *TREC-9 Interactive Track – Basics.* In Note papers of TREC-9. November 2000, Gaithersberg, MD, USA (pp 721-728).

[8] Salton G., Singhal A., Mitra M. and Buckley C. *Automatic Text Structure and Summarization.* Information Processing and Management. Vol. 33 (2) 193-207, 1997.

[9] Saracevic T. and Kentor P. *A Study of Information Seeking and Retrieving: I. Background and Methodology.* Journal of the American Society for Information Science. Vol. 39(3), 161-176. 1988.

[10] Voorhees E. M. *The TREC-8 Question Answering Track Report.* In proceedings of the Nipth Text Retrieval Conference (TREC-8). Novemeber 1999, Gaithersberg, MD, USA.

[11] Witten I., Moffat A. and Bell T. *Managing Gigabytes: Compressing and indexing documents and images.* Van Nostrand Reinhold. 1994.

[12] Wu M., Fuller M. and Wilkinson R. *Using clustering and classification approaches in interactive retrieval.* Information Processing & Management, 37 (2001) 459-484.

[13] Wu, M., Fuller, M. and Wilkinson, R. *Searcher performance in question answering.* In Proceedings of 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 375-381, 2001.

Thin Parsing: A Balance between Wide Scale Parsing and Chunking

Jon Patrick and Pham Hong Nguyen
Sydney Language Technology Research Group
School of Information Technologies
University of Sydney
{jonpat,pham}@it.usyd.edu.au

Abstract

This work presents a type of parser that takes the process of chunking to the stage of producing full parse trees. This type of parser, denoted Thin Parsers (TP) in this work has the characteristics of: following a given grammar, creating full parse trees, producing only a limited number of full parse trees, parsing in linear time of sentence length. Performance standards on the Penn Tree Bank show results slightly under that of stochastic parsers but faster performance. Various types of Thin Parsers are presented.

1 Introduction

Bottom-up and Top-down methods of parsing, invented in 1950s, can parse in time which is an exponential function of the length n of an input sentence $O(c^n)$. Tabular parsing methods such as the CYK (Cocke-Younger-Kasami) and Earley can parse context free languages in polynomial time $O(n^3)$, a significant improvement compared with the previous parsers. These methods are usually called "traditional methods". However these methods are relatively slow, as they produce the complete parse trees. Also they use exhaustive search to find all possible parse trees, subsequently leading to the problem of needing to select an optimal parse from a very large set of solutions. Hence, researchers have tried to find other parsing methods and/or other formalisms for natural languages. For example, some researchers have tried to convert natural languages into regular languages to parse them by finite state automata (Black 1989, Pereira 2003). Some others work with parsers based on LL or LR grammars/parsers. However, they still have been unable to gain anything but small improvements.

Recently, with the creation of large annotated corpora such as the Penn Tree Bank, probabilistic parsing methods based on their manifest grammars appear to be a useful solution for the parsing problem. In this situation, the induction of probabilistic context free grammars (PCFG) derived from the annotated corpora has become a new objective and many researchers have quickly developed improved performance (Ratnaparkhi 1994, Collins 1996).

A different use of tree bank corpora is made by the chunking or shallow parsing process (Abney 1991, Brants 1999, Tjong Kim Sang 2000).The main idea behind this parsing strategy is that both human and machine usually do not require full information of the parsing processes. These parsers usually produce only partial parse trees instead of full trees, which can be a disadvantage, but, the trade off is that the parse time becomes linear with the length of input sentences $O(n)$. These parsers are called shallow parsers and the inference of probabilistic grammar rule sets is one of their useful characteristics.

Unfortunately, shallow parsers cannot be used in applications in which full parse trees are required. Under these circumstances there is a need to use traditional parsers or improve shallow parsers to produce full parse trees. The challenge here is to solve the problems created by inferring a parser from a corpus. The strategy used in this research is to give attention to improving the parser without recourse to comprehensive generative probabilistic models of the tree bank but rather by developing fast

algorithms with accuracies rivaling that of probabilistic models.

Justification for giving dominant attention to speed performance in the context of an inductive inference strategy is well supported by some established heuristics and by certain utilitarian values. As well, a search of the literature has failed to reveal any substantial attempt to analyse, on the basis of computability and speed performance, the advantages of an inductive inference approach to constructing a language parser from tree banks.

Well recognized heuristics that have been established and support the motivation for this work are that language is predictable. It is well recognized that there is significant redundancy in language at the lexicogrammar and semantic levels and that corpus analysis has demonstrated there are reasonable predictors for subsequent phenomena from a given point in a sentence.

Furthermore, Memory Based Learning has established the principle that "forgetting is harmful" to performance (Daelemans, van den Bosch & Zavrel 1999) and so remembering is helpful.

Functional motivations include: large scale grammars can be better utilized by developing parser from their outputs; applications exist that don't need wide coverage grammars but rather high speed. Current research in speech recognition is moving into a phase of exploiting lexicogrammar to improve performance (Lemon, 2004).

This paper promotes and develops a specific class of parsing methods, Thin Parsers (TP). Like traditional parsers, this method considers natural languages as context free languages and/or probabilistic context free languages. The aims of Thin Parsers are to overcome some weak points of both traditional (full) and shallow parsers. In comparison to full parsers, the Thin Parsers should work faster in practice and produce one or a few parse trees instead of large number. The main advantage of TPs over shallow parsers is that they can build full parse trees instead of partial trees, however, thin parsers do not have the ability to infer new grammar rules as shallow parsers do. Thus Thin Parsers could be considered closer to full parsers than shallow parsers. The Thin parser can be considered as a balance between the issues of the language class, parser speed and accuracy, and the need for full parse information. Hence, in this research the focus is on building performance models which are potentially useful in real situations rather than to build competence models.

2 Pre-processing of the PTB

The TPs in this research are developed from the Penn Tree Bank (PTB). As with other corpora, the PTB has some errors. Some of them can be corrected automatically by program whereas the others require manual reviews and corrections.

As this research deals with CFGs, which lack the ability to smooth errors, better data for establishing a "gold standard" for both training and testing models is needed. From many authors working with the PTB, it appears that only Bod's (1995) research is based on cleaned data as is this study.

Each sentence of the PTB corpus is presented in three different formats: raw sentences, sentences with part-of-speech annotation, and skeletal syntactic bracketed sentences each in a separate file[1]. Correct matching of these triples across the three files is needed to build up the full parse trees. However, due to errors in the data, the numbers of sentences are quite different in each file, making the work of extracting triples much harder and requiring human intervention.

From the development of suitable software, around 22,000 triples or only half of the data were extracted automatically. After correcting the data manually, this has been increased to 34,000 triples. This data set is denoted as the "Large set". From these triples, a subset of 6,700 triples was constructed, denoted the "Clean set" by deleting any triples having null elements and/or having incorrect parse trees (the trees that have none or more than one *S* node at the top of tree). This set is used in this research. The new set is quite small, compared with the original data, but still enough for experimentation as it is much larger than the sets

[1] Raw sentences are not used in this work.

used in other research of a similar nature, for example Bod (1995) only used 500 sentences.

3 Finite State Automata (FSA)

The starting point for our modeling of the PTB is the inductive inference of FSAs for both the POS tag sequences and the phrasal tag sequences (see Fig. 1). Two tests using POS and Phrasal tags were run using 2 data sets (Large and Clean) to create a baseline for further work (Table 1). The results are quite similar with the Sekine's (1997) although they are slightly worse. The reasons appear to be: i) a smaller training set is used here (30,600 sentences compared with 47,219 sentences); ii) The training and testing here is on different sets.

The results for Phrasal tags are significantly better than POS tags (19.19 % compared with 4.22%). The main reasons are: i) the average length of sentences (14.71) is shorter (21.23), creating the potential for repeated structures to be more frequent; ii) the annotation set for POS (36 POS tags + 12 other tags) is much larger than in Phrasals (14 syntactic tags + 4 null elements) (Marcus 1993). These results are used as a base line for our research. However, they are still far from that of a good probabilistic parser (with over 70%).

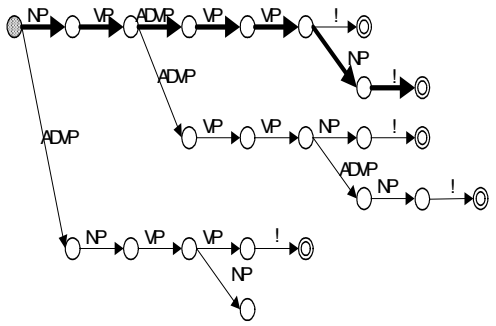


Figure 1. Inferred FSA for phrasal tag sequences in the PTB.

The Clean data set obviously brings some improvements. In the case of phrasal tags the FSA can parse successfully more than 44% of the phrasal test sentences. One of the reasons is that the average length of sentences is shorter. The POS tag results worsen between the Large and the Clean data sets compared to the phrasal tags. From these tests, it is evident that the lengths of sentences can affect the recognition results, the shorter the sentences, the better the result.

	POS	Phrasal
Large	4.22%	19.19%
Clean	3.66%	44.87%

Table 1. Coverage attained by inferred FSAs of PTB data sets.

In Charniak's (1996) work he ignored all sentences of length greater than 40 in the test data to avoid high parsing costs. An examination of the Big data set confirms his assessment that only a small sample are excluded. There are 29,462 sentences out of the 30,600 training set (96.28%) with a length of 40 words or under.

Our tests to identify relationships between the tag length and coverage show for POS tag sequences the FSA parser begins to show deterioration for lengths higher than 11 tags and for Phrasal tags about 14 tags.

3.1 Combination of POS and Phrasal FSAs

The POS tag FSA is useful in practice however, the main limitation is the low coverage. In contrast, the Phrasal tag FSA is not of much use even though it has a much higher coverage. To overcome the disadvantage of the first FSA and take the advantage of the second, they are merged into a new parser that works with inputs of POS and achieves a better coverage.

This new model can be viewed as two components, one is a simple FSA for recognizing phrasal tag sequences. The other part is a converter to convert sequences of part-of-speech tags into sequences of phrasal tags for input into the FSA. The Converter creates the mapping of the POS tags to phrasal tags. The Phrasal FSA after recognizing a string of phrasal tags identifies the tree top of a legal tree top in the training set to complete the full parse tree (see Fig 2).

3.1.1 Phrasal rules

The rule sets for converting POS tags into phrasal tags are extracted from the PTB. For example, from the tree in Figure 3, some phrasal rules as follows can be extracted:

NP→DT JJ NN NNS; VP→VBP; NP→NN; ADVP→RB; VP→VBG; QP→RB IN CD

From the 6700 sentences in the Clean dataset around 1200 phrasal rules were automatically extracted. This rule set is only a small subset of the PTB grammar, which has at least 10,000 rules as mentioned in Charniak (1996).

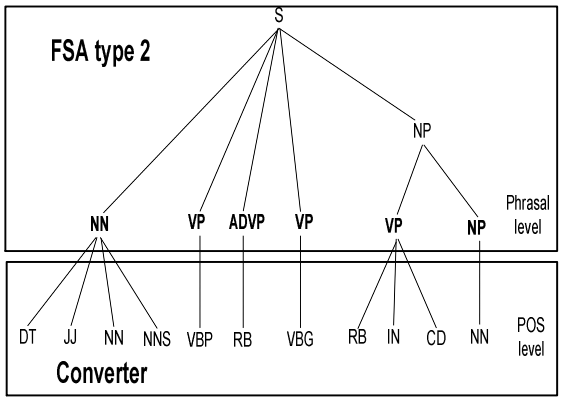


Figure 2. Combination of POS and Phrasal tag FSAs.

3.1.2 Limitations on a Converter

For parsing an unseen sentence a Converter is needed to perform an exhaustive search algorithm. That is it has to try every combination of POS tags to phrasal tag conversion for the set of POS tags in the sentence until the string of phrasal tags is recognized by the main FSA. However, this approach has exponential explosion even for a small rule set and short sentence.

3.1.3 Recursive Transition Network Induction – model TP1

In the first attempt to create an efficient Converter, the two kinds of FSA are combined by using Recursive Transition Networks (RTN) (Woods, 1970) because this model can help reduce the exponential time problem.

The two kinds of FSA are, one for part of speech tags and one for phrasal tags. The second one is the same FSA used in the initial tests above. The first one (type 1) uses a string of tags taken from the right side of phrasal rules instead of POS sentences. All rules which have the same left side are used to build one FSA. Thus there are many different FSA corresponding to a phrasal tag.

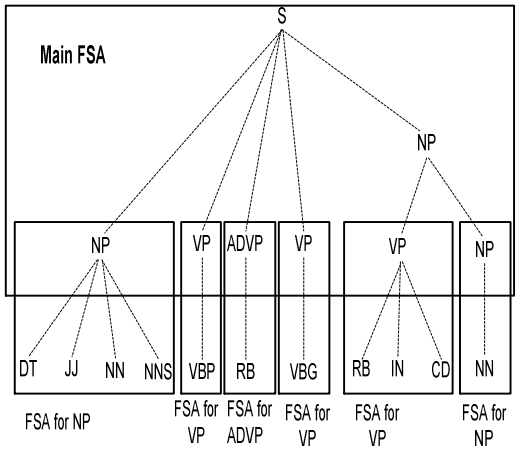


Figure 3. Combination of two types of FSA into an RTN.

3.2 Results and Analysis

The results of parsing by an RTN are shown in Table 2. The 6700 sentences of the Clean data set were used with 10 fold cross-validation. There is no restriction on sentence length for the first test, however, in the second test, like Charniak (1996), all the sentences with length in excess of 40 tags were removed. The RTN method has a very high coverage (98.32%) but the accuracy (R= 19%, P=23%, F=21%) are below the base line.

Theoretically, parsing by an RTN method is not as fast as using an FSA. Woods (1970) has shown that the time is about $O(c^n)$. However, in this work the RTN performs much faster, the average parse time per sentence is 0.0176 second[2] and slightly better 0.0156 if the sentences which are longer than 40 tags are ignored because:

- the phrasal FSA has only a depth of one
- there are no recursive loops
- the first level is recognized as a chunk, not only a tag, thus, the number of elements at the first level is reduced significantly

[2] All tests are done on a computer P4, 1.6 Ghz, 1 GB Ram. The time includes time for parsing, extracting parse trees and measuring accuracy.

- the number of phrasal rules is around 1400 – much smaller than the total number of grammar rules of the PTB[3]
- this model stops parsing after finding a solution.

In the worst case, the maximum parse time for all data is 1.603 seconds and for <41 data is only 0.911 seconds[4].

The results in Table 2 of the RTN show a very high proportion of coverage – 98%. The high coverage and fast parse time of this model could well be useful for some applications.

N	**C'age**	**R**	**P**	**F**	**Time**
all	98.32	19.74	23.92	21.83	0.0176
<41	**98.48**	**19.18**	**23.3**	**21.24**	**0.0156**

Table 2. Performance percentages of RTNs on the Clean data (10-fold cross-validation).

4 Shallow Parsers for building Parse Trees

To establish an understanding of the base performance of shallow parsers when used to build full parse trees we developed a method based on Abney (1991). The first experiment showed the coverage of the system is only 0.5% (only 3 of 600 test sentences).

With the aim of producing a better assessment of coverage, the accuracy of the base sub-trees (lower parts of trees) which are created by the shallow parser is computed by an alternative method, that is the tag *S* is added to every incomplete parse tree. This produced the results of P=36.65%, R=15.03%. These results can be considered as another baseline. Comparisons of these results to the works of other authors are not possible as they have only reported the accuracy of chunkers.

[3] An Earley parser (with parse time $O(n^3)$) was built and tested using a reduced set of 9000 rules of the PTB, in general this parser takes from 30 seconds / short sentence to some days for long ones (around 30 words). When using a much smaller and manually compacted rule set of 3000 the parser still takes several hours for a long sentence.

[4] The parse time is computed here to demonstrate how fast the model TP1 is in practice but not the fastest speed of this model.

5 Constructing Thin Parsers from Shallow Parsers – TP2

In this section, some improvements to the shallow parser model to extract the full parse trees are developed. These improvements can also be viewed as mixing models, that is a shallow parser and a finite state automaton (FSA), a shallow parser and a data-oriented parser (DOP), and the adaptation of a shallow parser with some non-deterministic components. The framework objective behind this blending is to reduce the number of levels needed in parse trees, reduce the number of elements of the parse trees, use of more stored pre-computed information, and to use non-deterministic mechanisms.

The large number of levels of the parse tree for the shallow parser is one of the major reasons for the poor coverage in extracting full parse trees. With each new level, the parser has more chance of being trapped and/or creating errors which will propagate to higher levels, that is the coverage and accuracy are affected seriously by the number of levels. Hence, the idea of improvement here is to let the parser work for the first level or at the most the few first levels only, which can work well. Then another process adds the top parts of trees to complete the full trees.

The tree tops that will be used to complete the parsing are pre-computed from the training corpus and saved into a database. One extra advantage of this method is that even though at the lower levels the trees are variable and large in number, at the higher levels the multiple forms decrease sharply. This means the number of tree tops is small, and easy to collect and to save to a database. Hence in testing the system has two parts. The first part works like a shallow parser and the second part is a search mechanism retrieving from this database.

After creating the base sub-trees of a parse tree (by working for only one or the first few levels), the shallow parser passes the result to the next process "Find and Join". This process retrieves from a database a saved tree top matching the tag configuration of these base sub-trees. If a suitable tree top is found, this function will attach it to the lower sub-trees to create a full parse tree. The condition to match the two parts

is straightforward: the top sequence of tags of the base sub-trees must be matched with the bottom sequence of the tree top.

Although four methods can be considered for assessment (Pham, 2004), the test results are shown here with a weaker measure, adjusted crossover brackets, to enable comparison with other authors. The data for training and testing is the Clean data set (6000 sentences for training and 600 for testing). After chunking *k* levels (column *Level/k*) the system gets the results sequence which is recognized by an FSA. Column *Succ* shows the numbers of parsed sentences.

Model TP2 can parse successfully a range from 21.67% to 55.83% of input sentences with accuracies (*F*) from 76.61% to 91.68% (Table 3). The accuracies are highest when the shallow parser works for only the first level (level 0). The coverage, unlike accuracy, increases by level and reaches the highest proportion at level 7 (55.83%) before decreasing slightly. As previously discussed, with the higher levels the variety of tree tops decreases sharply, helping the system match these tops more successfully. The model TP2 can parse successfully over 46% of total sentences with high accuracy of over 76% at k>=5.

Level/ k	Succ.	Coverage %	P %	R %	F %
1	239	39.8	92.3	91.1	91.7
2	130	21.7	90.4	89.7	90.0
3	219	36.5	82.3	83.1	82.7
4	248	41.3	79.9	80.5	80.2
5	277	46.2	78.5	79.3	78.9
6	304	50.7	78.8	79.4	79.1
7	335	55.8	77.3	78.1	77.7
8	307	51.2	76.5	76.7	76.6
9	316	52.7	76.6	77.1	76.8

Table 3. Results of model TP2 chunking, measures of the accuracy of nodes with adjusting crossover brackets.

6 Reduction of the number of tree elements - TP3

The number of grammar rules available for use in a parse tree is usually large. For each extra rule that is used in a parse the tree has more variation at higher levels. This provides more opportunities for the parse to be trapped or to propagate more errors. Furthermore, if two rules have the same right hand side, then the one with the highest frequency is used for building the tree. However this selection process of removing a less frequent rule is the main reason for shallow parsers not being able to build up a full parse.

Another idea for improvement of model TP2 is to reduce the number of grammar rules per parse tree by using the tree fragments which could be larger than these rules. By such a mechanism, the parser has more chance to use maximally frequent fragments, that is combinations of grammar rules, rather than grammar rules alone, thus avoiding the omission of less frequent grammar rules. Another advantage is that the number of levels can be reduced and so gain the benefit of getting higher coverage as well as faster execution time.

This idea is partly similar to Bod's (1995) model of DOP for using tree fragments, however in detail, there are differences in the ways tree fragments are collected and the method of parsing.

6.1 Collection of fragments

In the DOP model, Bod collects all possible tree fragments. However, this has the disadvantage that the number of fragments grows exponentially with size of grammar.

In the case of the TP2 model, tree fragments are collected based on the data set prepared for the shallow parsing experiments, that is the filled trees, that is all branches have nodes at each lower level. The conditions for collecting a sub-tree as a real fragment are that all leaves at the end of the tree start on the same level, as the tree is filled, and, it has a height of 2 levels or more.

For example, Figure 4 shows a sub-tree of a parse tree. The DOP model will collect a total of 32 fragments from this sub tree. In the TP3 model, firstly this sub tree is converted into a

filled tree as shown in Figure 5.a. From this data only two fragments are collected as in Figures 5.b and 5.c, hence an exponential increase of fragments is prevented.

6.2 Results

All data for training and testing model TP3 is reported for both shallow parsing (chunking, Table 4) and parsing (Table 5).

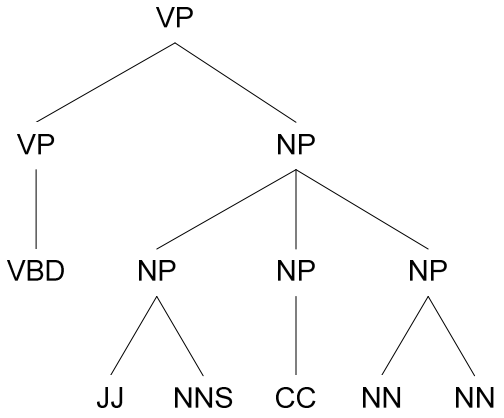


Figure 4. An example of a sub-tree from a parse tree.

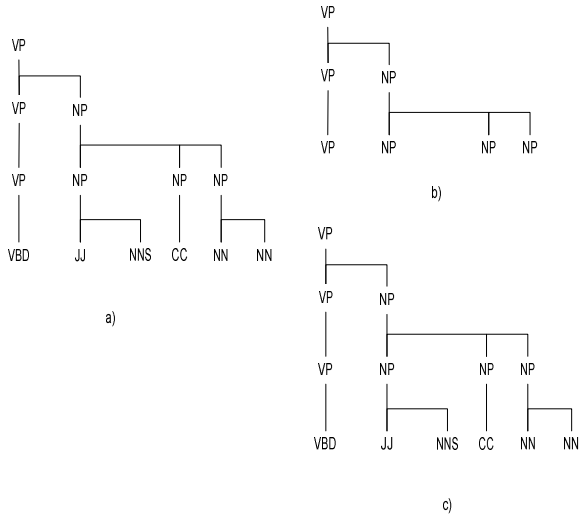


Figure 5. The parse sub-tree from Fig. 4 converted into filled sub-tree (a). From (a) in training only two fragments (b) and (c) are collected in model TP3.

Table 4 shows the results of chunking the Clean data set. This is quite similar to Table 5.3 text chunking except: the maximum level is only 1, and, the accuracy of the two highest levels appears to be exceptional because the number of sentences is quite small (844 and 586 respectively), compared with whole learning set (6000 sentence).

As the levels increase the number of sentences available for both training and testing decreases as there is a variable range of tree heights across the corpus.

Level	Precision	Recall	F
0	84.2	83.7	83.9
1	72.5	64.1	68.1
2	75.4	59.0	66.2
3	68.8	53.1	59.9
4	68.1	52.8	59.5
5	72.1	65.4	68.6
6	79.9	68.3	73.6
7	91.1	85.4	88.2
≥8	100.0	100.0	100.0

Table 4. Text chunking results using the Clean data set for model TP3.

Table 5 shows the results of the parse of 600 sentences from the Clean data set. The coverage of TP3 in the best case is over 61% (at level 5), better than the best coverage 55% of TP2. However, the accuracies in general are a little worse than the accuracies of TP2. Both TP2 and TP3 reach the highest accuracies at level 1 and reduce at higher levels.

Level	Succ.	Coverage	P	R	F
1	230	38.4	84.8	85.5	85.2
2	135	22.5	64.8	73.7	69.2
3	184	30.7	63.8	74.7	69.3
4	351	58.6	64.7	74.8	69.8
5	368	61.4	69.3	76.8	73.1
6	183	30.6	70.7	79.2	75.00
7	62	10.4	68.7	75.7	72.2
≥8	1	0.17	73.5	83.3	78.4

Table 5. Parsing results using the Clean data set with model TP3.

7 Non-Deterministic models –TP4, TP5

Two approaches have been investigated in an attempt to produce concurrent solutions to improving coverage and accuracy. Whilst the methods will be reported elsewhere the results are relevant to this study and show great promise. The first method, model TP4, uses the idea of a non-deterministic chunker with a deterministic attacher. The second model, TP5, uses a deterministic chunker with a non-deterministic attacher. By exploitation of the learning process of creating a tree tops database and using algorithms to match the database entries against the candidate parse trees rather than the other way the practical implementation requires linear time to parse. The performance of the two systems is presented in Table 6.

Model	Coverage	P	R	F
TP4	63.7%	83.0	84.6	83.8
TP5	82.2%	88.3	88.0	88.1

Table 6. Performance statistics of models TP4 & TP5.

8 Conclusions

Some improvements to building a thin parser from a shallow parser have been presented in the form of models TP2 and TP3, TP4 and TP5. They all include in the first stage a shallow parser. In general, these models can solve to some degree some problems of shallow parsing when extracting full parse trees. The problems are: deterministic mechanisms are not adequate enough for dealing with the ambiguities of natural language; the configuration of connections between layers make any error that occurs in the lower levels propagate to higher levels and worsen the performance; the Chunker and Attacher need to work with some limitations, such as using only the most probable rules; and not attach chunks with a size of 1. The results of these models are quite good: the coverage (from 0.5% of original shallow parser) has improved to more than 82%, and the accuracy has improved from 20% to over 88%.

9 References

Abney S. 1991. Parsing By Chunks. In: R. Berwick, S. Abney & C. Tenny (eds.), *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Black, A. (1989). Finite State Machines from Feature Grammars, *International Workshop on Parsing Technologies*, Carnegie-Mellon University, Pittsburgh, PA.

Bod R. 1995. Enriching Linguistics with Statistics: Performance Models of Natural Language, *Academische Pers*, The Netherlands. 143 pp. (Ph.D. thesis).

Brants T. 1999. Cascaded Markov Models. In *Proc of 9th Conf. of the Euro. Chapt of the ACL, EACL-99*, Norway.

Charniak E. 1996. Tree-bank grammars, *Tech Report CS-96-02*, Dept of Computer Science, Brown University.

Collins M. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. Proc of the *34th Annual Meeting of the ACL*, Santa Cruz.

Daelemans, W, van den Bosch, A & Zavrel, J. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*

Lemon, O. 2004. Context sensitive speech recognition in ISU dialogue systems: Results for the grammar switching approach. *Proc. 8th Workshop on the Semantics and Pragmatics of Dialogue, CATALOG'04*.

Marcus M., Santorini, B. Marcinkiewicz, M. 1993. Building a large annotated corpus of English: the Penn Treebank, *Jnl of Computational Linguistics*, Vol. 19 No.1.

Pham Hong Nguyen 2004. Thin Parsing, PhD Thesis, Univeristy of Sydney.

Tjong Kim Sang E. F. and Buchholz, S. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc of CoNLL-2000*, Lisbon, Portugal.

Sekine S., Grishman R., 1995, A Corpus-based Probabilistic Grammar with Only Two Non-terminals. *Fourth International Workshop on Parsing Technology*; Prague.

Sekine S. 1998. Corpus-based Parsing and Sublanguage Studies, *Ph.D. Thesis*, New York University.

Sha F. Pereira F., 2003. Shallow parsing with conditional random fields. In *Proc of HLT-NAACL*. ACL.

Tjong Kim Sang, E. F. 2000. Text Chunking by System Combination. In *Proc of CoNLL-2000*, Lisbon, Portugal.

Ratnaparkhi A., Reynar J., and Roukos S., 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In Proc *of the ARPA Human Language Technology Workshop*.

Woods W. A. 1970. Transition Network Grammars for Natural Language Analysis, *Comm of the ACM*, 13, 10.

Implementing a lexicalised statistical parser

Corrin Lakeland, Alistair Knott
Department of Computer Science
University of Otago
New Zealand
{lakeland,alik}@cs.otago.ac.nz

Abstract

Statistical parsers are extremely complex systems, yet papers describing them almost always only discuss theoretical issues instead of implementation issues. This paper attempts to address the imbalance by describing the implementation issues faced in building a state-of-the-art statistical parser. In the process, we will describe our own implementation of a statistical parser.

1 Introduction

Between 1996 and 1999, Michael Collins developed a statistical parser (Collins, 1996; 1999) which has become tremendously influential in NLP. Collins' thesis and published papers discuss the theoretical underpinnings of his system in a great deal of detail; he devotes considerable space to describing and justifying the grammar formalism and the probability model which his parser uses. His description of his parsing algorithm is much less detailed; it is given as a set of pseudocode routines in an appendix. However, the scale and complexity of a lexicalised statistical parser is such that implementing this pseudocode presents significant software engineering difficulties. The pseudocode actually disguises many of the interesting optimisations present in Collins' system. As far as we can tell, nobody has published how to actually implement such a system. The aim of this paper is to describe the important software engineering issues involved in implementing a lexicalised statistical parser, using the parser we implemented as an example. What we found was that if you ever consider performance a secondary consideration then the parser will go so slow as to be impossible to debug. Because of this, we will concentrate on the efficiency of algorithms not so much to improve on Collins', but just to get a working system.

We will begin in Section 2 by describing Collins' probability model. In Section 3 we describe Collins' parsing algorithm in high-level detail. The remainder of the paper describes our way of implementing the difficult parts of this algorithm. The chart data structure is described in Section 4; the generation of probabilities from the probability model in Section 5; the search strategy used by the parser is described in Section 6; Some general advice about software engineering in building a statistical parser is discussed in Section 7. And we conclude by noting that our parser performs almost the same as Collins' in Section 8.

2 Collins' probability model

The key idea in any statistical parser is to associate probabilities with grammatical rules. The probability of any given parse tree is then simply the product of the probabilities of all the rules applied in creating this tree. However, in practice, the probability of a parse tree being the correct parse of a sentence depends not just on the rules which are applied, but on the words which appear at the leaves of the tree. To illustrate, consider this well-known example of syntactic ambiguity:

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

The PP *with the telescope* can either modify *the dog* (as a relative clause) or the verb *saw* (as an adverbial). Intuitively, the latter reading should be preferred; we expect events of seeing to involve telescopes more frequently than dogs. The notion of a **lexical head** is useful in spelling out this intuition. We expect a VP headed by the verb *saw* to be quite frequently modified by a PP involving the word *telescope* in a representative corpus, while we expect a NP headed by *dog* only rarely to be modified by a PP involving the word *telescope* in such a corpus. We begin in Section 2.1 by describing a formalism for capturing this idea.

2.1 Unary and dependency productions

How can we modify our grammar to include the appropriate lexical information? A useful solution, originally proposed by Black *et al.* (1992), basically involves a huge increase in the number of phrases in the grammar. Instead of simply having a phrase NP, we need one phrase for each possible headword of an NP: i.e. NP-headed-by-dog, NP-headed-by-telescope, and so on. At this point, unfortunately, we are faced with a data sparseness problem: we are unlikely to find sufficient counts for individual productions, even with a very big corpus. The problem is largely due to **Zipf's law**; most words in the language only occur very infrequently, so most grammatical categories, when tagged with an open-classed headword, will be fairly rare. The problem is compounded by the fact that many grammars allow a node to take several children. If each child is already rare, then the combination of n such children will be exponentially so. With low counts, we cannot be confident in the probabilities we derive.

We will consider the problem due to Zipf's law in Section 2.3. However, the problem of multiple children can be addressed by finding a way of splitting a parse tree into events that are smaller than single context-free rule applications. A useful idea, originally proposed by Magerman (1995), is to break each single rule application into several components: a **unary production** which takes a phrase and generates its head constituent, and a set of **dependency productions** which take a phrase and its head constituent, and generate the remaining child constituents, either to the left or the right of the head. The occurrence of a parent node decomposing into a set of children is now represented using the kinds of events shown in Figure 1.

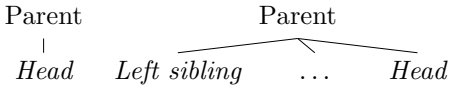


Figure 1: Unary and dependency productions

The conditional probabilities we are interested in are the probability of a head constituent given its parent (for a unary production) and the probability of a sibling constituent given its parent and its head (for a dependency production). These probabilities can be estimated from relative frequencies of events.

$$\mathcal{P}(Head|Parent) = \frac{Count(Head, Parent)}{Count(Parent)}$$

The notation here needs some explanation. If you know the parent and you are trying to derive the probability for a given head, you can estimate the probability by counting the number of times that head occurs as the head of that parent, and dividing by the total number of times that parent occurs. The categories 'head' and 'parent' are descriptions of constituents, which could be given at different levels of detail. If we are building a lexicalised grammar, these descriptions will each include a headword, as well as a grammatical category. The data sparseness problem due to Zipf's law is now reduced; unary productions only involve one lexical item, and dependency productions only involve two.

2.2 Collins' probabilistic grammar

Collins' probabilistic grammar is expressed in terms of unary and dependency productions, as just described. Of course, he includes more information about these productions than we expressed in the above equations. As well as a **head word** for each constituent, he includes information about the part of speech of this word: the **head tag**. He calls the grammatical category the **head nonterminal**, to distinguish it from the tag. For the dependency productions, he distinguishes between **complement** and **adjunct** siblings of a head (a complement sibling is tagged '-C'), and includes a **subcategorisation frame** representing the complements that the head needs. He also includes a measure of the **distance** between the head and sibling constituents. To explain Collins' representation of trees, we once again refer to a simple example tree — see Figure 2. In this figure, only the nodes pointed at by arrows take part in representing events.

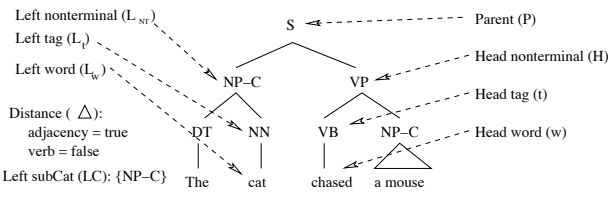


Figure 2: Collins' event representation

This figure shows the fields used to represent the dependency event of an NP-C attaching as a left sister to the head VP of a parent S node. Collins stores this event by encoding the terms pointed to with arrows in the figure. At this point, the event simply is the co-occurence of these values for these data fields. To represent a unary production, such as the generation of VP from the parent S node, we just need the terms on the right-hand side of Figure 2. To represent a subcat production, such as the generation of the VP's left subcategorisation frame (in this case, a bag containing one item, NP-C), we use the same terms as the unary event, plus one additional field: a bag of nonterminals.

The probabilities we need to compute for unary, subcat and dependency productions can now be given more precisely, in the following three equations.

$$\mathcal{P}_{unary}(H \mid P, w, t)$$
$$\mathcal{P}_{subcat}(LC \mid H, P, w, t)$$
$$\mathcal{P}_{dep}(L_n t, L_w, L_t \mid P, H, w, t, \Delta, LC)$$

2.3 Collins' event representation

To compute the above probabilities, we need to derive appropriate relative frequencies of events occurring in the WSJ corpus. Given that some events might be rare, or nonexistent, in this corpus, Collins uses a **backoff** technique. Basically, we estimate the probability of a very precisely specified event by looking up a less precisely specified one. For instance, the unary event in Figure 2 where it was decided VP should form the head of a sentence, would be as follows:

$$\mathcal{P}_{unary}(VP \mid S, chased, VB)$$

while a backed-off version of the event might leave out the head word *chased*:

$$\mathcal{P}_{unary}(VP \mid S, VB)$$

Collins makes use of three levels of backoff for all the events he represents: Level 1 contains all the terms pointed to in Figure 2, Level 2 drops a headword, and Level 3 drops everything except for nonterminals. To compute the probability of an event, its numerator, denominator, and a weighting factor are looked up at all three levels of backoff, and the resulting probabilities are interpolated using the weighting factors. In summary, to derive the probability of an event, we must perform nine separate lookups of event counts in a database of events derived from the WSJ corpus.

2.4 Preprocessing the WSJ

The WSJ corpus is the main source of training data for statistical parsers. The trees in the WSJ do not include information about headwords or complements, both of which are fundamental to Collins' approach. So Collins first has to add this information, using heuristics based on the syntactic and semantic annotations which are present. These heuristics are too numerous to doucment here, but a typical example would be that noun-phrases search right to left for their head child and prefer nouns. After applying the heuristics, he then has to transform the trees into a database of events to be counted when computing probabilities. While these preprocessing routines are relatively simple compared to the parser, Collins has not released his preprocessor, and it is the least documented part of the system, so it is useful to document it; interested readers are referred to Lakeland (forthcoming), where preprocessing code is given in detail.

3 Collins' parsing algorithm

We now present the high-level structure of the parsing algorithm.

Very briefly, in words, here is what happens. In the top-level function `parse`, we begin by initialising the chart with a set of **complete** edges, each of which is one word from the input string, and a set of **incomplete** edges, each of which is created by one or more unary productions on one of the complete edges. A complete edge is one that will not be expanded further. Then we call the function `combine` on every set of adjacent edges in the chart. The `combine` function attempts to join every pair of adjacent edges, using a dependency production, where the parent edge is incomplete and the child edge is complete. This is done by the functions `join_follow` and `join_precede`. Whenever two edges are successfully joined, the new complex edge is added to the chart; then this edge is expanded using unary productions (considering both a single unary production and chains of two or three unary productions), and adds these to the chart. This is done by the function `add_singles_stops`. The new edges which have been added to the chart will be found by subsequent calls to `combine`. Eventually, edges will be created which span the whole input string; when we have found all of these, we select the complete parse with highest probability.

```
parse(sentence)
  initialise(sentence)
  for start = 0 to length
   for end = start + 1 to length
    for split = start + 1 to end {
      left = spanning(start,split)
      right = spanning(split+1,end)
      combine(left,right)
    }

combine(left, right) {
 foreach (l left)
 foreach (r right) {
   if (!l.complete && r.complete)
     joined = join_follow(l,r)
   if (l.complete && !r.complete)
     joined = join_precede(l,r)
   add_singles_stops(joined)
 }
}

join_follow(left,right) {
  e = new edge(left)
  e.add_child(right,at_end)
  e.prob *= right
  e.prob *= dep_prob(left,right)
  chart.add(e)
}

join_precede(left,right) {
  (as per join follow)
}

add_singles_stops(edges,depth=5) {
  if (depth == 0) return edges
  foreach (e edges)
    e_stop += add_stop(e)
  foreach (e e_stop)
    e_ns += add_singles(e)
  add_singles_stops(e_ns,depth-1)
}

add_singles(e) {
  foreach (parent nonterminals)
    foreach (lc subcats)
      foreach (rc subcats)
        if(grammar(e,parent,lc,rc)) {
          result = unary(e,parent,lc,rc)
          chart.add(result)
          results += result
  return results
}
```

Figure 3: Simplified parser pseudocode

Looking at the pseudocode it is hard to see where the implementation difficulty lies. The answer can be seen by counting loops: `parse` contains three loops; `combine` contains two; `add_singles_stops` contains one; and `add_singles` contains three. Since these functions are all nested, the parser has nine nested loops. To address this complexity, two things are needed. Firstly, in general, we want to implement everything *efficiently*, so that the algorithm is as fast as possible. But as well as efficiency considerations, we also need to build some genuine shortcuts into the algorithm, by applying **search heuristics** which discard edges unlikely to be in the final parse. Search heuristics are applied in two places in the parsing algorithm: a **beam search** algorithm is used in `add_singles_stops` to stop unlikely unary productions from being generated; and **dynamic programming** is used in the chart insertion routine to discard an edge if a more probable edge covering the same span already exists. These two issues — code efficiency and implementating heuristics — are what we focus on in the remainder of the paper.

4 The chart data structure

The goal of the chart is to store and provide access to all the edges covering each span of the input string. The grammar is very large because it contains a separate rule for each headword in each production, and because of this a great many edges cover each span. This means that the wrong choice of chart data structure will make parsing impossibly slow.

The most natural way of implementing such a data structure would be as a three dimensional array, in which the first two dimensions specify the start and end of the span respectively, and the last dimension stores the edges. Unfortunately, we do not know how many edges will be needed for any given span of the input string, which makes allocating such an array impossible (or at least extremely wasteful). To get around this problem, note that the flow of control of the parsing algorithm means that edges with a given start and end position in the input string are added consecutively. This means that we can store the chart as a huge one dimensional array of edges, with a two dimensional **index array** of pointers indicating where the set of edges associated with each span are stored. There is then no wasted space in the chart, and we still have constant time access to any span.

A related optimisation comes from noting that the control structure of the `complete` function means that we always process one complete edge and one incomplete edge, so it would be more efficient if we could loop over all complete edges and all incomplete edges separately. It thus makes sense to have two separate charts, one for complete edges and one for incomplete edges.

Another optimisation relates to the use of a very simple dependency grammar within the `complete` algorithm. Whenever two edges are joined, we must compute the dependency probability for the join operation. If this probability is zero, there is no need to store the edge. In general we cannot predict when a dependency event will have a probability of zero in advance, but there is one exception: we can look at the nonterminal head of the parent and sibling, and if this combination was never seen in the training corpus then we know the dependency event will have a probability of zero. The optimisation involves precomputing a simple dependency grammar specifying which nonterminal categories are found in dependency productions in the WSJ. (For instance, the top production in Figure 2 would allow a parent S whose head child is VP to have an NP-C as a left child.) Now, in the `combine` function, we iterate over every left and every right edge consistent with this simple grammar. To permit efficient access to grammatically consistent edges in the chart, we add a third dimension to the index array, to hold the edge's parent nonterminal.

Another kind of optimisation in the chart comes from noting that it is possible for two different phrases to have the same representation as events in Collins' probability model. For instance, note in Figure 2 that Collins' event language makes no reference to the *Det* phrase associated with the subject NP-C *cat*. Since the goal of the parser is to find the single best parse, if we ever have two phrases with the same representation at a given span in the chart, we can simply discard the one with lower probability; it will never be involved in the best parse of the sentence. This is known as the **Viterbi optimisation**. A closely related optimisation is to discard any edge with a probability significantly lower than the best edge over this span, since it is very unlikely that a parse involving this edge will outscore a parse involving the most likely edge for this span. These last two optimisations are examples of the **dynamic programming** approach.

5 Computing probabilities

As mentioned in Section 2.3, to compute probabilities, Collins derives nine counts — that is, he looks up the number of times nine different subevents have occurred in the database of events derived from the WSJ corpus. This database cannot be stored as an indexed array since there is no obvious index; we therefore make use of the standard way of storing large data sets, hash tables. The training data requires storing around fifty million events, and parsing a single sentence requires many millions of probabilities to be computed. Because performance is so critical it is worth being careful about the implementation details.

Firstly, there is no need to store a hash table for every type of event. Instead we can use a single huge hashtable and include the type of event in the key. This does not make the system inherently faster but does make it much easier to control the density of the hashtable which will lead to performance improvements. Secondly, it is conventional in hashtables to store both the key and value in the table so that hash collisions can be detected but here the hash key is many bytes and so it is more appropriate to just ignore collisions and accept that probabilities will be slightly incorrect. Finally, over ninety percent of probabilities computed in the parser are used more than once, so by storing all generated probabilities in a 'cache' hashtable, the speed of the whole system can be improved by an order of magnitude.

6 Implementing the beam search

The function `add_singles_stops` includes three nested loops and is itself called recursively about five times. While none of these loops is dependent on the size of the input sentence (i.e. the function is $O(1)$), an unconstrained implementation would result in approximately 2000^5 edges being created (the number of nonterminals times the number of possible left subcategorisation frames times the number of right subcategorisation frames, recursively called five times). Even if these edges were discarded by the chart on creation, the time taken to create them would make it impossible to parse a simple sentence. To resolve this, Collins only expands edges likely to be part of the final parse. Collins' thesis notes he uses a constrained best first search known as a **beam search** for this

process. The benefit of this is that instead of an unmanageable number of nodes being created, perhaps only a few hundred are created (of which dynamic programming in the chart will still discard all but a handful).

Search generally involves creating new nodes for each child being expanded. But as is mentioned in Section 7, allocating memory is a computationally expensive operation and is undesirable in a program where efficiency is critical. Since a beam always has exactly n nodes on it, it seems intuitively obvious that beam-search could be implemented without allocating memory but it proves surprisingly difficult to do efficiently. Our implementation of beam search uses **skiplists** (Pugh, 1989). Skiplists are a variant on linked lists in which a number of 'next' pointers are kept on each node instead of just one. These extra pointers allow the algorithm to 'skip' along the list and lead to insertion and access times of $O(\lg n)$ (the same as binary search, but much simpler to implement). As an extension to Pugh's idea, we implemented double-ended skiplists (analogous to doubly-linked lists). This gives $O(1)$ access and insertion to both the start and end of the list.

Having developed a suitable data structure, we apply it to beam search. By allocating $n+1$ nodes for a beam of length n, we can provide `add_singles` with an empty node in $O(1)$ by simply returning the last node in the list (technically, these functions are not $O(1)$ but $O(lg(lg(n))$ due to pointer management code, but this closely approximates 1 for even huge values of n). In practice, insertions are almost always at the start or the end of the list (both approximately $O(1)$). When they are at other parts of the list, insertion is an $O(\lg n)$ operation.

The obvious comparison for this approach would be using a heap, as a heap is the data structure most commonly used to implement priority queues. Using an array based heap (since the size of the queue is bounded) we can access the front in $O(1)$, but to remove the last node and reinsert is $O(\lg n)$. Compared to this implementation, skiplists are somewhat more efficient at $O(\lg(\lg n))$.

Overall, double-ended skiplists have proven to be an interesting and efficient method of implementing beam-search for large n. Where n is low, it is probably more efficient to simply use a doubly-linked list but Collins' noted he used a beam size of 10,000, and so a more sophisticated approach is called for. After implementing the skiplists search, Collins released his code and it is very interesting to compare his approach; it turns out he does not actually implement classical beam-search, but instead uses an array of edges being expanded with a threshold — if an edge is a certain amount worse than the best then it is discarded. This is significantly simpler and somewhat more efficient than my approach. However it would perform very poorly anywhere where the heuristic evaluation improves as we move away from the start state.

7 General software engineering issues

The core difficulty in implementing a statistical parser is that it processes a vast amount of data. The event file created by the preprocessor contains perhaps fifty million events; the beam is searching through perhaps ten thousand local possibilities; and the chart contains hundreds of thousands of ambiguous partial parses. All this means that we must keep code as efficient as possible throughout the development process, or the parser will simply fail. In addition, that sophisticated code and data file verification techniques are crucial, because small bugs can have far-reaching consequences. In this section, we present some of the software development lessons we have learned in building our parser.

7.1 Start by solving a smaller problem

A lexicalised statistical parser is a very complex system, where a single poor choice results in a program that is too slow to test. However, building a part-of-speech (POS) tagger has many of the same issues as a statistical parser but without the asymptotic complexity. We found it was useful to begin by building a full reimplementation of Collins' probability model which was only used for POS tagging (Lakeland and Knott, 2001). This enabled about half of the system to be verified.

7.2 Choice of programming language

Initially our parser was implemented in LISP, because it is a language ideally suited to both tree processing and prototyping. It was far too slow, and while various optimisations could make it fast, it was obvious that the easiest approach would be to reimplement using a language capable of breaking the rules built into high-level programming languages, in which allocation of memory can be done by hand, pointers can be manipulated directly, and shortcuts

can be hacked into the control structure of the program.

It is worth explaining why direct memory management is essential. Allocation of memory is an extremely slow function and any program desiring efficiency must not allocate memory inside its inner loop. By preallocating data structures (e.g. allocating all the memory for the chart and the beam before parsing begins), it is possible to avoid any memory allocation during the core parsing loops, saving a great deal of time. Languages such as Java, C# and Python are therefore a bad idea; their automatic memory management (normally a key selling point) is precisely what we need to sidestep to implement an efficient parser. Consequently, like Collins, we chose to implement the parser in C, which provides low-level memory management support. (However, for preprocessing the corpus, we stuck with LISP, since it is not time critical.)

7.3 Version control

Anybody building a nontrivial program will use a source code control system such as CVS or subversion. But we found that naive use is insufficient – for instance we frequently found improvements to the preprocessor would break the parser since it depended on the older format for the data files. We also needed to make use of 'branching'.

Another related step was the development of a build script. There are a large number of steps involved in converting the treebank and other data into a format suitable for parsing. It is relatively easy to perform these steps sequentially. However that means any change to one of the earlier steps (such as a tweak to the tokeniser) requires every subsequent step to be repeated. Since there is usually output from the previous version lying around, it was often the case that output files from different versions of the code would be used at the same time — leading to subtle errors.

Finally, version control only applies to files but we often found that we needed to write *almost* identical blocks of code, but often we could not write the code as a general function which decided its behaviour based on arguments and writing the same code twice invariably leads to bugs being fixed in one version but not in another. Our solution to this was to use source code preprocessing so that our single 'meta' version generates multiple functions, each with slightly different logic. We used the tool `funnelweb` for this purpose.

7.4 Efficiency versus debuggability

It is often the case that the most efficient data structure is harder to debug. For instance, our hash keys can be easily compared to the data used in generating the key and so a bug in key generation is easily identifiable while Collins' keys bear too little correspondence to data and so cannot be easily debugged but they can be generated faster. Similarly, Collins uses array offsets to refer to edges where we use pointers which will make our code slightly faster, but it makes tracking an edge through parsing much easier in Collins' system.

'Magic numbers' are another area in which bugs can easily creep into the system — for instance, setting the maximum number of nonterminals to 100 might be correct at first, but later adding -C complements could easily overflow this and lead to data corruption. We managed to avoid many of the problems here by automatically generating the declarations of constants from the input files, so any change to the input files will automatically appear in the source code. Similarly, many functions in the probability model take a dozen or so parameters and getting these in the wrong order will not cause any typecast errors since they are all integers, it will just generate invalid output. This problem was avoided by implementing basic datatypes as different classes so that incorrect orders does result in typecast errors. Curiously, Collins uses magic numbers everywhere and I often wondered how he managed to debug them in his parser.

7.5 Debugging methodology and test suites

Debugging the parser turned out to be extremely difficult. It is not so hard to detect the presence of a bug, but isolating where in the process this bug is introduced can take a week. In a normal program a bug can be isolated by stepping through its operations on simple input but with a statistical parser there are far too many operations to do this for even the most trivial input. The best approach we found was to spend a lot of effort detecting bugs as soon as possible after they are introduced. For instance, if a bug in the tokeniser leads to a small number of events not being generated then it is critical to detect this problem during the generation of the event file rather than during the execution of the parser.

In order to facilitate this, after testing every function we wrote an automated test suite that rechecks functions every time the system is built. For example, the probability model can be checked by comparing the counts it derives to those produced with `grep`. If a bug is later introduced in the input to this function then it will likely cause some testcase to fail. Similarly, the system is liberally scattered with `assert` statements that perform everything from internal bounds checking to checking that the skiplist is in sorted order and still has n elements. As a last resort, we also made extensive use of the `memprotect` kernel call to lock any data that was not currently being edited (such as the hash tables). This allowed us to catch a number of bugs where we had forgotten an assertion.

A final comment is that we found high-level debugging to be much less useful than low-level debugging. For instance, by examining the sentences the parser performs poorly on it may be possible to infer it has a problem. But this approach turned out to be significantly more time-consuming than simply verifying every function independently, mainly because the parser was too big to find where the bug was after the high-level approach found the existence of a bug.

8 Conclusion: results of our own parser

The parser we implemented performed almost identically to Collins' as regards precision and recall (84.5% as opposed to 85%). In over 95% of cases, our parser produces exactly the same output as Collins', with differences partly caused by small undocumented tweaks Collins made, such as using the headword from a child instead of the parent during coordination, and partly due to some late design changes made as our understanding of Collins' algorithm improved. Our system is significantly more modifiable than Collins. This is because it was designed with that in mind, and also because all of the seperate components used are tightly seperated out into different classes with well specified interactions. Because of this, my system is well suited as a platform for further research.

References

Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R., and Roukos, S. (1992). Towards history-based grammars: using richer models for probabilistic parsing. In M. Marcus, editor, *Fifth DARPA Workshop on Speech and Natural Language*, Arden Conference Center, Harriman, New York.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. Ph.D. thesis, Computer Science Department, University of Pennsylvania.

Lakeland, C. (forthcoming). *Lexical Approaches to Backoff in Statistical Parsing*. Ph.D. thesis, Department of Computer Science, University of Otago.

Lakeland, C. and Knott, A. (2001). Pos tagging in statistical parsing. In *Proc. of the Australasian Language Technology Workshop*, Sydney, Australia.

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

Pugh (1989). Skip lists: A probabilistic alternative to balanced trees. In *WADS: 1st Workshop on Algorithms and Data Structures*.

Controlled Natural Language meets the Semantic Web

Rolf Schwitter and Marc Tilbrook
Centre for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
{schwitt|marct}@ics.mq.edu.au

Abstract

In this paper we present PENG-D, a proposal for a controlled natural language that can be used for expressing knowledge about resources in the Semantic Web and for specifying ontologies in a human-readable way. After a brief overview of the main Semantic Web enabling technologies (and their deficiencies), we will show how statements and rules written in PENG-D are related to (a subset of) RDFS and OWL and how this knowledge can be translated into an expressive fragment of first-order logic. The resulting information can then be further processed by third-party reasoning services and queried in PENG-D.

1 Introduction

> *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." (Berners-Lee et al., 2001)*

This claim falls short, since the languages – based on RDF – that have been designed for making statements about Web resources and for specifying ontologies are not built for human consumption. These languages have been developed with machine processability and automatic information exchange in mind (Manola and Miller, 2004; Smith et al., 2004). They are painful to read, write and understand for non-specialists.

To overcome the disadvantages of machine-processable formal languages based on RDF and full natural languages that are far too expressive for the task at hand, we will introduce PENG-D, a controlled natural language that reconciles rigid formality and natural familiarity.

In a nutshell, a controlled natural language is a subset of a natural language that has been restricted with respect to its grammar and its lexicon. Grammatical restrictions result in less complex and less ambiguous sentences, while lexical restrictions reduce the size of the lexicon and the meaning of the lexical entries for a particular domain (Huijsen, 1998). Using a controlled language for knowledge representation, specification texts become easier to read and understand for humans, and easier to process for machines (Schwitter, 2004).

Traditionally, controlled natural languages have been classified into two major categories: human-oriented and machine-oriented controlled natural languages. Human-oriented controlled natural languages have been designed to improve the readability and understandability of technical documents, particularly for non-native speakers. An example are aircraft maintenance documents in the aerospace industry (AECMA, 2004). Machine-oriented controlled natural languages have been developed to ameliorate the quality of machine translation for technical documents (Mitamura, 1999) and for writing specification texts that can be easily translated into a formal language (Fuchs et al., 1999; Schwitter, 2002; Sowa, 2004).

Most of these machine-oriented controlled natural languages are defined by their translation into first-order logic that automatically restricts their expressivity to a small subset of constructions compared to full English.

Since it is very likely that the emerging Semantic Web will finally rely on a variant of description logic as knowledge representation formalism and since description logic is a decidable fragment of first-order logic (Grosof et al., 2003), a machine-oriented controlled natural language is required that is a compromise between expressive power, complexity and computability.

PENG-D is a proposal for such a machine-oriented controlled natural language that fulfils these requirements.

2 Semantic Web Enabling Technologies

The Semantic Web aims at making Web resources better accessible to automated agents by adding information (meta-data) that describes Web content in a machine-readable way. It is based on RDF, which relies on eXtensible Markup Language (XML) for syntax, Uniform Resource Identifiers (URIs) for naming, and on the RDF Vocabulary Description Language: RDF Schema (RDFS) for describing meaning and relationship of terms (Manola and Miller, 2004). RDF and RDFS form the lowest layers in the functional architecture of the envisioned Semantic Web. Web ontology languages and rule languages are expected to build the next two layers on top of RDFS to supply richer modelling primitives and reasoning support. Unfortunately, the relationships between RDFS and Web ontology languages are not clearly specified (Horrocks and Patel-Schneider, 2003).

2.1 RDF

RDF is a datamodel for representing meta-data about Web resources. The basic RDF model contains just the concept of a statement, and the concept of reification - making a statement about a statement. RDF is based on the idea of using URI references to identify the resources referred to in a RDF statement. RDF uses a particular terminology for talking about the various parts of a statement. The part that identifies what the statement is about is called the *subject*. The part that identifies the property of the subject that the statement specifies is called the *predicate* and the part that identifies the value of that property is called the *object*. In RDF, the English sentence

Nic is a human.

could be represented by a statement having three URI references:

http://www.example.org/about-nic#nic
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.example.org/biology#Human

RDF models statements as nodes and arcs in a graph. In this notation a statement is represented as a node for the subject, a node for the object, and an arc for the predicate directed from the subject node to the object node. RDF/XML is a graph serialisation syntax and provides a machine-processable way to record and exchange graph-based RDF statements (Beckett, 2004). Here is an excerpt of an RDF/XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<rdf:RDF
  xmlns:rdf=
   "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://www.example.org/about-nic#"
  xmlns:bio="http://www.example.org/biology#"
  xmlns:con="http://www.example.org/contact#">

 <bio:Human rdf:about=#nic>
  <con:title>Dr.</con:title>
  <con:name>Nic Miller</con:name>
  <con:name>Nicolas Miller</con:name>
  <bio:pet>
   <bio:Labrador>
    <con:name>Rex</con:name>
   </bio:Labrador>
  </bio:pet>
 </bio:Human>

</rdf:RDF>
```

The *rdf:RDF* element is the root of the RDF document and defines the XML document to be an RDF document containing a reference to the *xmlns:rdf* namespace and to three other namespaces. Qualified names such as *bio:Human* or *con:title* are shorthands for full URI references whereas an URI reference with an empty prefix stands for the current document.

RDF/XML is the standard interchange format for RDF on the Semantic Web, but there exists a non-XML serialisation alternative called Notation3 (N3) that is generally accepted to be easier to use and is convertible to RDF/XML and vice versa (Berners-Lee, 2001). N3 was designed with human-readability in mind and was created as an experiment in optimising the "expression of data and logic in the same language" (Palmer, 2002). This optimisation can be seen as a first step towards the design of a controlled natural language. Below is the automatic translation of our RDF/XML excerpt into N3 using CWM, a forward chaining reasoner that supports data conversion (Berners-Lee, 2003):

```
@prefix : <http://www.example.org/biology#>.
@prefix con: <http://www.example.org/contact#>.

<#nic> a :Human;
  :pet [a :Labrador; con:name "Rex"];
  con:name "Nic Miller", "Nicolas Miller";
  con:title "Dr.".
```

Without doubt this representation is easier to read, to create and to understand by humans. However, there are still a couple of nota-

tional particularities which non-specialists have to master.

It would be much easier if a human could express this information in a well-defined subset of natural language, for example as follows:

Nic is a human who has Dr. as title.
Nic's name is Nic Miller and Nicolas Miller.
Nic's pet is a labrador that has the name Rex.

Obviously the writing process would need to be supported by an intelligent writing assistant that tells the user which constructions are admissible.

2.2 RDF Schema

So far, we have only addressed the syntax of RDF and have not considered the meaning of terms used in XML/RDF and N3. RDF itself provides no means for defining application-specific classes and properties. Instead, such classes and properties need to be described via RDF Schema (RDFS), an RDF-based vocabulary description language (Brickley and Guha, 2004). RDFS does not provide an application-specific vocabulary for classes and properties but it offers the facilities that are needed to describe such classes and properties. Basically, RDFS provides the means for constructing a type hierarchy for RDF.

2.2.1 Describing classes

A class in RDFS corresponds to the generic concept of a type or category. In N3 we can declare a class such as *Dog* in the following way:

```
:Dog a rdfs:Class.
```

This states that *Dog* is a type of *rdfs:Class*. If there is a hierarchical relationship between two classes, then this can be stated, for example, as follows:

```
:Labrador a rdfs:Class; rdfs:SubclassOf :Dog.
```

This says that *Labrador* is a *rdfs:Class* which is a subclass (*rdfs:SubclassOf*) of the class *Dog*.

2.2.2 Describing properties

In RDFS, properties are used to declare a relationship between two things and are described as instances of class *rdf:Property*:

```
:pet a rdfs:Property.
```

This states that *pet* is a type of *rdfs:Property*. RDFS also provides vocabulary for describing how properties and classes are intended to be used together. For example in

```
:pet rdfs:domain :Human; rdfs:range :Animal.
```

the *rdfs:domain* property is used to indicate that the values of the subject of the property *pet* are instances of the class *Human* and the values of the object of the same property are instances of the class *Animal*.

In some respect, RDFS is a very limited "knowledge representation" language with few modelling primitives and more expressive power is necessary to describe Web resources in sufficient detail. Apart from this expressive restriction, RDFS has a non-standard and non-fixed layer meta-modelling architecture, which makes some elements in the model have dual roles in the RDFS specification. This makes it extremely difficult to layer more expressive ontology and rule languages on top of RDFS (Pan and Horrocks, 2003).

2.3 OWL

The recognition of RDFS's limitations led to the development of OWL. The OWL language provides three increasingly expressive sublanguages (OWL Lite, OWL DL and OWL Full) that have been developed as a trade-off between expressivity and efficiency. OWL Lite supports users who need a classification hierarchy and simple constraint features combined with desirable computational complexity. OWL DL is closely related to description logic (DL) and supports users who want the maximum expressiveness without losing computational completeness and decidability. OWL Full is designed for users who require the meta-modelling facilities of RDFS with no computational guarantees (Smith et al., 2004).

OWL uses the same syntax as RDF (and RDFS) to represent ontologies and uses RDFS resources directly whenever the required functionality already exists in RDFS. For example, OWL uses *rdfs:subClassOf* to assert subclass relationships and specific classes and properties to extend RDFS functionality, e.g., the *equivalentClass* axiom is used to link a class description to another class description:

```
<owl:Class rdf:ID="Man">
  <rdfs:subClassOf rdf:resource="#Human" />
  <owl:equivalentClass rdf:resource=
    "#MaleAdult" />
</owl:Class>
```

While syntactic layering of the these languages satisfies the design requirements of the Semantic Web, the semantic layering is more problematic, since OWL is largely based on

DL which is a decidable fragment of first-order logic. The semantics of a DL is normally given by a standard first-order model theory, while the semantics of RDFS is given by a non-standard model theory, because of its meta-modelling facilities (Hayes, 2004). This incompatibility leads to serious problems when trying to layer first-order based languages on top of RDFS, since it is not clear how applications would be able to reason with these languages.

3 Description Logic Programs

Instead of relying on RDFS, conventional first-order logic (FOL) has been proposed as the semantic underpinning of the Semantic Web. From a theoretical point of view, it is well known that reasoning in FOL is undecidable. However, there are many decidable subsets of FOL that have been extensively studied and many reasoning systems have been developed for FOL and its sublanguages. Although this FOL-based approach is not immediately compatible with RDFS, it is compatible with a simplified version of RDFS – "the FOL subset of RDFS"(Horrocks and Patel-Schneider, 2003).

Recently, the fusion of DL ontologies and rule languages (Horn logic) has been studied and this effort resulted in the Description Logic Programs (DLP) paradigm (Grosof et al., 2003). DLP is defined as the expressive intersection of OWL Lite and Horn logic (without negation and without function symbols) and captures a significant part of OWL Lite. The interoperability between OWL Lite and Horn logic is expressed by a meaning preserving bidirectional translation of premises and inferences from the DLP fragment of OWL Lite to Horn logic, and vice versa from the DLP fragment of Horn logic to OWL Lite. In this paradigm, a mapping function translates for example the DL axiom

DL: $A \sqcap \exists R.C \sqsubseteq B \sqcap \forall P.D$

into Horn logic (HL) rules

HL: b(X) ← a(X), r(X,Y), c(X).
d(Z) ← a(X), r(X,Y), c(X), p(X,Z).

and vice versa (Grosof et al., 2003). Please note that the Horn rules here correspond to definite Horn clauses (with exactly one positive literal). All variables are universally quantified at the outer level and have scope over the entire clause and only fact-form conclusions (as in logic programs) are available.

4 PENG-D

PENG is a machine-oriented controlled natural language that has been developed to write specifications for knowledge representation (Schwitter, 2002; Schwitter et al., 2003; Schwitter, 2004). While PENG was designed for writing specifications that are first-order equivalent, the proposed language PENG-D has formal properties that are equivalent to DLP. Although PENG-D is weaker than PENG, it provides a clear computational pathway to layer more expressive constructions on top of it.

4.1 Architecture of PENG-D

The planned architecture of the PENG-D system looks similar to the PENG system but offers support for ontology construction. The PENG-D system consists of four main components: a look-ahead text editor, a controlled language (CL) processor, an ontology component and an inference engine.

4.2 The text editor

The text editor can be used either to construct ontologies (TBox mode) or to assert knowledge about a specific domain (ABox mode). The user does not need to learn the rules of the controlled language explicitly, since the writing process of the user is guided by the look-ahead text editor. The first thing that the user sees, for example, after opening the text editor in the ABox mode, are look-ahead categories:

[Proper Noun, Determiner, [There is a]]

After entering, for example, the name *Nic*, the look-ahead editor displays further look-ahead categories:

Nic *[Verb, Relative Pronoun, ['s]]*

The user can now type the word that belongs to one of these look-ahead categories directly into the editor or select it from a context menu. This context menu contains entries that have been derived from the ontology used. Namespace definitions are handled via the editor.

4.3 The grammar of PENG-D

The grammar of PENG-D describes how simple sentences are built and provides constructors to join simple sentences into complex sentences. Anaphoric relations between nominal constituents in sentences can be expressed via definite noun phrases, proper nouns, and variable names.

4.3.1 Simple sentences

Simple sentences are used for making statements in the ABox. Note that the result of these statements are always ground terms containing no variables. In a first approximation, we can describe the structure of simple sentences in the following way:

sentence	→	subject + predicate
subject	→	nominal head
subject	→	specifier {+ pre-nominal modifier} + nominal head {+ post-nominal modifier}
predicate	→	verbal head + complement

These rules need to be carefully restricted to be useful for our purpose:

Specifier. There are only two approved determiners that are available in the specifier position: the definite determiner *the* and alternatively the indefinite determiner *a* (with a "specific reading"):

The *dog Rex ...*
A *dog Rex ...*

Apart from that, a possessive construction such as *Nic's* can be used in the specifier position, for example:

Nic's *dog Rex ...*

Once a specific instance has been introduced, we can refer to it:

The dog Rex ... ***The dog*** *...* ***Rex*** *...*

Pre-nominal modifier. A pre-nominal modifier can only consist of one single adjective in the positive form. Adjectives can be used to give additional information about a resource (and are interpreted intersectively):

The ***friendly*** *dog Rex ...*

Nominal head. The nominal head must be realised by a proper noun, a common noun or a relational noun. Common nouns and relational nouns (in subject position) always need a determiner. The structure of nouns can be simple or compound like in full English:

Nicolas Miller *...*
The ***hunting dog*** *Rex ...*
The ***dog*** *Rex of ...*

Post-nominal modifier. A post-nominal modifier can be realised in form of an *of*-construction, a finite relative clause or a named variable that starts with one of the four capital letter *X, Y, Z* or *W* (and is interpreted as a proper noun):

The dog Rex ***of Nic*** *...*
The dog Rex ***which is a labrador*** *...*
The dog ***X*** *...*

Verbal head. Only transitive verbs, the construction *has ... as ...* and the copula *be* are available in PENG-D. Furthermore, verbs can only be used in the simple present tense, the active voice, the indicative mood, and the third person singular:

Nic ***likes*** *Rex.*
Nic ***has*** *Rex as friend.*
Nic ***is*** *...*

Verbs are used for property assertions, apart from the copula *be* that can be used in constructions for class assertions and property assertions. Note that the possessive construction *Nic's dog Rex* and the *of*-construction *The dog Rex of Nic* result in the same translation as the sentence *Nic has Rex as dog*. The editor will displays a paraphrase for the possessive construction and for the *of*-construction.

Complement. The complement position can be realised by most of the syntactic structures that are approved for the subject position. Additionally, it allows for the prepositional construction *has ...* ***as*** *...* and for coordinated structures:

Nic is married ***to Sue****.*
Nic has ***Rex as dog****.*
Nic has ***Rex as dog and Tweety as bird****.*

There-sentences. A special case are "skolemized" *there*-sentences that have the following form:

There *is a dog Rex that is happy.*

4.3.2 Compound sentences

Constructors join simple sentences into complex sentences. The main constructors that are used on this level are: *if*, *iff*, *and* and *or*, for example:

If *X is a dog then X is an animal.*
If X is a man then X is a male ***and*** *X is an adult.*

Conditional (and biconditional) sentences are only available in the TBox for constructing ontologies. The nominal heads of such sentences can be realised by variables.

4.4 From PENG-D to HL and DL

We will now show how sentences of PENG-D are related to HL and DL statements, explain the function of these statements and explain which constraints apply. We do this by first looking at the RDFS statements that belong to the FOL

subset of RDFS and then by considering those OWL Lite statements that extend the expressivity of the FOL subset of RDFS but are still contained within DLP, that is, the intersection of HL and DL. Thereafter, we will discuss how constructors in PENG-D sentences are reflected in HL and DL and which consequences the use of these constructors has.

4.4.1 RDFS statements

There are only two types of ABox statements that belong to the FOL subset of RFDS: *class assertions* and *property assertions*. The TBox statements that belong to this subset are: *subclass*, *subproperty*, *range* and *domain* statements.

Class assertions. Classes let us express membership information about individuals:

CL: *Nic is a human and Rex is a dog.*
HL: human(nic). dog(rex).
DL: nic : Human ⊓ rex : Dog

Property assertions. Properties let us express specific facts about individuals:

CL: *Nic has the title Dr.*
HL: has_title(nic,doctor).
DL: <nic,doctor> : hasTitle

Subclass. Subclasses let us organise classes into a hierarchical taxonomy. Whenever the individual being an instance *X* of one class, this individual will necessarily be an instance of some other class:

CL: *If X is a labrador then X is a dog.*
HL: dog(X) ← labrador(X).
DL: Labrador ⊑ Dog

Subproperty. Properties like classes can be arranged in an hierarchy. We can declare a property as a subproperty (specialisation) of an existing property:

CL: *If X has Y as dog then X has Y as animal.*
HL: has_animal(X,Y) ← has_dog(X,Y).
DL: hasDog ⊑ hasAnimal

Domain of a property. We can restrict the domain of a property. In our example, the property *has as dog* has a domain of *human*. It relates instances of the class *human* to instances of *Y*:

CL: *If X has Y as dog then X is a human.*
HL: human(X) ← has_dog(X,Y).
DL: ⊤ ⊑ ∀hasDog$^-$.Human

Range of a property. Similarly, we can restrict the range of a property. In our example, the property *has as dog* has a range of *animal* and ties instances of class *animal* to instances of *X*:

CL: *If X has Y as dog then Y is an animal.*
HL: animal(Y) ← has_dog(X,Y).
DL: ⊤ ⊑ ∀hasDog.Animal

To restrict the domain and the range at the same time, we can write:

CL: *If X has Y as dog then X is a human and Y is an animal.*

4.4.2 OWL Lite statements

OWL Lite extends RDFS with additional TBox axioms. It adds explicit statements about class and property equivalence as well as the inverse of a property and transitivity.

Class equivalence. It is sometimes useful to indicate that a particular class in an ontology is equivalent to (i.e. has the same extension as) another class. In PENG-D, equivalence can be expressed via two conditional sentences but – as we will see below – this can be simplified using a biconditional operator:

CL: *If X is a man then X is a male adult.*
If X is a male adult then X is a man.
HL: male(X) ← man(X).
adult(X) ← man(X).
man(X) ← male(X), adult(X).
DL: Man ≡ Male ⊓ Adult

The biconditional operator *Iff* is a shorthand to express in one sentence that two (possible complex) class descriptions have precisely the same instances, for example:

CL: *Iff X is a man then X is a male adult.*

Property equivalence. Similar to class equivalence, we can express that a particular property is equivalent to another property:

CL: *Iff X has Y as price then X costs Y.*
HL: cost(X,Y) ← has_price(X,Y).
has_price(X,Y) ← cost(X,Y).
DL: cost ≡ hasPrice

Inverse of a property. If a property is the inverse of another property, then the variables in the first property switch their argument position in the second property, for example:

CL: *Iff X has Y as child then Y has X as parent.*
HL: has_parent(Y,X) ← has_child(X,Y).
has_child(X,Y) ← has_parent(Y,X).
DL: hasChild ≡ hasParent$^-$

Transitivity of a property. Transitivity is a property of a binary relation such that if *X* and *Y* are related, and *Y* and *Z* are related, then it follows that *X* and *Z* are also related, for all *X*,

Y, and *Z* for which the relation may apply. For example the property *ancestor of* is transitive:

CL: *If X is an ancestor of Y and Y is an ancestor of Z then X is an ancestor of Z.*
HL: ancestor(X,Z) ← ancestor(X,Y), ancestor(Y,Z).
DL: ancestor $^{+} \sqsubseteq$ ancestor

4.4.3 Constructors

The use of constructors is restricted in PENG-D, because they can not be expressed in HL or because they only be used in a restricted form in DL.

Conjunction. A conjunction of classes in the antecedent of a conditional sentence can be directly expressed in the body of a HL rule and creates no problem for DL:

CL: *If X is married and X is a woman then X is a wife.*
HL: wife(X) ← married(X), woman(X).
DL: Married ⊓ Woman ⊑ Wife

Conjunction in the consequent of a conditional sentence becomes a conjunction in the head of the corresponding HL rule, however this can be transformed into a pair of HL rules:

CL: *If X is a man then X is male and X is a person.*
HL: male(X) ← man(X).
person(X) ← man(X).
DL: Man ⊑ Male ⊓ Person

Disjunction. Disjunction of classes in the antecedent of a conditional sentence becomes a disjunction in the body of the corresponding HL rule. This again can be transformed into a pair of HL rules:

CL: *If X is a woman or X is a man then X is a human.*
HL: human(X) ← woman(X).
human(X) ← man(X).
DL: Woman ⊔ Man ⊑ Human

When a disjunction of classes occurs in the consequent of a conditional sentence, then it becomes a disjunction in the head of the HL rule, but this cannot be expressed within HL.

Universal restriction. In DL, the universal quantifier can only be used in restrictions of the form $\forall P.C$. Therefore, universal restriction can only be expressed in the following form in PENG-D:

CL: *If X is a women and X is married to Y then Y is a husband.*
HL: husband(X) ← woman(X), married_to(X,Y).
DL: Woman ⊑ ∀marriedTo.Husband

Expressing universal restriction of the form in the consequent of a conditional sentence would require negation in the rule body of HL.

Existential restriction. In DL, the existential quantifier can only be used in existential restrictions of the form $\exists P.C$. When an existential restriction occurs in the antecedent of a conditional sentence, it becomes a conjunction in the body of HL:

CL: *If X is married to Y and Y is a husband then X is a wife.*
HL: wife(X) ← married_to(X,Y), husband(Y).
DL: ∃marriedTo.Husband ⊑ Wife

However, if the existential restriction occurs in the consequent of a conditional sentence, then it becomes a conjunction in the head of corresponding HL rule with a variable that is existentially quantified. This cannot be handled in HL and would require transformation and skolemization in a logic program.

4.4.4 Beyond DLP

PENG-D is potentially a good starting point for language layering. More expressive language constructs could allow, for example, for expressing full existential quantification, instance equivalence, enumerating members of a class and cardinality constraints. While such descriptions cannot be directly expressed in DLP, many of them can be implemented in logic programing environments. Note that recursive HL rules such as transitivity need to be rewritten anyway for practical applications.

4.5 The inference engine

We are currently experimenting with various DL (and FOL) inference engines for question answering in PENG-D. Although not optimal, available FOL provers could provide reasoning services for more expressive DLs.

5 Conclusions

In this paper we referred to a number of deficiencies of RDFS as a "knowledge representation" language for the envisioned Semantic Web. Layering more complex ontology and rule languages on top of RDFS is not straightforward, because of its non-standard and non-fixed layer meta-modelling architecture. The relatively new DLP paradigm offers a promising first-order based alternative that enables ontological definitions to be combined with rules. To make such machine-processable information easily accessible for non-specialists, we proposed the use of PENG-D, a machine-oriented controlled natural language that has the same ex-

pressivity as DLP. We expect that PENG-D is easy to write for non-specialists with the help of a look-ahead text editor, easy to read in contrast to RDF-based notations, and easy to translated into a corresponding machine-processable format. In brief: PENG-D has the potential for complementing these more machine-oriented notations.

Acknowledgments

The research reported here is supported by the Australian Research Council (Discovery Project DP0449928). The authors would also like to thank two anonymous reviewers for the valuable comments on a previous version of this paper.

References

AECMA. 2004. The European Association of Aerospace Industries. *AECMA Simplified English*, AECMA Document PSC-85-16598. A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language. Issue 2, January 15.

D. Beckett (ed). 2004. RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-syntax-grammar/>.

T. Berners-Lee, J. Hendler, Ora Lassila. 2001. The Semantic Web. In: Scientific American. May 17.

T. Berners-Lee. 2001. Notation 3. An RDF language for the Semantic Web. <http://www.w3.org/DesignIssues/Notation3.html>.

T. Berners-Lee. 2003. Processing your data using N3 and CWM. <http://www.w3.org/2000/10/swap/doc/Processing>.

D. Brickley, R. V. Guha. 2004. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-schema/>.

N. E. Fuchs, U. Schwertel, and R. Schwitter. 1999. Attempto Controlled English - Not Just Another Logic Specification Language. Lecture Notes in Computer Science 1559, Springer, pp. 1-20.

B. N. Grosof, I. Horrocks, R. Volz, S. Decker. 2003. Description logic programs: Combining logic programs with description logic. In: Proceedings of the Twelfth International World Wide Web Conference (WWW 2003), pp. 48-57.

P. Hayes. 2004. RDF Semantics. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>.

I. Horrocks, P. F. Patel-Schneider. 2003. Three theses of representation in the semantic web. In: Proceedings of the Twelfth International World Wide Web Conference (WWW 2003), pp. 39-47.

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

F. Manola, E. Miller. 2004. RDF Primer. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-primer/>.

T. Mitamura. 1999. Controlled Language for Multilingual Machine Translation. Invited paper. In: Proceedings of Machine Translation Summit VII. Singapore, September 13-17. <http://www.lti.cs.cmu.edu/Research/Kant/>.

S. Palmer. 2002. A Rough Guide to Notation3. <http://infomesh.net/2002/notation3/>.

J. Z. Pan, I. Horrocks. 2003. RDFS(FA): A DL-ised Sub-language of RDFS. : In Proceedings of the 2003 Description Logic Workshop (DL 2003), volume 81 of CEUR, pp. 95-102.

R. Schwitter. 2002. English as a Formal Specification Language. In: Proceedings of the Thirdteenth International Workshop on Database and Expert Systems Applications (DEXA 2002). Aix-en-Provence, pp. 228–232.

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

R. Schwitter. 2004. Representing Knowledge in Controlled Natrual Language: A Case Study. In: M. G. Negoita, R. J. Howlett, L. C. Jain (eds.), Proceedings KES2004, Part I, Springer LNAI 3213, pp. 711-717.

M. K. Smith, C. Welty, D. L. Mc Guinness. 2004. OWL Web Ontology Language. Guide. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>.

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

Information Assembly for Adaptive Display

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

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

Abstract

In a world where information is increasingly delivered to users via different devices with dramatically different constraints and capabilities, it is becoming crucial to consider how the presentation of information must be adapted to suit specific devices and user contexts. To avoid confusing and disorienting the users as they switch between devices, the content and structure of information should be kept constant while its presentation must be optimised for each device to ensure usability. In this paper, we distinguish between two types of decisions that must be made during the presentation planning stage of an information delivery system: local decisions, which are based only on the content or features of the node itself, and global decisions, which are based on the entire structure of the discourse tree. We present a generic algorithm for making global decisions, driven by the discourse tree structure and contextual characteristics.

1 Introduction

In a world where information is increasingly delivered to users via different devices with dramatically different constraints and capabilities, it is becoming crucial to consider how the presentation of information must be adapted to suit specific devices and user contexts. To perform this adaptation and customisation manually is an expensive and time-consuming task. Additionally, as devices become increasingly interconnected, and as users are able to switch from one to another at will, it is essential to avoid confusing and disorienting the user (Chincholle, 2000). To this end, it is desirable to keep the content and structure of a document constant while optimising the presentation for each device to ensure usability. This is especially true for small or mobile devices.

Figure 1 illustrates this issue. The screen on the left shows a report that might have been dynamically generated for a PC based web browser in response to a query about "financial threats to the Russian space program". This report integrates information retrieved from multiple sources, and the information is structured and organised in order to be easily understandable. In particular, each piece of information is labelled with its role in the report. The most important part of the report is shown in bold at the very top. Then there is *elaboration* information, which gives more information about this first sentence. This is followed with information indicating the likely impact (*consequence*). The right hand side of Figure 1 shows the presentation of the same information, now delivered on a mobile phone. Given the different affordability of the delivery channel, the presentation and, in particular, the navigation is quite different, while the content and structure of the information being presented is constant.

Our approach to ensuring coherence and consistency of information across devices is to use Myriad (Paris *et al*., 2004), a platform for contextualised information retrieval and delivery based on theories and techniques from natural language generation. Our system produces virtual documents (or, more generally, presentations) by essentially going through two main stages: first it selects, retrieves and organises information to present to the user, taking the context into account (e.g., the task at hand, the user, the environment); then, taking the result of the first stage, it decides how to best deliver the selected information, once again taking the context into account, in particular the delivery device.

The remainder of this paper is structured as follows: we first present the Myriad architecture and briefly explain how it allows for coherent tailored information delivery. We introduce the two main stages involved: content and structure planning, and presentation planning. We then explain in detail the presentation planning stage, presenting how we came to decide that this stage was itself divided into two steps, which we present in turn, focusing especially on the "information assembly" step.

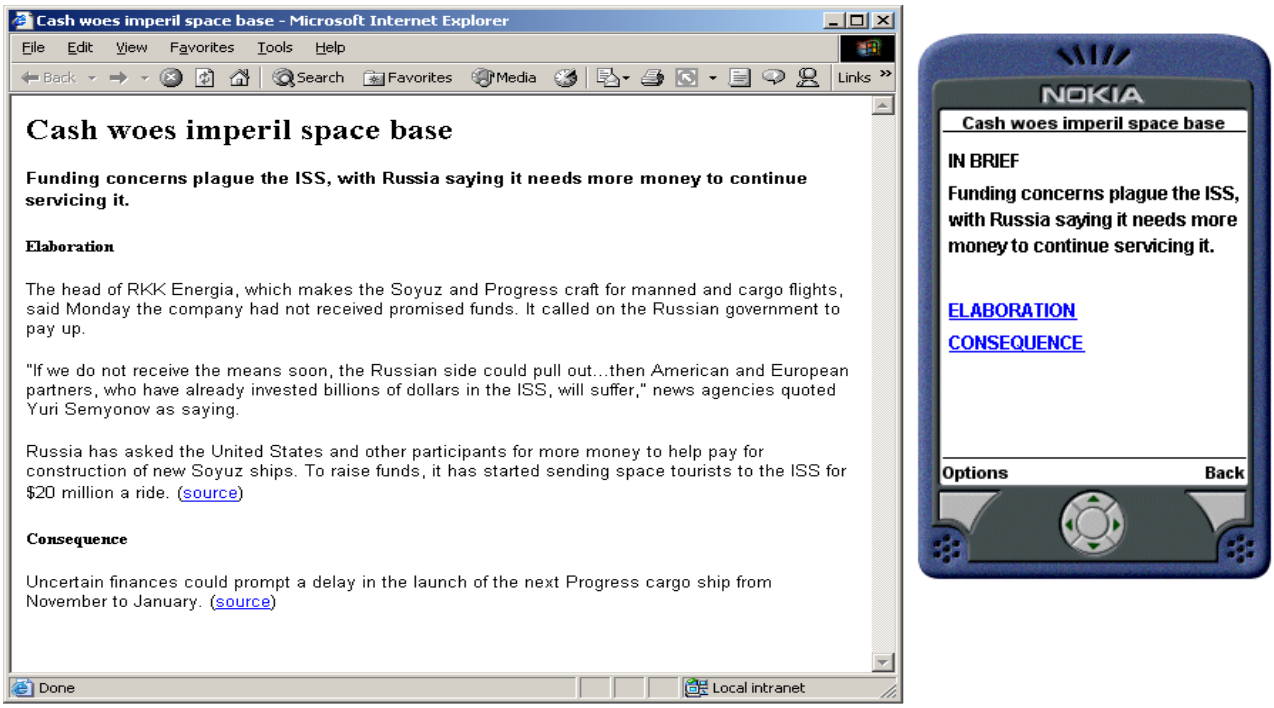


Figure 1: Adapting the presentation of information to suit different devices

Finally, we briefly introduce the last stage of the information delivery process, the realisation. The paper ends with some concluding remarks.

2 The Myriad Architecture

The core of the Myriad architecture is our information planning engine, which we call the Virtual Document Planner (VDP) (Colineau *et al.*, 2004b). The VDP is based on a typical Natural Language Generation (NLG) architecture, where the linguistic resources are separate from the planning engine. The VDP is based on the Moore and Paris (1993) text planner, and, as in that text planner, the resources are represented as plans.

The VDP works essentially as follows. Given a top level communicative goal (an overall purpose for presenting information), the engine uses a library of discourse rules (plan operators) to select and organise the content. Then, a library of presentation plans is employed to make local presentation decisions about how to present content. The output of the discourse planning stage is a discourse tree, which is then augmented (extended) during the presentation planning stage. This is shown schematically in Figure 2, where *content* represents the tree constructed as a result of the content and structure planning stage, and *presentation* represents the extension done during the presentation planning stage to take the specific delivery device into account. As we see from the figure, content and structure can thus remain constant across devices. Finally, the content represented in the extended tree is realised in syntax appropriate for the output device during the realisation phase. For example, referring back to Figure 1, during this phase, the content tree would be realised into HTML for the PC based display, while the mobile delivery might require WML.

As in (Moore and Paris, 1993), our approach exploits rhetorical relations, also called coherence relations, based on Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) to guarantee the coherence of the resulting presentation. The discourse and presentation rules both specify coherence relations that must hold between sibling sub-goals created by each goal decomposition. These coherence relations indicate how the various discourse segments and pieces of information work together to achieve the top level communicative goal. This was illustrated in Figure 1, where the second paragraph was related to the first paragraph by an *elaboration* relation, while the third paragraph was related to it by a *consequence* relation. Using terminology from RST, the first paragraph is the *nucleus*, while the other two paragraphs are *satellites*.

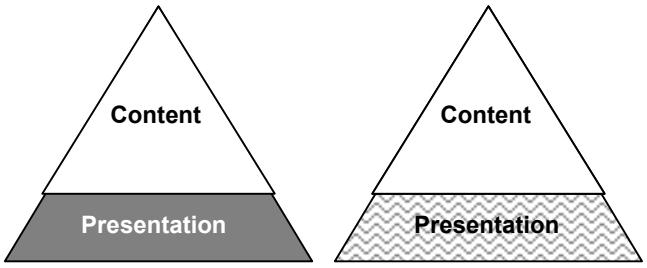


Figure 2: Content and presentation planning for two different devices

For reasons of simplicity and modularity, we explicitly maintain a conceptual and architectural separation between the different processes of content planning, presentation planning and surface realisation. This separation is closely analogous to the distinction between document planning, micro planning and surface realisation adopted by many natural language generation systems (e.g., McKeown, 1985; Hovy 1988; Moore and Paris, 1993).

We exploited this generation paradigm to build several prototypes of information delivery systems, in particular one in the travel domain (cf., Wilkinson *et al.,* 2000; Paris *et al.*, 2001; Paris, 2002) and one in the corporate domain, where users received a brochure about CSIRO tailored to their interest and needs (cf. Paris *et al.*, 2003). In these prototypes, the process was exactly as illustrated in Figure 2.

Through these prototypes, however, we have now come to recognise that, during the presentation planning stage, we must distinguish between two types of decisions:

- *local* decisions, which are based only on the content or features of the *node* itself, and
- *global* decisions based on the *entire structure* of the discourse tree.

An example of a local decision might be to 'present the content of this node in a table' or to present that content as a bulleted list. In contrast, deciding to create a navigation index and organising content pages appropriately around this index requires more global knowledge of the discourse tree structure.

Thus, the process we illustrated in Figure 2 can really only be used to make local decisions. Failing to make the distinction between these two types of decisions resulted in discourse operators that embedded decisions that should really occur during the presentation stage, while taking the whole discourse tree into account. This thus blurred the separation between content and presentation planning that we originally desired.

To address these shortcomings, we have designed a new layer within the presentation planning stage. This layer implements the reasoning that must occur during presentation planning but that must make decisions based on the global discourse structure. Our aim in the work presented here was four fold:

- keep the conceptual separation between content and structure planning and presentation planning, while still being able to make a number of decisions that affect the final presentation (in particular, decisions about navigation);
- explicitly decouple, in the presentation planning, local decisions from global decision;
- decouple the decision process (or algorithm) from the information needed to make decisions. This information might include characteristics of the device or of the user; and, finally,
- produce a generic set of operators which can be used to produce reasonable (but not necessarily highly optimised) output for a large range of information domains, discourse structures and output devices. This is in contrast with our other work, including the DFDMSA[1] project, where our aim is to produce multimedia output that is optimised for the task at hand (Colineau *et al.*, 2004).

In the remainder of this paper, after briefly describing the process implementing the local presentation decisions, we focus on the processing stage which enables the system to make global presentation decisions based on the discourse tree.

3 Information Presentation

We now propose to account for the two types of decisions we introduced in the previous section by having two steps, which we term "local presentation planning" and "information assembly". Each of these steps exploits its own mechanism. We describe them in turn. As mentioned above, when the system reaches the presentation planning stage, a discourse tree has already been constructed through discourse planning.

3.1 Local Presentation Decisions

Local presentation planning is as we had originally envisioned and implemented it. This stage thus extends the current discourse tree. It is performed using standard Moore and Paris (1993) inspired plan operators.

[1] Data Fusion, Data Management and Situation Assessment.

```
<operator>
      <effect>InitialiseAssembly ?nodeId</effect>
      <constraint>(discourse:isRootNode(?nodeId))</constraint>
      <constraint>(set ?user (user:getCurrentUser()))</constraint>
      <constraint>(set ?deviceModel (device:getCurrentDevice(?user)))</constraint>
      <constraint>(set ?Layout (device:chooseLayout(?deviceModel)))</constraint>
      <constraint>
      (set ?contentFrame (layout:chooseContentFrame(?Layout ?nodeId)) )
      </constraint>
      <operation>
      (discourse:annotateFeature(contentPaneSpace device:getSpace(?contentFrame))
      </operation>
      <operation>(discourse:annotateFeature(contentFrame ?contentFrame))</operation>
      <operation>(discourse:annotateFeature(Layout ?Layout))</operation>
      <operation>(post ProcessRootNode ?Layout ?contentFrame)</operation>
</operator>
<operator>
      <effect>ProcessRootNode ?Layout ?contentFrame</effect>
      <constraint>(layout:hasIndexFrame(?Layout))</constraint>
      <constraint>(layout:isRefillable(?contentFrame))</constraint>
      <constraint>(set ?indexFrame (layout:getIndexFrame(?Layout)))</constraint>
      <constraint>(set ?depthLimit (layout:getDepthLimit ?indexFrame))</constraint>
      <constraint>(set ?realisationOrder 1)</constraint>
      <operation>(discourse:annotateFeature(indexDepth 0))</operation>
      <operation>(discourse:annotateFeature(indexFrame ?indexFrame))</operation>
      <operation>(discourse:annotateFeature(indexDepthLimit ?depthLimit))</operation>

      <operation>(foreach ?child (discourse:getChildrenInOrder(?nodeId))
         (post     ProcessNode     ?child     ?Layout     (layout:chooseContentPage(?child,
      ?contentFrame))
       ?realisationOrder++ ?indexFrame))
      </operation>
</operator>
```

Figure 3: Assembly operators for the root node of the discourse tree

As explained earlier, this stage augments the discourse tree at the leaf nodes by performing further goal decomposition. This stage uses a specific library of presentation plan operators. Decisions on how to present specific discourse content (contained in the leaf nodes of the discourse tree) are made during this step. Examples of presentation decisions made could include marking up titles with specific style and font information, or choosing to display certain information as a list. Throughout this process, presentation decisions are made based on node-local information; the global structure of discourse tree is not considered. In certain circumstances, presentation planning may reason about relations between immediate siblings, but not about more global tree structure. For example, if a node is related to a sibling nucleus by an RST relation "preparation", a decision might be made to mark up the content of a node with specific style and font information appropriate for a heading. Previous work in natural language generation has shown that rhetorical text organisation, as embodied in the RST relations in a discourse tree, can be used to motivate text-formatting decisions (e.g., Hovy and Arens, 1991).

3.2 Global Presentation Decisions

The second stage deals with presentation decisions which require knowledge of the global structure of the discourse tree. This stage primarily *assembles* and *combines* pieces of information based on how they are related to each other within the discourse tree and on the constraints of the environment (e.g., the delivery device). We thus choose to refer to this stage as the i*nformation assembly* stage.

In this stage, the tree constructed thus far is traversed in a top-down fashion, as opposed to being extended. Each node is annotated with specifications that will be interpreted during the final realisation tree-walk. (Note that it is only during the final realisation pass through the discourse tree that the final output is actually produced.) This approach can be seen as an extension of the realisation process used in the DFDMSA project (Colineau *et al*. 2004).

For the sake of simplicity, information assembly is performed using another specific library of operators which implement the following algorithm.

Starting at the root node of the discourse tree, the VDP performs a top down pass through the tree, executing the following two functions at each node:

1. It annotates the current tree node with information required for correct realisation. These annotations are based on contextual characteristics. As mentioned before, the annotations are interpreted during final realisation.
2. It posts a new information assembly goal for each of its child nodes (if any). In this way, the discourse tree is traversed in a recursive manner, using the same sub goal decomposition mechanism of the existing plan operators and planning engine.

As we process the root node of the tree, a number of contextual models are accessed. In particular, information about the device used by the user is retrieved. This determines how to proceed. We have identified a number of attributes that may characterise the device. They include:

- the size of the screen (if there is a screen);
- styles of navigation;
- available modalities;
- software capabilities; and
- layout templates available for formatting on that specific device (e.g., stylesheets).

This stage of the information assembly is implemented by the operators shown in Figure 3. As illustrated in these operators, the contextual models are accessed through the *constraints* mechanism, allowing the system to choose appropriate operators based on these models. Note also the *operations* of the operators. It is through them that (1) the tree is augmented with annotations and (2) the system posts additional goals to continue the traversal of the tree.

Importantly, these operators are generic, and the decision process is decoupled from the information needed to make decisions, as was our goal. This information is encoded in declarative models outside the engine. These include the device model.

It is envisaged that the device model would eventually be encoded in a standard format such as that defined in the W3C's Composite Capabilities/Preferences Profile (CC/PP) framework (Klyne *et al*., 2004). The RDF-based CC/PP framework provides a way to describe generic profiles which are accessible via the web, and could be created and maintained by hardware and software vendors for their own devices and applications. This mechanism would work well as a distributed contextual device model for the Myriad framework. Because the CC/PP framework has not yet been widely adopted, for now we are using a simpler XML-based representation for our device models.

Layout templates act as stylesheets to organize the presentation of information. Layout templates are not device specific, meaning that the same layout template can often be used across multiple devices. Equally, the same device may support a number of different layout templates for displaying content. Layout templates are characterised by properties such as:

- the set of frames contained;
- the content type, e.g., MIME content types (Freed and Borenstein, 1996), supported by each frame; and
- the relative size of each frame.

Once instantiated for a specific device, a layout template and its frames are additionally constrained by the capabilities of the device. For example, a device may determine the amount of content that can be displayed in each frame of a layout template, whether a frame can spawn new windows for displaying new content, and whether the frame supports scrollbars. All these features affect later decisions about how to allocate space to individual pieces of content, and how to display all the planned content if there is not enough space. Assembly operators reason about layout template characteristics to determine how the planned content will be presented in that template.

An example of a frame is an index frame. This contains a hierarchically indented set of links to content displayed in other frames. Index frames have additional attributes such as the maximum hierarchy depth supported for index entries.

If the chosen layout template includes an index frame, it will contain hyperlinked references to some of the content pages. Exactly which pages are able to add a reference to themselves in the index frame is controlled by the assembly operators. The maximum hierarchy depth of an index frame signifies the *indent depth limit* of the index. The assembly operators keep track of the number of content page references already inserted. At some point, a depth threshold will be reached (which represents a maximum level of indentation), and nodes will no longer be able to insert references into the index. At this point, all content for the discourse tree branch below that node must be delivered into a single, linear space (for most devices, this represents a single page instance). In this situation, an algorithm for allocating space is needed to divide up that linear space amongst the child nodes. Two possible algorithms are discussed below.

After initialisation is complete, the information assembly stage recursively processes the remaining nodes in the tree. This results in further annotations being made to the tree. Examples of features that may be annotated in each node include:

- The order in which a node should be realised in relation to its sibling nodes.

Importantly, this is determined by the discourse relations between each of the sibling nodes. For example, Figure 4 illustrates that, for the elaboration RST relation, the nucleus is realised before the satellite;

- Hyperlink anchor text;
- Whether a node should add a reference to itself in the navigation index (if there is one); and
- The exact page instance(s) to be used to realise the node's content.

Table 1 illustrates some specific examples of feature names and possible values, exactly as they are encoded in the discourse tree.

Feature Name	Example Value	Description/ Interpretation
realisationOrder	2	Realise this node second, after one sibling node.
addToIndex	True	Add a reference to this node's content in the navigation index frame
indexDepth	2	The depth of index entries at this node is 2.
indexText	Section Name	Use this text as the anchor in the index to link to this node's content.
anchorText	More Information	Use this text as the anchor text for this hypertext link.
linkTarget	Page001	The target of this hypertext link is the specified page.
contentPage	Page001	Realise the content of this node in the specified page.

Table 1: Examples of features that are annotated into the discourse tree during information assembly

The coherence relations are ranked in another declarative resource, so that authors can declare whatever subjective realisation ordering suits their specific purposes or information domain.

```
<relation>
  <name>elaboration</name>
  <type>RST</type>
  <description>The elaboration RST
        relation</description>
  <library>discourse</library>
  <importance>medium</importance>
  <order>N:S</order>
</relation>
```

Figure 4: Definition for the elaboration RST relation

Space Allocation within a single page

A layout template may not have a separate navigation index frame, as is often the case on small screen devices such as mobile phones or PDAs. Alternatively, the system may have descended through the discourse tree beyond the index depth threshold. At that point, there is a single linear piece of space in which to realise an entire segment of our presentation. Often, in such a situation, there will not be enough space to realise all the content directly. As a result, sections of content must be summarised or realised on separate pages that are hyperlinked into the single shared space.

We have identified two generic heuristics that could be used to allocate content to the available space. (This assumes we can characterise available space, e.g., by the number of lines of text that can be displayed). These heuristics are embodied in additional operators.

One heuristic allocates space to sibling nodes in a manner proportional to their importance in the text, as indicated by the discourse relations in the tree. Let's consider a node with 3 children: one nucleus, and two satellites, a preparation and an elaboration. The nucleus is allocated half the available space, and the two satellites a quarter each, assuming preparation and elaboration have been declared with equal importance. At each level of recursion, the available space is further divided. This means that after a certain recursion depth, the algorithm reverts to producing a list of links to each node's content (similar to a page of search results).

Another heuristic allocates most of the available space to the nucleus, and provides hypertext links to each of the satellites. This is the algorithm used to produce the small-screen output in Figure 1.

These two heuristics provide generic mechanisms for allocating content to space. It is also always possible to define operators very specific to a domain or a device.

As a final remark on the information assembly stage, as the tree is not extended during this stage, strictly speaking, there is thus no further planning involved. The operators used are not plan operators proper. However, for the sake of convenience, we have chosen to express these assembly operators using a very similar syntax to our plan operators, and, like plan operators, they are declarative rules used by the VDP engine. This allows us to reuse our existing planning engine and plan processing code to perform the information assembly stage.

4 Realisation

As noted, the annotations created during information assembly are interpreted during the final realisation tree walk. This makes the realisation process a relatively simple annotation interpretation process, which allows us to have a very generic realisation module.

As an example, when processing the children of a node, the realisation module simply needs to refer to the *realisationOrder* feature annotation in each node to know the correct order in which to process the children. Similarly, the presence of *addToIndex* annotations instruct the realisation module to add a reference to a particular node into the index, and hyperlink *anchorText* and *linkTarget* annotations specify how to create hyperlinks according to the decisions made during information assembly.

The realisation process results in the content actually being placed in specific page instances which are then displayed in specified frames of the chosen layout template. These pages are generated in a device-specific syntax, such as HTML or WML.

5 Implementation Status

This work is being carried out in the context of our work on contextualised information retrieval and delivery. The Myriad framework has been implemented and is being exploited in a number of domains and applications – e.g., Tiddler (Wilkinson *et al.*, 2000; Paris *et al.*, 2001), PERCY (Paris *et al.*, 2003), DFDMSA (Colineau and Paris, 2003, Colineau *et al.,* 2004a) and skil (Müller-Tomfelde *et al.*, 2004). The framework continues to be enhanced and extended.

The work presented here has recently been added into the framework and is currently being tested in an application requiring the automatic production of tailored reports.

Parts of the framework and the approach have been evaluated (e.g., Paris *et al.*, 2001; Paris *et al.*, 2003, Wilkinson and Wu, 2004; Wu *et al.*, 2004). We intend to perform further evaluations.

6 Conclusion

In our work, we are concerned with delivering information consistently and coherently across heterogeneous devices. To do so, we propose an approach which allows a presentation to be planned once, and yet be delivered appropriately on the delivery medium of choice.

In this paper, we distinguished between two types of decisions that must be made during the presentation planning stage of an information delivery system: *local* decisions, which are based only on the content or features of the *node* itself, and *global* decisions, which are based on the *entire structure* of the discourse tree.

We presented a generic algorithm for making global decisions, driven by the discourse tree structure and contextual characteristics. This algorithm, implemented through declarative operators, complements the more traditional local presentation decisions made through presentation planning.

7 Acknowledgements

We wish to thank Keith Vander Linden for his ideas embodied in the first space allocation algorithm described as part of this work, and for his work on the Myriad framework. We also thank other members of the Information Engagement group at the CSIRO ICT Centre who have helped us implement the algorithm, in particular MingFang Wu and Akshay Bhurtun.

References

Chincholle, D. (2000). Designing effective mobile services on small communication devices. Tips and Techniques. Tutorial notes in *Proceedings. of OzCHI 2000: Interfacing Reality in the New Millennium*. December 4-8, 2000. Sydney.

Colineau, N. and Paris, C. (2003). Task-Driven Information Presentation. *In Proceedings of the 2003 Annual Conference of the Computer-Human Interaction (OZCHI'03)*, the Special Interest Group of the Human Factors and Ergonomics Society of Australia, Brisbane, Australia, Nov 25-28, 2003.

Colineau, N., Lampert, A. and Paris, C. (2004a). Task-Sensitive User Interfaces: grounding information provision within the context of the

user's activity. In *Proceedings of the working conference on Advanced Visual Interfaces (AVI) Conference*, Gallipoli, Italy, May 25-28 2004, 218-225.

Colineau, N., Paris, C. and Wu, M. (2004b). Actionable Information Delivery. In *Revue d'Intelligence Artificielle* (RSTI – RIA), Special Issue on Tailored Information Delivery, 18(4), 549-576. September 2004.

Freed, N and Borenstein, N. (1996). Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types – Published as "Request For Comments: 2046".

Hovy, E. (1988). Planning coherent multisentential text. In *Proceedings. of the 26th Conference of the ACL*, Buffalo, NY, 163-169.

Hovy, E., and Arens, Y., (1991). Automatic generation of formatted text. In *Proceedings of the 8th Conference of the American Association for Artificial Intelligence.* 92 – 96. Anaheim, CA: AAAI.

Klyne,G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. and Tran, L. (2004). *Composite Capabilites/Preference Profiles (CC/PP): Structure and Vocabularies 1.0*. W3C Recommendation 15 January 2004.

Mann, W. and Thompson S. (1988). *Rhetorical Structure Theory: Towards a Functional Theory of Text Organization.* In *Text*, 8(3), 243-281, 1988.

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

Moore, J. and Paris, C. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. In Computational Linguistics, 19(4), 651-694.

Müller-Tomfelde, C, Paris, C. and Stevenson, D. (2004). Interactive Landmarks: Linking Virtual Environments with Knowledge-Based Systems. To appear in the *Proceedings of the 2004 Annual Conference of the Computer-Human Interaction (OZCHI'04)*, the Special Interest Group of the Human Factors and Ergonomics Society of Australia, Wollongong , Australia, November 22-24, 2004.

Paris, C. (2002). Information Delivery for Tourism. In *IEEE Intelligent Systems* Intelligent Systems for Tourism: Trends and Controversies, Stephen Staab and Hannes Werthner (eds), November/December, 2002. pp 53 – 66, 2002.

Paris, C., Wan, S., Wilkinson, R. and Wu, M. (2001). Generating Personal Travel Guides – and who wants them? In *Proceedings of the International Conference on User Modelling (UM2001);* M. Bauer, P. Gmytrasiewicz and J. Vassileva (eds). Sonthofen, Germany, July 13-18, 2001. 251-253.

Paris C, Wu M, Vander Linden K, Post M, and Lu S. (2004). Myriad: An Architecture for Contextualized Information Retrieval and Delivery (2004). In *AH2004: International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*. August 23-26 2004, The Netherlands. pp.205-214.

Paris, C., Wu, M., Vercoustre, A-M., Wan, S., Wilkins, P. and Wilkinson, R. (2003). An Empirical Study of the Effect of Coherent and Tailored Document Delivery as an Interface to Organizational Websites. In *Proceedings of the Adaptive Hypermedia Workshop at the 2003 User Modelling Conference*, Pittsburgh, USA, June 22, 2003. 133-144.

Wilkinson, R., Lu, S., Paradis, F., Paris, C., Wan, S. and Wu, M. (2000). Generating Personal Travel Guides from Discourse Plans. In Proceedings of International Conference on Adaptive Hypermedia and Adaptive Web-based Systems. Trento, Italy, August 2000.

Wilkinson, R and Wu, M. (2004). Evaluation Experiments and Experience from the Perspective of Interactive Information Retrieval, In the *Proceedings of the Third Workshop on Empirical Evaluation of Adaptive Systems*, in conjunction with AH2004. August 23-26, The Netherlands. pp.216-225.

Wu, M., Muresan, G., McLean, A. Tang, M., Wilkinson, R. ,Li, Y., Lee, H. and Belkin, N. (2004). Human versus Machine in the Topic Distillation Task, In the *Proceedings of the 27th SIGIR Conference on Research and Development in Information Retrieval*, July 25th-29th 2004, Sheffield, UK. pp. 385-392.

Referring Expression Generation as a Search Problem

Bernd Bohnet
Bohnet@iis.uni-stuttgart.de
Institute for Intelligent Systems
University of Stuttgart
70569 Stuttgart
Germany

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

Abstract

One of the most widely explored issues in natural language generation is the generation of referring expressions (GRE): given an entity we want to refer to, how do we work out the content of a referring expression that uniquely identifies the intended referent? Over the last 15 years, a number of authors have proposed a wide range of algorithms for addressing different aspects of this problem, but the different approaches taken have made it very difficult to compare and contrast the algorithms provided in any meaningful way. In this paper, we propose a characterisation of the problem of referring expression generation as a search problem; this allows us to recast existing algorithms in a way that makes their similarities and differences clear.

1 Introduction

A major component task in natural language generation (NLG) is the generation of referring expressions: given an entity that we want to refer to, how do we determine the content of a referring expression that uniquely identifies that intended referent? Since at least (Dale, 1989), the standard conception of this task in the literature has been as follows:

1. We assume we have a knowledge base that characterises the entities in the domain in terms of a set of attributes and the values that the entities have for these attributes; so, for example, our knowledge base might represent the fact that entity e_1 has the value *cup* for the attribute *type*, and the value *red* for the attribute *colour*.

2. In a typical context where we want to refer to some e_i, which we call the *intended referent*, there will be other entities from which the intended referent must be distinguished; these are generally referred to as *distractors*. So, for example, we may want to distinguish a particular cup from all the other items present in the context of a dining table.

3. The goal of referring expression generation is therefore to find some collection of attributes and their values which distinguish the intended referent from all the potential distractors in the context.

Over the last 15 years, a wide variety of algorithms have been proposed to deal with specific aspects of this problem. For example, while earlier algorithms focussed on the use of attributes that correspond to simple one-place predicates, later work attempts to address the use of relational predicates, and other work looks at the incorporation of boolean operators such as *not* and *and*. The consequence is that we now have a considerable body of research in this area, but it is difficult to establish just how these different algorithms relate to each other.

This paper represents a first step towards consolidating the results in this area, with the aim of developing a framework within which different algorithms can be compared and assessed. The structure of the paper is as follows. In Section 2, we provide a brief overview of work on the generation of referring expressions to date. In Section 3, we borrow a standard approach used in Artificial Intelligence (AI) to represent problems in an elegant and uniform way (see, for example, (Simon and Newell, 1963); (Russell and Norvig, 2003)), sketching how GRE algorithms can be expressed in terms of *problem-solving by search*. In Section 4, we explore how the most well-known algorithms can be expressed in this framework. In Section 5, we discuss how this approach enables a more fruitful comparison of existing algorithms, and we point to ways of taking this work further.

2 A Brief Review of Work To Date

Although the task of referring expression generation is discussed informally in earlier work on NLG (in particular, see (Winograd, 1972; McDonald, 1980; Appelt, 1981), the first formally explicit algorithm was introduced in Dale (1989). This algorithm, which we will refer to as the Full Brevity (FB) algorithm, is still frequently used as a basis for other GRE algorithms. The FB algorithm searches for the best solution amongst all possible referring expressions for an entity; the algorithm derives the smallest set of attributes for the referent in question, producing a referring expression that is both *adequate* and *efficient*.

This initial algorithm limited its application to one-place predicates. Dale and Haddock (1991) introduced a constraint-based procedure that could generate referring expressions involving relations (henceforth IR), using a greedy heuristic to guide the search.

As a response to the computational complexity of greedy algorithms, (Reiter and Dale, 1992; Dale and Reiter, 1995) introduced the psycholinguistically motivated Incremental Algorithm (IA). The most used and adapted algorithm, this is based on the observation that people often produce referring expressions which are informationally redundant; the algorithm uses a preference ordering over the attributes to be used in a referring expression, accumulating those attributes which rule out at least one potential distractor.

In recent years there have been a number of important extensions to the IA. The Context-Sensitive extension (CS; (Krahmer and Theune, 2002)) is able to generate referring expressions for the most salient entity in a context; the Boolean Expressions algorithm (BE; (van Deemter, 2002)) is able to derive expressions containing boolean operators, as in *the cup that does not have a handle*; and the Sets algorithm (SET; (van Deemter, 2002)) extends the basic approach to references to sets, as in *the red cups*.

Some approaches combine algorithms which reuse only parts of other algorithms: the Branch and Bound (BaB; (Krahmer et al., 2003)) algorithm uses the Full Brevity algorithm, but is able to generate referring expressions with both attributes and relational descriptions using a graph-based technique.

We have identified here what we believe to be the most cited strands of research in this area, but of course there are many other algorithms described in the literature: see, for example, (Horacek, 1997; Bateman, 1999; Stone, 2000). Space limitations prevent a complete summary of this other work here, but our intention is to extend the analysis presented in this paper to as many of these other algorithms as possible.

All these algorithms focus on the generation of definite references; they are typically embedded in a higher-level algorithm that includes cases for when the entity has not been previously mentioned (thus leading to an initial indefinite reference) or when the referent is in focus (thus leading to a pronominal reference); see, for example, (Dale, 1989; Krahmer and Theune, 2002; Dale, 2003).

3 GRE from the Perspective of Problem Solving

With so many algorithms to choose from, it would be useful to have a uniform framework in which to discuss and compare algorithms; unfortunately, this is rather difficult given the variety of different approaches that have been taken to the problem.

Within the wider context of AI, Russell and Norvig (2003) present an elegant definition of a general algorithm for problem solving by search. The search graph consists of *nodes* with the components *state* and *path-cost*; the problem is represented by an *initial-state*, an *expand-method* which identifies new states in the search space, a *queuing-method* which determines the order in which the states should be considered, and a *path-cost-function* which determines the cost of reaching a given state. In this framework, the search strategy is determined by the combination of *queuing-method* and *path-cost-function* used.

In the following, we use this framework to provide a characterisation of existing GRE algorithms in terms of problem solving by search. We conceptualise the search space as consisting of states that have three components: a description that is true of the intended referent, the set of distractor entities that the description also applies to besides the intended referent, and the set of properties of the intended referent that have not yet been considered to describe the referent.

1. The *initial-state* is of the form $\langle\{\}, C, P\rangle$, where C is the set of distractors in the initial context, and P is the set of all properties true of the intended referent.

2. The goal state is of the form $\langle\{\lambda x P_1, \lambda x P_2, \ldots\}, \{\}, P'\rangle$, where the first term contains a set of properties of the intended referent that, by virtue of the second term (the set of distractors) being an empty set, distinguish the intended referent; P' contains any properties of the intended referent not yet used in the description.

3. All other states in the search space are then intermediate states through which an algorithm will move as it adds new properties to the description.

4. The search strategy is carried out by the *expand-method* and the *queuing-method*, which together characterise the specific GRE algorithm (for example, FB, GH or IA) that is used.

5. The *path-cost-function* allows us to route the search as required; this can be used to take account of salience weights, or to embody some kind of heuristic search.

For any given algorithm, not all of the methods and functions need to be implemented; in particular, some algorithms do not require a *path-cost-function*.

4 GRE Algorithms in Terms of Problem Solving

We adopt here an object oriented formalism,[1] since this allows the representation of dependencies between the algorithms by means of inheritance and overwriting.

To enable more fruitful comparison of the different GRE algorithms, we want to distinguish those aspects of the algorithms which are true of all algorithms, and those which are unique to each particular algorithm. In Section 4.1, we first describe the elements that are shared by all the algorithms; we then go on to describe the distinct aspects of each algorithm in turn.

[1]We follow the code conventions as used in OO-languages, where the names of classes start with upper case characters, and the names of methods and variables start with lower case characters.

4.1 Common Elements

This approach allows us to separate out those aspects of the various algorithms which remain constant.

Following from the previous section, the definitions of the *node* and *state* classes are as shown in Definition 1. This figure also shows the definitions for *initial-state* and *goal*, which remain constant across the algorithms.

Definition 1: The Node and State Classes

```
class Node {
  s // State
  path-cost // Cost of the path
  getState()
   {return s} // returns the state of the node
}
class State {
  L // Set of chosen properties and/or relations
  C // Set of distractors
  P // Set of available properties and/or relations
}
initialState() {return new State(∅,C,P)}
// the goal is the empty set of distractors
goal(s) {
   if s.C = ∅ then return true
   else return false
}
```

Given these components, the main method *makeRefExp* is then as represented in Definition 2. This takes two arguments, which serve as the parameters that distinguish one algorithm from another: an *expand* method to create the successors of a given state, and a *queue* method, which defines how to insert nodes into the node queue. Depending on the order in which the nodes are inserted, different search strategies can be realized: for example, when the nodes are inserted at the front of the queue, the search strategy is depth-first; when the nodes are inserted at the end of the queue, the search strategy is breadth-first; when the nodes of the queue are sorted by the estimated distance to the goal, then the search type is best-first; and so on.

In addition, we may require a number of general-purpose methods which can be used by a number of different algorithms. One such method is the method *rulesOut*, which takes a property or relations p and a set of distractors, and returns the set of distractors which are ruled out by p.

Definition 2: The Basic Algorithm Structure

```
makeRefExp() {
  // create a initial queue with a single node
  nodeQueue ← [new Node(initialState())]
  while nodeQueue ≠ ∅ do
    node ← removeFront(nodeQueue)
    if goal(node.getState()) then
      return node // success
    end
    nodeQueue ← queue(nodeQueue,expand(node))
  end
  return nil // failure
}
```

With this machinery in place, we can now redefine the existing algorithms in terms of their core differences, which correspond essentially to different ways of expanding the search space.

4.2 The Full Brevity Algorithm

The distinctive property of the Full Brevity (FB) algorithm is that it computes all combinations of the available properties P with increasing length, so that it may find the shortest combination that succeeds in identifying the intended referent.

This behaviour is captured by the *expand* method shown in Definition 3. The method creates a set of successors by creating a node for each property p_i which has not so far been checked, provided that p_i rules out at least one distractor.

The FB algorithm uses a breadth-first search implementation of the queue, as shown in Definition 4. Consequently, any solution for which *goal* returns *true* will have a minimal number of properties, since the breadth-first search considers smaller combinations of properties first.

The FB algorithm uses the *expand* method, and *createNode* method which are shown in Definition 3 and it is invoked by a call of *makeRefExp* method which is shown in Definition 2.

4.3 The Incremental Algorithm

The distinctive property of the Incremental Algorithm is that it reduces the computational complexity of constructing a referring expression by considering properties to use in sequence from a predefined ordering of the available properties. The implementation of the *expand* method shown in Definition 5 provides this behaviour.

Definition 3: The Full Brevity Algorithm

```
expand(node) {
  N ← ∅
  s ← node.getState()
  foreach p ∈ s.P do
    N ← N ∪ { createNode (node, p)}
  end
  return N
}
createNode(node, p) {
  s ← node.getState()
  out ← rulesOut(p, s.C)
  if out ≠ ∅ then
    return new Node(s.C − out, s.L ∪ {p},
                    s.P − {p})
  else return new Node(s.C, s.L, s.P − {p})
}
```

Definition 4: Breadth-first Queueing

```
queue(actNodes, newNodes) {
  // append the nodes at the end
  return actNodes ∪ newNodes
}
```

If the set of properties of the current state $s.P$ is not empty, then the first property p according to the given order O is chosen from the set of properties of the current state $s.P$, and a node is created with a new state by the method *createNode*. Note that the *createNode* method is the same as that used in the FB algorithm and shown in Definition 3.

Unlike the *expand* method used in the FB algorithm, however, the set of nodes returned here contains only one node. The main method applies the *goal* predicate to this node; if this returns true, then the node containing the state with the list of properties for the referring expression is returned.

4.4 Extension of the IA to Sets

All the algorithms considered so far have been concerned with constructing descriptions for individual referents; van Deemter (2002) introduced an algorithm which extends the IA to sets. The extension is shown in terms of our framework in Definition 6.

Definition 5: The Incremental Algorithm

```
O // Predefined constant order of properties
expand(node) {
  N ← ∅
  s ← node.getState()
  if s.P ≠ ∅ then
    p ← choose the first p in O, where p ∈ s.P
    N ← N ∪ { createNode(node, p) }
  end
  return N
}
```

Definition 6: The Set Algorithm

```
R // Set of referents
createNode(node, p) {
  out ← rulesOut(p, s.C)
  if (¬∃x ∈ R&x ∈ out)&(∃x ∈ C&x ∈ out) then
    return new Node(s.C − out, s.L∪{p},s.P − {p})
  else return new Node(s.C, s.L, s.P − {p})
}
```

Note that, precisely because this algorithm is an extension of the IA algorithm, we reuse the *expand* method from that algorithm. Consequently, the extension requires only the rewriting of the *createNode* method, whereby an attribute p_i is only chosen when it does not rule out entities from the set of referents R and when it rules out at least one entity from the set of distractors C. If a property does not fulfil that condition, then a node with the current state is returned and the process is continued, as in the IA, with the next property.

4.5 GRE Involving Relations

The algorithm for GRE Involving Relations (IR) introduced by Dale and Haddock (1991) is constraint-based. The search strategy used to fulfil the constraints is a combination of a greedy search, which chooses the relation that leads to the smallest set of distractors, and depth-first search to describe the entities, that is, the intended referent as well as entities which are referenced in the relations.

The strategy can be explained best by means of an AND/OR-tree, as shown in Figure 1. Here, the top node represents a state in which relational properties are to be considered as additions to the set of chosen properties. Each search step consists of two stages: in the first stage, we choose the relation p_i which rules out the largest number of distractors; in the second stage, each entity which is referenced by the chosen relation has to be described by repeating the process recursively. This is done in a depth-first manner, but if the related entity is not uniquely distinguished then the next p_j that the intended referent participates in is chosen, and so on. This process continues until all entities are uniquely described (*success*) or no further relations can be chosen (*failure*).

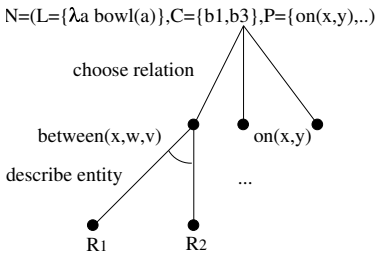


Figure 1: Expansion tree for the IR algorithm

The algorithm is represented in the problem solving paradigm as in Definition 7. Here, the *expand* method chooses a relation which rules out the largest number of distractors; it then calls the method *createNode*, which recursively calls *makeRefExp* for each new referent contained in the relation.

4.6 Context-Sensitive GRE

Krahmer and Theune (2002) also introduced a number of extensions to the IA: the use of salience weights in order to add a definite article to the description for the most salient entity; contrastive properties in order to add properties which impose a contrast between two entities; and a relational extension, similar in spirit but not in form to that in the IR algorithm described above.

Again, as for the *sets* algorithm, the commonality with the IA algorithm surfaces as the reuse of the latter algorithm's *expand* method; only the *createNode* method needs to be rewritten, as in Definition 8. To model this variant in our framework, we introduce the following additional methods (cf. (Krahmer and Theune, 2002)):

- *contrastive* takes a referent r and a property p_i; it checks whether the property under consideration is contrastive.
- *mostSalient* takes a referent r, a set of prop-

Definition 7: Involving Relations

```
r // Referent
expand(node) {
  s ← node.getState()
  p_c ← nil
  // Chosen relation is p_c, where p rules out
  // the largest number of distractors
  foreach p ∈ s.P do
    if p_c = nil or
      |rulesOut(p, s.C)| > |rulesOut(p_c, s.C)|
    then p_c ← p
    end
  end
  node_c ← createNode(node, p_c)
  if (node_c = nil) then return ∅
  else return {node_c}
}
createNode(node, p) {
  s ← node.getState()
  C ← rulesOut(p, s.C)
  L ← s.L ∪ {p}
  // Extend Description
  foreach r' ∈ {r_p | r_p ∈ referents(p) & r_p ≠ r} do
    node_r' ← makeRefExp(r')
    if node_r' = nil then return nil // failure
    L ← L ∪ node.getState().L
  end
  return new Node(C, L, s.P − {p})
}
```

Definition 8: Context-Sensitive Algorithm

```
r // Referent
createNode(node, p) {
  s ← node.getState()
  out ← rulesOut(p, s.C)
  C ← s.C − out
  L ← s.L
  if (out ≠ ∅ or contrastive(r, p)) then
    L ← L ∪ {p}
    if v expresses a relation between r and r' then
      node_r' ← makeRefExp(r')
      L ← L ∪ node_r'.getState().L
    end
  end
  if mostSalient(r, L, C) then
    L ← L ∪ { defArt }
    // The most salient rules out all distractors:
    C ← ∅
  end
  return new Node(C, L, s.P − {p})
}
```

erties, and a set of distractors; it checks whether every entity in the set of distractors has a lower salience weight than r.

5 Conclusions and Future Work

In the foregoing, we have shown how a number of the most frequently discussed algorithms for the generation of referring expressions can be represented within a common framework. There are three significant advantages to this approach.

First, it allows us to determine what the algorithms have in common. This is particularly interesting in that it allows us to begin to assemble a collection of core functionalities that are usable in a variety of different approaches to GRE. This is apparent not only in terms of the general framework (where, for example, the notions of states and their initialisation, and definition of what it is to be a goal state, and the overall algorithmic pattern) are shared, but in terms of 'helper' routines (such as *rulesOut* and *mostSalient*) which can be modularised out of the essence of different algorithms.

Second, it makes it possible to see what the differences between the algorithms really consist in. In their original forms, these differences are obscured, due to the absence of a common vocabulary for expressing the algorithms; by representing the algorithms within a common framework, it becomes easier to see where the algorithms differ, and where the differences are simply due to differences in notation or presentation. By using the framework of problem solving as search, we have effectively decomposed the algorithms into a number of key elements: a search srategy, represented by the *queuing-method*, and an *expand-method*, which encompasses two aspects of each algorithm: the basic strategy adopted and the particular kinds of referring expressions covered. Furthermore, the *expand-method* decomposes into a general strategy for expansion (as found in, for example, the Full Brevity algorithm and the Incremental Algorithm), and a *createNode* method, which varies depending upon the kind of referring expression targetted.

Third, it allows us to see more clearly the logical space within which the algorithms reside, and to see ways of combining aspects of different algorithms. At its simplest, this is clearest with respect to the kind of search strategy used in the algorithms. Present formulations conflate the choice of search strategy with the other aspects of the algorithm (such as how subsequent nodes in the search space are com-

puted); our approach separates out these different facets of the algorithms, and makes it much easier to see that the choice of search strategy is an independent decision. Consequently, for example, we can easily experiment with a variant of the IR algorithm that uses breadth-first search rather than depth-first search.

So far, we have used the framwework to express the most widely-known algorithms in the literature. Preliminary examination of the algorithms in (Krahmer et al., 2003), (van Deemter and Krahmer, forthcoming), and (Horacek, 2004) suggests that these will also be relatively straightforward to express within the framework described here. As we capture more algorithms in the framework, our intention is to tease out an inventory of basic constituent elements which can then be reassembled and integrated in different ways, so that we can derive a better understanding of the nature of the problem of referring expression generation.

References

D. E. Appelt. 1981. *Planning Natural Language Utterances to Satisfy Multiple Goals*. Ph.D. thesis, Stanford University.

J. Bateman. 1999. Using Aggregation for Selecting Content when generating Referring Expressions. In *Proceedings of the 37th Annual Meeting of the ACL*. University of Marylad.

R. Dale and N. Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of the 5th EACL*, pages 161–166, Berlin, Germany.

R. Dale and E. Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

R. Dale. 1989. Cooking up referring expressions. In *Proceedings of the Twenty-Seventh Annual Meeting of the ACL*, pages 68–75, Vancouver, British Columbia.

R Dale. 2003. One-anaphora and the case for discourse-driven referring expression generation. In *Proceedings of the Australasian Language Technology Workshop*. University of Melbourne.

H. Horacek. 1997. An Algorithm for Generating Referential Descriptions with Flexible Interfaces. In *Proceedings of the 35th Annual Meeting of the ACL*. University of Madrid.

H. Horacek. 2004. On Referring toSets of Objects Naturally. In H. Bunt and R. Muskens, editors, *Third International Natural Language Generation Conference*, pages 70 – 79. Springer-Verlag Heidelberg.

E. Krahmer and M. Theune. 2002. Efficient context-sensitive generation of referring expressions. In *Information Sharing: Reference and Presupposition in NLG and Interpretation*, pages 223–264. CSLI.

E. Krahmer, S. van Erk, and A. Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

D. D. McDonald. 1980. *Natural Language Generation as a Process of Decision-making Under Constraints*. Ph.D. thesis, Massachusetts Institute of Technology.

E. Reiter and R. Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th ACL*, pages 232–238.

S. Russell and P. Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition.

H. A. Simon and A. Newell. 1963. GPS, a program that simulates human thought. *Computers and Thought*, pages 279–293.

M. Stone. 2000. On identifying sets. In *Proceedings of the First International Natural Language Generation Conference*. Mitzpe Ramon.

K. van Deemter and E. Krahmer. forthcoming. Graphs and booleans: On the generation of referring expressions. In H. Bunt and R. Muskens, editors, *Computing Meaning Vol. III*. Dordrecht: Kluwer.

K. van Deemter. 2002. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

A Meta-grammar for CCG

Mark FOREMAN
Dept. of Electrical and Electronic Eng.
The University of Adelaide
Adelaide SA 5005
Mark.Foreman@csiro.au

Daniel McMICHAEL
Information Enhancement Group
CSIRO ICT Centre
Waite Road, Urrbrae SA 5064
Daniel.McMichael@csiro.au

Abstract

Applying CCG to domains outside of linguistics could require different sets of combinators to be developed for each domain. The meta-grammar described in this paper aims to assist such development by enabling simple, succinct expression of both existing and new combinator definitions. It favours the development of an easily-configurable, one-time-coded module that can perform CCG combinations for any combinator set of the researcher's choosing. A preliminary implementation shows both the feasibility and potential of the meta-grammar.

1 Introduction

The merits of Combinatory Categorial Grammar (CCG) have been established via natural language parser implementations like that of Hockenmaier and Steedman (2002) and Clark and Curran (2004). But recent findings show that categorial grammars based on CCG also display promise in domains outside of linguistics (McMichael et al., 2004). Over the years, new combinators have been developed to extend the system of pure categorial grammar (Steedman 2003), but although the set of combinators for CCG seems to have stabilised, this same set may not necessarily be applicable to analyses in other domains. In fact, McMichael et al. introduce two combinators – *functional application* and *modificational application*, both defined in section 2.1 – that are not part of the existing set of CCG combinators. Additionally, functional application cannot even be cleanly defined via a traditional combinator pattern. It is partly this inability of the existing techniques to cleanly define new combinators that motivates the proposal of a meta-grammar for combinator specification.

In this paper, we explain the motivations for and give a specification of the meta-grammar, along with complete examples of how it applies to new and existing combinators. Lastly, we examine the workings and potential of a preliminary implementation. The remainder of this section, however, presents a brief introduction to CCG.

1.1 A Practical Introduction to CCG

CCG operates by first assigning a syntactic category to each word in the sentence, as will be demonstrated in the proceeding example borrowed from Hockenmaier (2003). At this point, the notation for describing categories should be observed. Assuming the simplistic subject-verb-object (SVO) pattern for English, the phrase "buys shares" will form a complete sentence S if it is preceded by a noun phrase, and we write this as:

```
buys shares ├ S\NP
```

So the phrase "buys shares" can be thought of as a function that takes a noun phrase NP as an argument to its left and returns a sentence S.

Furthermore, "buys" will form a S\NP if it is followed by a noun phrase, and this is denoted as:

```
buys ├ (S\NP)/NP
```

In doing this, we have eliminated the need for a separate verb category V, leaving us with the following category assignments:

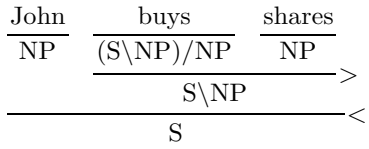
```
John ├ NP
buys ├ (S\NP)/NP
shares ├ NP
```

Formally, a category may be either atomic (S, NP, etc) or complex (S\S, (S\NP)/NP), etc). Complex categories take the general form α/β or α\β, where α and β are themselves categories.

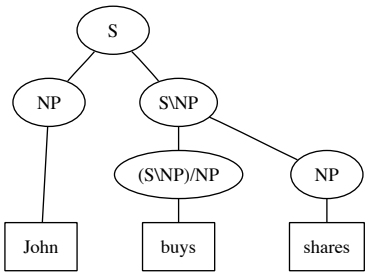
Given the above category assignments, a derivation proceeds as follows: "buys" is combined with "shares" under the operation of *forward application* (the term *forward* referring to both the direction of the slash). The phrase "buys shares" is combined with "John" under the operation of *backward application*. The combinators (operators) that govern these two operations are defined as follows:

```
X/Y   Y   ⇒>   X
Y   X\Y   ⇒<   X
```

where X and Y represent any category. Typically, a derivation is represented in the following manner:



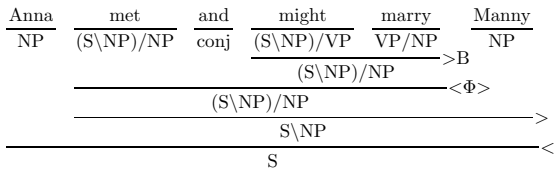
which corresponds to the following tree:

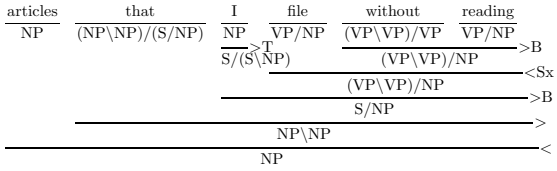


Several other combinators are defined by Steedman (2000) for capturing long-range dependencies in the English and Dutch languages: coordination (Φ) type-raising (T), composition (B) and substitution (S). These combinator families are listed below:

```
X   conj   X      ⇒<Φ>    X
X                 ⇒>T     Y/(Y\X)
X                 ⇒<T     Y\(Y/X)
X/Y   Y/Z         ⇒>B     X/Z
X/Y   Y\Z         ⇒>Bx    X\Z
Y\Z   X\Y         ⇒<B     X\Z
Y/Z   X\Y         ⇒<Bx    X/Z
(X/Y)/Z   Y/Z     ⇒>S     X/Z
(X/Y)\Z   Y\Z     ⇒>Sx    X\Z
Y\Z   (X\Y)\Z     ⇒<S     X\Z
Y/Z   (X\Y)/Z     ⇒<Sx    X/Z
```

The usage of some of these combinators is shown below, using example derivations taken from Steedman (2000) and Hockenmaier (2003).





For further reference on CCG, the reader is directed to Steedman (1996) and (2000).

1.2 An Historical Note on CCG

The slash notation seen in categories of CCG stems from that used in the early works on pure categorial grammar by Ajdukiewicz (1935), Bar-Hillel (1953) and Lambek (1958). Steedman (1993) explains that he and Dowty refined these earlier notations, leading to the more consistent and readable style that is described in section 1.1. The only rules permitted by pure categorial grammar are forward and backward application. CCG extends this system with the above-listed rules, based on Curry and Feys' *combinators* – a term coined in their 1958 work on combinatory logic, where they examined devices that operate on functions, irrespective of the number of arguments. The combinatory nature of CCG rules enables a transparent mapping between syntactic and semantic form, thus providing one of the major appeals of this grammar formalism.

2 Motivations

As can be seen in section 1.1, enumerating the entire set of combinators can be lengthy. Given that combinators in a family like {>B, >Bx, <B, <Bx} differ only by the direction of the slashes and order of the operands, it seems wasteful to present each one explicitly. A more compact representation is afforded by specifying only the pattern for >B, along with the transformations required to obtain the variations >Bx, <B and <Bx. The proposed meta-grammar provides a method for succinctly specifying these variations.

Without recognising the similarity within combinator families, and even between combinator families, writing code to apply combinators can be laborious, error-prone and wasteful, unless these similarities are exploited for optimum code reuse. The implementation of this meta-grammar takes full advantage of intra- and inter-family similarities.

2.1 New Domains

Providing a meta-grammar by which to specify combinator families lends itself to a single-module implementation that can be easily configured to handle new combinators. This capability is important because, although the set of combinators for use in linguistics seems to have matured, there

are other domains that stand to benefit from CCG analyses, but for which grammar development is still in its infancy. In particular, the authors are currently developing a generic parser capable of being configured to specific domains, including, but not limited to, NLP and situation assessment. Research into applying CCG to these domains is being assisted by an ability to perform rapid prototyping on their grammars.

The two new combinators mentioned in the introduction – functional application (F) and modificational application (M) – are defined as follows:

$$X/Y \quad Y \Rightarrow_{>F} X$$
$$Y \quad X\backslash Y \Rightarrow_{<F} X$$
$$X/X \quad X \Rightarrow_{>M} X$$
$$X \quad X\backslash X \Rightarrow_{<M} X$$

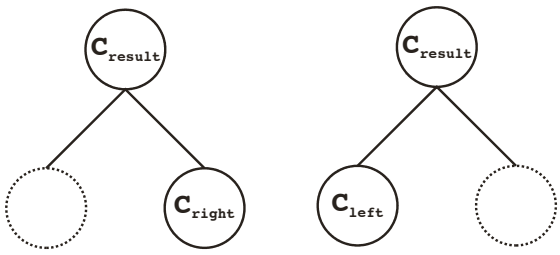
with the caveat that X≠Y. These combinators find use in the authors' research in both English parsing and situation assessment. Note that >F and >M together cover all combinations possible under the traditional vanilla forward application rule >; similarly for <F and <M with <. It will suffice to say here that the reason for splitting the traditional rules into two was to correctly handle head inheritance while maintaining a simple mathematical model; further explanation is not within the scope of this paper[1]. The >F and <F combinators defined above would incorrectly be interpreted as the vanilla > and < combinators if it were not for the caveat. So this example not only demonstrates new combinators, but also highlights the shortcomings of the current methods for specifying combinators. The proposed meta-grammar provides an elegant (caveat-free) solution to this problem.

Another domain that is planned for investigation is geology. The general intent is to analyse vertical sequences of discrete sedimentary layers in a manner analogous to English parsing, where sequences of (discrete) words are analysed. Griffiths (1989) founded the precursor to this research by demonstrating that meaningful analyses could be performed on sedimentary sequences using a context-free grammar. He also speculated that context-sensitive analyses might be able to resolve some ambiguities, lending weight to the application of CCG, which is mildly context-sensitive, to such a task.

[1] The authors would like to credit the work of Geoff Jarrad in developing these two combinators.

2.2 Converting Treebanks

When employing a statistical parser, a suitable corpus of pre-parsed sentences is required for training the probabilities. However, altering the grammar through the addition or deletion of combinators (as is done when applying CCG to new domains) requires a new corpus to be marked-up accordingly. This process typically requires converting the context-free derivation trees from the Penn Treebank (Marcus et al., 1993, 1994) to intermediate binary context-free trees and then finally to CCG trees. The second stage of conversion (binary CF to CCG) requires a technique referred to as *inverse combination*, where an unknown left or right category is determined given the result category. This contrasts to regular combination, where the unknown result is deduced from two known operands. There are two types of inverse combination: *missing-left* (when the right operand and result are known) and *missing-right* (when the left operand and result are known. Missing-left and missing-right scenarios are shown left and right respectively below.



It will be shown in section 4.2 that a standard implementation of the meta-grammar can be made to perform these operations by merely permuting some of the configuration information passed to the module.

2.3 Python Implementation

The implementation described in this paper was coded in Python (Python Programming Language, 2004). Python was chosen for its ability to aid rapid prototyping and for its ease of integration with much faster C code. Thus we hope to benefit from lower coding times and easier debugging, with the option to port to C and re-integrate any mature code that is deemed time-critical.

3 The Meta-grammar

This section details the meta-grammar that controls the specification of a CCG. The set of combinators are specified as a list of combinator templates, one template for each combinator family:

```
COMBINATOR-SET :=
```

COMBINATOR-TEMPLATE$_1$
…
…
COMBINATOR-TEMPLATE$_m$

Each combinator template defined separately, as well as any atomic variations referenced in the templates. These are both described below, and proceeded by some example templates.

It is worth noting that this proposal focuses primarily on specifying combinators for the express purpose of performing combinations. The corresponding semantics (logical forms) may be associated with operands and the result, following from Steedman (2000).

3.1 Specifying a Combinator Template

A combinator template is specified as a tuple:

```
COMBINATOR-TEMPLATE :=
  ( TYPE,
    OPERAND-PATTERN-LIST,
    RESULT-PATTERN,
    PERMITTED-VARIATIONS )
```

where the entries in the tuple are defined as:

TYPE: an identifier for the combinator that should be unique across all other combinator templates. Typically it is a single character; in section 1.1 we saw them as Φ, T, B and S.

OPERAND-PATTERN-LIST: the ordered list of n operand patterns. Typically `n=2` since most combinators are binary operations, although `n=1` for type-raising (T). The syntax for these patterns is given in section 3.2.

RESULT-PATTERN: a pattern that specifies how to construct the resulting category from operand categories that successfully match the operand patterns.

PERMITTED-VARIATIONS: as mentioned in section 2, only one pattern set is specified per family of combinators. Each variation in the family is specified as a tersely coded entry in this list. For instance, the composition family (B) would have permitted-variations = `{ >, >x, <, <x }`.

3.2 Specifying a Pattern

The patterns specified in the combinator template must conform to the following EBNF syntax:

```
<PATTERN> := <ATOMIC>|
   <COMPOUND>
<ATOMIC> := <A>['e'|'n']<N>
<A> := ('A'|…|'Z')+
<N> := ('0'|…|'9')+
<COMPOUND> := <LEFT>
   (<RIGHT>|'['<RIGHT>']')
<LEFT> := <ATOMIC>|
   '('<COMPOUND>')'
<RIGHT> := <SLASH><LEFT>
<SLASH> := ('\'|'/')<N>
```

For simplicity of expression, we introduce the semantic requirement that an `<A>n<N>` pattern may only occur immediately after a slash. Alternatively, we could provide a completely context-free grammar for patterns through a slightly less intuitive EBNF, by redefining `<ATOMIC>` and `<RIGHT>`:

```
<ATOMIC> := <A>['e']<N>
<RIGHT> := <SLASH>
   (<LEFT>|<A>'n'<N>)
```

Some patterns that conform to this syntax include:

X_1
$Y_1/_2Y_3$
$(X_1/_1Y_1)\backslash_2X_2$

Atomic patterns (patterns without slashes or brackets) are specified as alphanumeric strings which to allows for greater control over pattern specification. Any two atomic patterns (`A'N'` and `A"N"`) and the categories they match (`C'` and `C"` respectively) are governed by the following constraints:

$A'=A'', N'=N'' \Rightarrow C'=C''$
$A'=A'', N'\neq N'' \Rightarrow C'\neq C''$

As an example, suppose we want to match some category to the pattern $((X_1/_1X_1)/_2X_2)/_3Y_1$, then the subcategory in the position of the first X_1 must be equal to the subcategory in the position of the second X_1, but must be *distinct* from the subcategory in the position of the X_2 (and any other $X_{<N>}$ that might have been in the pattern). The subcategories in the positions of X_1, X_1 and X_2 are *independent* of the subcategory in the position of Y_1. For example, this pattern would match the categories `((A/A)/B)/A`, `((A/A)/B)/B` and `((A/A)/B)/C`, but not `((A/C)/B)/A` or `((A/A)/A)/A`.

The presence of an 'e' in an atomic pattern indicates that the atomic pattern will only match with an atomic category. Thus X^e_1 will match category `A`, but not `A/A`.

The presence of an 'n' in an atomic pattern indicates that the atomic pattern will allow matching to an unlimited number of arguments, similar to the "$ convention" described in

(Steedman 2000). A pattern $X_1/_1Y^n_1$ would match categories A/B, (A/B)/C, ((A/B)/C)/D, etc.

Square brackets (if present) in a pattern surround an optional portion of that pattern. For example, the pattern $X^e_1[/_1Y^e_1]$ would match categories A and A/B, but not (A/B)/C or A/(B/C).

3.3 Specifying Variations

Each combinator in a given family corresponds to exactly one variation in the permitted-variations list of that family's combinator-template. Suppose we have operand and result patterns:

```
TYPE = B
OPERANDS = X1/1Y1, Y1/2Z1
RESULT   = X1/3Z1
```

then a > in the permitted-variations list corresponds to the combinator:

$$X_1/Y_1\ Y_1/Z_1 \Rightarrow_{>B} X_1/Z_1$$

That is, forward combination > does not alter slash directions or operand order. On the other hand, backward combination < reverses all slashes and operand order, so a < in the permitted-variations list would correspond to the combinator:

$$Y_1\backslash Z_1\ X_1\backslash Y_1 \Rightarrow_{<B} X_1\backslash Z_1$$

Other atomic variations may be defined and used with either < or >. An atomic variation that is used in generating the composition (B) family is:

$$x{:}\{/_2,/_3\}$$

That is, the x variant reverses the direction of slash 2 and slash 3. The effect of atomic variations is successive. So a variation like <x would have operands and all slashes reversed by <, but the x would reverse slashes 2 and 3 back to their original orientation (in this case, forward):

$$Y_1/Z_1\ X_1\backslash Y_1 \Rightarrow_{<Bx} X_1/Z_1$$

Taking this one step further, suppose we invent an arbitrary variant $i{:}\{/_3\}$, then the combinator corresponding to variant <xi would be:

$$Y_1/Z_1\ X_1\backslash Y_1 \Rightarrow_{<Bxi} X_1\backslash Z_1$$

Slash 3 has been reversed once by <, again by x and again by i, giving an overall effect of a single reversal.

3.4 Some Example Templates

This section is a simple demonstration of how the above-described meta-grammar can be used to specify both existing (type-raising, composition) and new (functional application, modificational application) combinators.

3.4.1 Type Raising

```
TYPE = T
OPERANDS = X1
RESULT = Y1/1(Y1\2X1)
VARIATIONS = {>,<}
```

$$X_1 \Rightarrow_{>T} Y_1/(Y_1\backslash X_1)$$
$$X_1 \Rightarrow_{<T} Y_1\backslash(Y_1/X_1)$$

3.4.2 Composition

Note that this template specifies general composition (B^n).

```
TYPE = B
OPERANDS = X1/1Y1, Y1/2Z^n1
RESULT = X1/3Z^n1
VARIATIONS = {>,>x,<,<x}
```

$$X_1/Y_1\ Y_1/Z^n_1 \Rightarrow_{>B} X_1/Z^n_1$$
$$X_1/Y_1\ Y_1\backslash Z^n_1 \Rightarrow_{>Bx} X_1\backslash Z^n_1$$
$$Y_1\backslash Z^n_1\ X_1\backslash Y_1 \Rightarrow_{<B} X_1\backslash Z^n_1$$
$$Y_1/Z^n_1\ X_1\backslash Y_1 \Rightarrow_{<Bx} X_1/Z^n_1$$

3.4.3 Functional Application

```
TYPE = F
OPERANDS = X1/1X2, X2
RESULT = X1
VARIATIONS = {>,<}
```

$$X_1/X_2\ X_2 \Rightarrow_{>F} X_1$$
$$X_2\ X_1\backslash X_2 \Rightarrow_{>F} X_1$$

3.4.4 Modificational Application

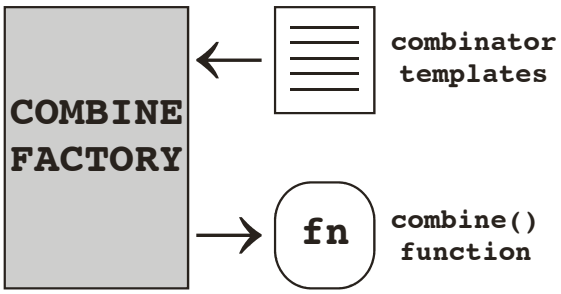
```
TYPE = M
OPERANDS = X1/1X1, X1
RESULT = X1
VARIATIONS = {>,<}
```

$$X_1/X_1\ X_1 \Rightarrow_{>F} X_1$$
$$X_1\ X_1\backslash X_1 \Rightarrow_{>F} X_1$$

4 Using the Implementation

A prototype module has been developed in Python to implement the meta-grammar described in this paper. The module can be thought of as a factory which takes a combinator-set conforming to the

definition in section 3 and returns a single function, `combine()`.



The `combine()` function takes any number of operand categories as arguments and returns a list of (result-category, combinator) tuples that corresponds to all possible categories that can be derived from the input categories.

4.1 Combination

As an example, let us consider the module when configured by the type-raising and composition combinator templates given in sections 3.1 and 3.2. The input to the `combine()` function is a sequence of operand categories, and the output is a list of possible resulting categories and their corresponding combinators.

Suppose the input is a pair of categories, `A/B` and `B\C`. Type-raising is immediately discounted by the function because it is unary and thus cannot operate on a pair of categories. Consequently, the function only considers the composition (B) combinators. The function attempts to match the first category, `A/B`, with the first operand pattern, $X_1/_1Y_1$, and the second category, `B\C`, with the second operand pattern, $Y_1/_2Z^n_1$, ignoring slash directions for the moment. This match is successful, and results in a *match dictionary* of $\{X_1{:}A,\ Y_1{:}B,\ Z^n_1{:}C\}$. The slashes are then found to match those required for forward crossing (`>x`) composition $\{s_1{:}/,\ s_2{:}\backslash\}$, but not for vanilla forward composition $\{s_1{:}/,\ s_2{:}/\}$. From these matches, the result can be built: $X_1\backslash Z^n_1{:}A\backslash C$. To attempt the backwards combinations, the function then tries to match the input categories to the reversed sequence of operand patterns, i.e. `A/B` with $Y_1/_2Z^n_1$ and `B\C` with $X_1/_1Y_1$. This attempt fails because $Y_1{=}A$ in the first category, while $Y_1{=}C$ in the second category. So for the input `A/B B\C`, the output is a single category-combinator pair: `(A\C >Bx)`.

If the input were `A/B C`, the output would be an empty list since the second category `C` will not match the structure of either of the operand patterns.

Suppose now the input is a single category `A`. The composition combinators can be immediately discounted since they require two operands. However, it does match the single operand pattern for the type raising combinator, giving match dictionary $\{X_1{:}A\}$. This conforms to both the forward and backward type-raising, so the function would output a pair list `{(*/(*\A) >T), (*\(*/A) <T)}`, where the `*` character indicates that there was no match for Y_1 in the input. The handling of these wildcards rests with the client software.

4.2 Inverse Combination

A very useful property of the meta-grammar and its associated implementation is that it can be configured to deduce a child category given the derived category and the other child/children. A process we term *inverse combination*.

Consider the case of a binary combinator with pattern:

```
OPERANDS = A B
RESULT   =     C
VARIATIONS = {>*,<*}
```

where `A`, `B`, `C` are pattern *placeholders* (– they are obviously not valid patterns themselves), `>*` represents some number of forward variations and `<*` represents some number of backward variations. Now suppose we know the left and result categories (c_A and c_C), and wish to enumerate all valid right categories (c_B) – the *missing-right* scenario. This is achieved via a two-step process, involving the instantiation of two `combine()` functions:

```
OPERANDS = A C
RESULT   =     B
VARIATIONS = {>*}
    ↓
[COMBINE FACTORY]
    ↓
combine1()

OPERANDS = C A
RESULT   =     B
VARIATIONS = {<*}
    ↓
[COMBINE FACTORY]
    ↓
combine2()
```

Simple addition of the two returned lists obtains the desired result:

$$combine_1(c_A,c_C) + combine_2(c_A,c_C)$$

This works because $\texttt{combine}_1\texttt{()}$ returns the list of c_B's that result from valid matches of $\texttt{A:}c_A$, $\texttt{C:}c_C$. $\texttt{combine}_2\texttt{()}$ also matches $\texttt{A:}c_A$, $\texttt{C:}c_C$, but its operation is a little less obvious: because $\texttt{combine}_2\texttt{()}$ only considers backward combinations, it always reverses the order of its operands c_A and c_C, so A is still compared with c_A, and C with c_C.

Inverse combination for the *missing-left* scenario can be performed similarly, so in the interest of brevity its detail is omitted.

While the above approach may seem awkward, keep in mind that no changes are required to the meta-grammar definition or to the implementation's code base. So inverse combination is obtained for free.

5 Conclusion

We have defined a meta-grammar for specifying complete families of CCG combinators. This meta-grammar covers existing combinators, but more importantly, it provides a guide for specifying and using new combinators. A brief look at a preliminary implementation reveals that the meta-grammar is indeed practical, and lends itself to powerful exploitation.

6 Acknowledgements

This paper is based on work supported by Boeing and CSIRO, and by a Commonwealth scholarship (APA) plus Woodside PhD top-up scholarship to the first author. The authors would like to thank Geoff Jarrad for his input to discussions and assistance with reviewing. Many thanks also to the official reviewers for their helpful suggestions.

References

(2004). Python Programming Language, Python Software Foundation. 2004.

Ajdukiewicz, K. (1935). "Die syntaktische Konnexität." *Studia Philosophica* 1: 1-27.

Bar-Hillel, Y. (1953). "A Quasi-Arithmetical Notation for Syntactic Description." *Language* 29: 47-58.

Clark, S. and J. Curran (2004). Parsing the WSJ using CCG and Log-Linear Models. *42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain.

Curry, H. and R. Feys (1958). Combinatory logic. Amsterdam, North-Holland.

Griffiths, C. M. (1989). "The nature of the geological representation language and consequent constraints on machine interpretation." *Advances in Geophysical Data Processing* 3: 49-77.

Hockenmaier, J. (2003). Data and Models for Statistical Parsing with Combinatory Categorial Grammar. School of Informatics. Edinburgh, University of Edinburgh: 280.

Hockenmaier, J. and M. Steedman (2002). Generative Models for Statistical Parsing with Combinatory Categorial Grammar. *40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia.

Lambek, J. (1958). "The mathematics of sentence structure." *American Mathematical Monthly* 65: 154-170.

Marcus, M. P., G. Kim, et al. (1994). The Penn treebank: Annotating predicate argument structure. *Human Language Technology Workshop*.

Marcus, M. P., B. Santorini, et al. (1993). "Building a Large Annotated Corpus of English: The Penn Treebank." *Computational Linguistics* 19(2): 313-330.

McMichael, D., G. Jarrad, et al. (2004). Modelling, Simulation and Estimation of Situation Histories. *7th International Conference on Information Fusion (Fusion 2004)*, Stockholm, Sweden, International Society for Information Fusion.

Steedman, M. (1993). "Categorial grammar." *Lingua* 90(3): 221-258.

Steedman, M. (1996). Surface Structure and Interpretation. Massachusetts, MIT Press.

Steedman, M. (2000). The Syntactic Process. Massachusetts, MIT Press.

Steedman, M. and J. Baldridge (2003). Combinatory Categorial Grammar (Draft 4.0, August 10, 2003). ftp://ftp.cogsci.ed.ac.uk/pub/steedman/ccg/manifesto.pdf [Internet]. Accessed 15 September 2004.

Intelligent Multi Media Presentation of information in a semi-immersive Command and Control environment

Cécile Paris, Nathalie Colineau
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
ICT Centre
Sydney, NSW, Australia
Cecile.Paris@csiro.au
Nathalie.Colineau@csiro.au

Dominique Estival
Defence Science & Technology Organisation (DSTO)
Human Systems Integration Group
Command and Control Division
Edinburgh SA 5111, Australia
Dominique.Estival@dsto.defence.gov.au

Abstract

We describe the framework for an intelligent multimedia presentation system we designed to be part of the FOCAL laboratory, a semi-immersive environment for Command and Control Environment. FOCAL comprises a number of input devices and output media, animated virtual conversational characters, a spoken dialogue system, and sophisticated visual displays. These need to be coordinated to provide a useful and effective presentation to the user. In this paper, we describe the principles which underlie intelligent multimedia presentation (IMMP) systems and the design of such a system within the FOCAL multi-agent architecture.

1 Introduction

1.1 Description of FOCAL

FOCAL (Future Operations Centre Analysis Laboratory) was established at the Australian Defence Science and Technology Organisation (DSTO) to "*pioneer a paradigm shift in command environments through a superior use of capability and greater situation awareness*". The facility was designed to experiment with innovative technologies to support this goal, and it has now been running since 2000.

FOCAL contains a large-screen, semi-immersive virtual reality environment, where large quantities of information can be displayed. A number of modalities and media are available to display the information to the end-user. These include visual display mechanisms, such as 3-D virtual batttlespace, and spoken dialogue interaction with virtual conversational characters (VCCs) that allow presentation of information through speech as well as through textual displays (Taplin *et al*. 2001; Broughton *et al*., 2002; Estival *et al*., 2003). While these have so far been studied and implemented somewhat independently of each other, ultimately all the different available means to present the information to the end-user must work together and be combined into a coherent whole; otherwise, the result would be very confusing to the user.

From the delivery perspective (as opposed to the input fusion aspect) with which we are concerned here, FOCAL can be considered as an instance of an intelligent multimedia presentation (IMMP) system (see Bordegoni *et al*., 1997 for a reference model).

1.2 An IMMP Architecture for FOCAL

The framework for the design of an intelligent multimedia presentation system (IMMP) within FOCAL was the result of a collaboration between DSTO and CSIRO. The aim was to design an architecture for the information delivery component, taking into account the existing architecture for the overall system, the available data sources and the type of desired presentations.

One of the main idea in FOCAL is that a VCC will serve as a Virtual Adviser (VA) to the team of commanding officers engaged in the planning or conduct of an operation. The aim of the VA is to engage in interactions with the officers, presenting information and offering advice. VAs are able to present the information and justify their advice through multimedia presentations (e.g., speech, video, text, map, etc.).

The remainder of this paper is structured as follows: in Section 2, we first briefly explain how research in multimedia presentation has grown from notions and systems developed in natural language generation. We then describe the process of generating multimedia

presentations, and, in particular, an approach to integrate coherently multimedia content, with examples from the FOCAL scenario. In Section 3 we describe the design for an IMMP architecture based on the reference model for FOCAL. We conclude in Section 4 with a short discussion of the evaluation to be undertaken.

2 IMMP Systems

Intelligent multimedia presentation systems (IMMP) are characterised by their capacity to automate the design of multimedia presentations. IMMP systems typically base their design decisions on explicit representations of diverse knowledge, and combine mechanisms and techniques that select, organise and coordinate relevant information across appropriate media. Such systems present the advantages to be:

- Adaptable and flexible by generating on-the-fly multimedia presentations of various combinations of information and media characteristics;
- Consistent by coordinating content within and across media, thus maintaining the coherence of the presentation; and,
- Effective by designing presentations that take into consideration the characteristics of the information source, the task that the users need to perform and the communicative goals to be achieved.

We first provide in Section 2.1 an overview of the principles that have guided the research in multimedia information presentation and describe in Section 2.2 the standard Reference Model for IMMP. We then present the process of generating multimedia presentations proposed by Colineau and Paris (2003), highlighting the main steps. In Section 2.3, we discuss the various issues encountered in integrating information across multiple media and illustrate the approach with an example from the FOCAL scenario.

2.1 Background

Studies in natural language generation have considerably influenced the research directions in multimedia information presentation, in particular on the issues of how to represent the global discourse structure, and how to organise and integrate each source of information in relation to the others. Several important notions have contributed to the progresses made in this domain:[1]

- the notion of discourse structure and the generation of multi-sentential texts, as embodied, for example in (McKeown, 1985a; 1985b; Moore and Paris, 1993);
- the notion of coherence and the rhetorical dependencies between discourse parts, as defined, for example, in Rhetorical Structure Theory (RST) (Mann and Thompson, 1988); and finally,
- the hierarchical planning approach as a means to structure and to represent a discourse goal hierarchy and the relationships between them, as in (Hovy, 1988; Moore and Paris, 1993) inter alia.

Starting from these notions, the generation of multimedia information presentations has been considered by many researchers (e.g., André and Rist, 1990; 1993; Maybury, 1993; Bateman *et al*., 1998; Green *et al*., 1998; Mittal *et a*l., 1998) as a goal-directed activity that starts from a communicative goal (i.e., a presentation intent), which is then further refined into communicative acts. Indeed, based on studies done in linguistics and philosophy (e.g., Austin 1962; Searle, 1969), in discourse (e.g., Grosz and Sidner, 1986) and in text planning (e.g., Hovy, 1988; Arens *et al*., 1993; Moore and Paris, 1993), the multimedia generation community has built on the idea that the internal organisation of a discourse or a presentation is composed of a hierarchy of communicative acts, each act supporting a specific communicative goal that contributes to the whole. It has then extended this principle to multimedia material. Thus, as pointed out by Maybury (1993, p.61):

"*As text can be viewed as consisting of a hierarchy of intentions, similarly, multimedia communication can be viewed as consisting of linguistic and graphical acts that, appropriately coordinated, can perform some communicative goal*".

Consequently, a question arises as to how to coordinate linguistic acts with other forms of expression (picture, graphics, video, etc.), so that the communicative goal is achieved in a coherent and consistent manner.

---

[1] See (Colineau and Paris, 2003) for details.

2.2 A Reference Architecture for IMMP

In recent years, a standard Reference Model (RM) for IMMP systems has been proposed by Bordegoni *et al*. (1997), aiming at providing a conceptual design of IMMP systems. The architecture is decomposed into five layers as follows:

- The Control Layer controls the generation process by prioritising the communicative goals to be processed;
- The Content Layer organises the content and makes explicit the relationships between discourse segments. It selects relevant information and chooses the appropriate modalities and media to be employed to convey the information and best achieve the communicative goals;
- The Design Layer distributes to dedicated media/modality design modules communicative acts to be encoded. It also determines the spatial and temporal arrangements of media objects in the presentation. The design plan specifications produced for media objects are then passed onto the realisation layer;
- The Realisation Layer distributes the design plan specifications to dedicated modules for the production of specific media objects. Specifications of displayable media objects with layout prescriptions are finally given to the presentation display layer; and,
- The Presentation Display Layer combines media objects, defines the document or the display layout and finally delivers the multimedia presentation through specialised media devices. The result is a coordinated fusion of the output of the different devices. Here, Bordegoni *et al*. point out a clear distinction between the design and the production of media objects and their presentation.

We will not discuss here the Control Layer, as in the FOCAL system it is integrated with the overall dialogue and interaction management process (see Section 3). The other four layers are illustrated in Figure 1 and constitute the actual multimedia generation process.

The main steps that drive the multimedia presentation design are:

- *The content planning*: this stage aims at selecting and organising the content of the presentation. A discourse structure is produced, which makes explicit the role of each piece of content regarding to the whole presentation.
- *The media allocation and content realisation*: this stage aims at specifying how the content should be presented. One has to decide on the best way to realise the content and to combine the different discourse parts in a unified and integrated whole. We group here both the "Design of the presentation structure" and the "Realisation of the media objects"; and
- *The layout planning*: this stage aims at assigning location to content, grouping and aligning element of content to contribute to the legibility and readability of the presentation. For dynamic presentations, there is also a need to program the execution of the presentation, in particular setting the timing of all components.

In this paper, we focus on the content planning stage, but interested readers are referred to (Colineau and Paris, 2003) for details about the other stages.

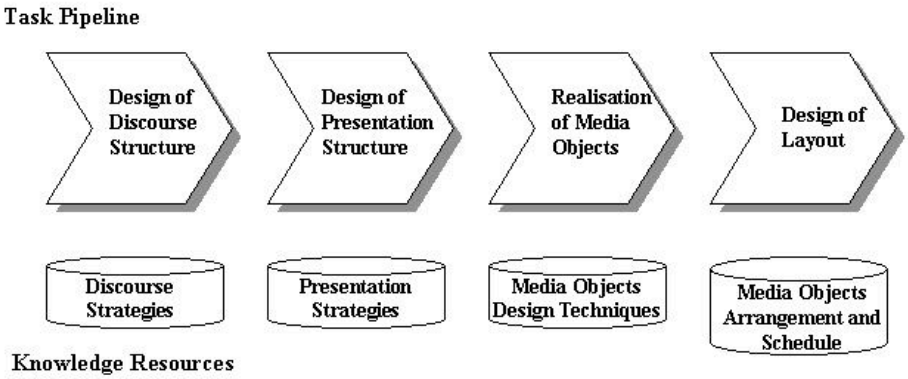


Figure 1: Multimedia generation process

2.3 Content Planning

When dealing with multimedia presentations, a number of issues that do not occur in simple text planning arise:

- How can we maintain the coherence of a presentation when the content is realised through different modalities (i.e., language, graphics, video, etc.)?
- How do graphical representations, animations, etc. work? Do they have an internal structure (as text does) that can be expressed in term of rhetorical and discursive dependencies?
- Can we use a common representation to express both textual and graphical acts?

Following research in the field of text generation, most multimedia information presentation systems have taken a unified approach, based on hierarchical planning, to structure and organise multimedia data. In parallel, by applying the principle of textual coherence to multimedia information presentation, researchers have generalised the RST theory of coherence to the broader context of multimedia information.

Using this theory, the organisation of the document or the presentation is represented by a tree structure (i.e., the document discourse structure). It is the output of the content planner, and it provides a detailed representation of the content to be produced, indicating how parts of the structure are related and which purposes different parts of the generated content serve (see Figure 2). In particular, this permits an explicit representation of the relationships and dependencies between discourse segments, whichever modalities and/or the media are selected afterwards.

The discourse structure[2] shown in Figure 2 illustrates the discourse representation that might be built to represent and organise the content of an "induction brief executive summary", from a military plannning exercise (with a fictitious scenario and fictitious data). This structure organises the different content elements (e.g., executive summary sentences, maps) and highlights their respective roles within the presentation (e.g., providing background information or evidences supporting a claim). We see that this executive summary is an integrated combination of text and illustrations (potentially static or dynamic illustrations). This example shows that the discourse structure may represent text as well as other multimedia contents, and that it can explicitly represent relationships across modalities (e.g., an illustration that supports text) and within modality (e.g., text that elaborates on another text part). If we examine the top of the discourse tree, it is organised into three discourse segments:

- the main node, which is a complex discourse segment considered as the nucleus (segment [2-5] + additional illustrations); and,
- two other discourse segments considered as satellites. One of the satellites (segment [1]) is linked to the nucleus by the rhetorical relation called *preparation*. This relation indicates that the satellite presents information which introduces the content presented by the nucleus. The other satellite is a complex discourse segment linked to the nucleus by the *elaboration* relation,[3] which indicates that the satellite provides additional information (e.g., geographic illustrations).

The discourse structure that is produced at the end of the content planning process presents several advantages. It provides a rich structure that can be reasoned about for a number of purposes, e.g., appropriate realisation in language, the placement of hypertext links, reasoning about user feedback and prior discourse, the coordination (as opposed to juxtaposition) of text, image, graphics or video.

Using a hierarchical planning approach ensures the unity of the whole multimedia information presentation by organising the entire presentation as one discourse structure, even though subparts may correspond to elements to be realised in different modalities and/or media. Having one overall discourse structure enables and facilitates the integration of various discourse elements.

[2] The discourse tree has been simplified for readability.

[3] Depending on the role of the satellite and the purpose of the information, the link between the satellite and the nucleus could also have been realised by the *enablement* relation. In that case, the information carried by the satellite would have supported the hearer in locating the region discussed in the summary.

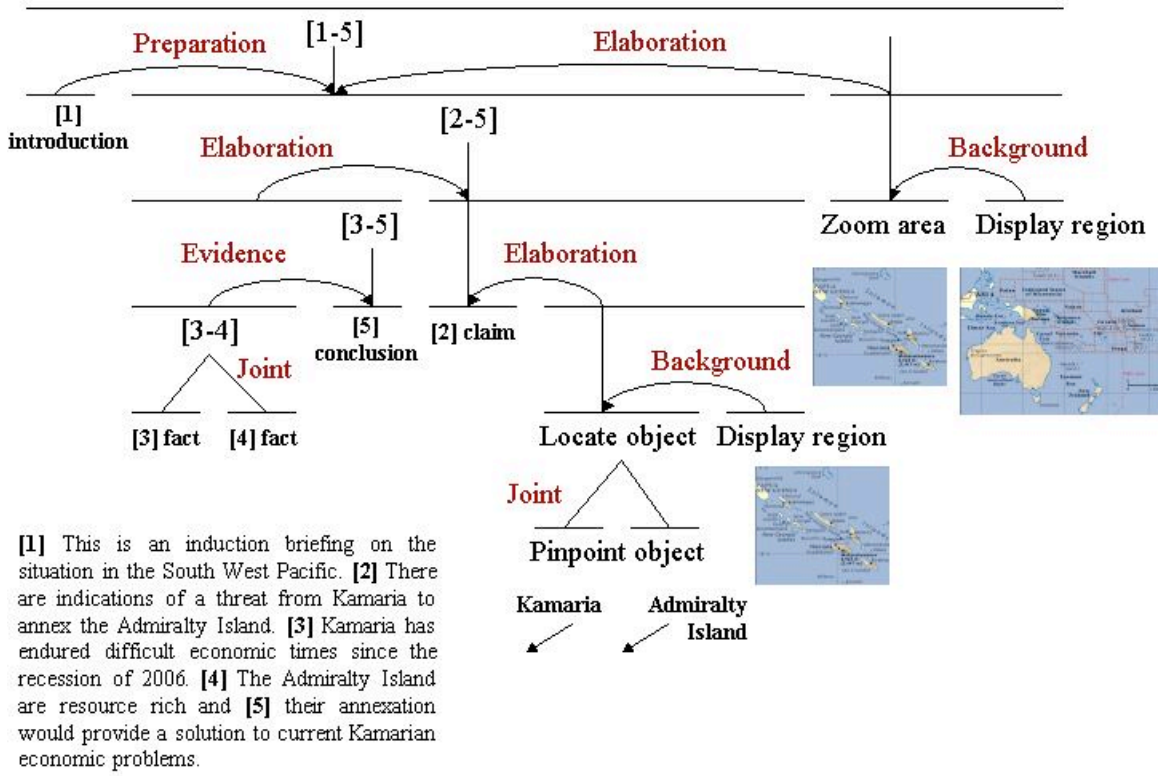


Figure 2: Example of multimedia content represented with RST (a fictitious scenario)

It also allows cross-references from one modality to the other (e.g., from text to graphics). This is explicitly stated in André and Rist (1995, p.9):

"*It seems reasonable to use text planning approaches not only for the organization of the textual parts of a multimedia presentation, but also for structuring the overall presentation. An essential advantage of a uniform structuring approach is that not only relationships within a single medium, but also relationships between parts in different media can be explicitly represented.*"

This integrated representation enables the delivery component of the system to act as a media coordinator in the preparation of the final presentation script, ensuring for example that parts of the presentation are not duplicated. It then becomes possible to factor out the needs of each individual presentation segment and to share the media objects throughout the presentation. This integrated view of the multimedia presentation also ensures that parts of the presentation are coherent and well integrated with each other. With this approach, we can set and evaluate some basic multimedia principles, such as the principle of modality, contiguity, coherence and redundancy.

3 FOCAL: a Command and Control Environment

We now describe how we have extended the original FOCAL architecture to support the generation of multimedia presentations for military planning information.

3.1 Multimedia Presentation in the Focal Architecture

FOCAL is based on a multi-agent architecture, implemented using ATTITUDE, a high-level language developed at DSTO (Lambert and Relbe, 1998). ATTITUDE is capable of representing and reasoning with uncertainty about multiple alternative scenarios (Lambert, 1999). Extending the original FOCAL architecture (Taplin *et al.* 2001) for IMMP involved adding agents to explicitly handle the design, the composition and the realisation of multimedia objects. The original "Conductor" agent, which had so far only been concerned with spoken input, was renamed Dialogue Manager (DM). It is now responsible for dialogue flow control and for understanding users' query, deciding what best answers the user's needs (i.e., the communicative goal). A new MultiMedia Presenter (MMP) agent has been introduced. It organises the presentation of information within FOCAL and carries out most of the multimedia generation process shown in Figure 1. The third stage (i.e., the realisation of media objects) is left

to specific media generators, such as the natural language generator or the virtual video generator as shown in Figure 3. The data to be presented is accessed through the MMP agent, which determines, as part of the discourse plan, what information to include and integrate. The data come from various and heterogeneous sources, including spoken and typed input.

Figure 3 shows the FOCAL architecture from a presentation of information point of view, leaving aside aspects related to the processing and fusion of the input and connections to the Information Sources. It shows the different components and the main interactions amongst them. The architecture is organised around the two main agents for IMMP: the DM agent, responsible for the overall interaction, and the MMP agent, responsible for building a presentation and realising it using different media. They both act as "conductors": one for understanding, the other for generation.

Comparing the new architecture for FOCAL with the reference architecture for IMMP proposed by Bordegoni *et al*. (1997), the DM agent can be seen as corresponding to the Control Layer, deciding which communicative goals should be processed (i.e., the purpose of the presentation), while the MMP agent assumes the processes performed by the Content and Design layers.[4]

3.2 Interaction flow process

In the current FOCAL scenarios, there are two modes: (1) the virtual adviser (VA) "pushes" the information that needs to be presented, namely delivers the briefing content, and (2) the VA allows users to ask questions to repeat or gain information.

With the IMMP architecture for FOCAL shown in Figure 3, these two modes follow the same flow process. In both cases, the aim is to answer either an explicit or an implicit information need by presenting information through complementary media. The information need may have been initiated by the system (i.e., briefing mode) or initiated by the user (i.e., question-answering/dialogue mode).

When the system is answering a user's query, the DM agent has to understand the user's query in order to identify what is the user's information need.[5] This requires the DM agent to have access to domain knowledge, e.g., an ontology of the domain (see Nowak *et al*., 2004), to ensure that the query makes sense (i.e., is syntactically and semantically well-structured). Then, the DM's aim is to determine a communicative goal which answers this information need and to send this goal to the MMP agent. The communicative goal thus constitutes the input to the MMP agent. From this input, the MMP selects the appropriate discourse strategies to be developed (e.g., "explain mission").

Once the MMP receives a communicative goal, it can develop a discourse plan to satisfy this goal. The discourse plan aims at selecting the relevant content and organising it. A set of queries is thus sent to the Query agent to acquire the content identified.

Depending of the level of knowledge and expertise of the Query agent, the queries can either be forwarded to a specific Information Source (IS) agent responsible for the information requested, or be forwarded to all IS agents. In the latter case, the Query agent will have to choose the most appropriate amongst the responses received and send these to the MMP. The Query agent thus acts as an interface between the IS agent and the MMP agent. When the content of the information to be presented has been retrieved, the MMP allocates the realisation of each discourse segment (i.e., presentation unit) to the media-specific generators. The decision to encode information under a particular modality is made by taking into account several criteria represented as declarative rules and used by the planner engine.

Finally, the MMP has to supervise the realisation of each discourse segment. It acts as a media coordinator to ensure that each media-specific generator agent is working towards a consistent and synchronised presentation. The MMP ensures that the presentation plan is built cooperatively and that alternatives are negotiated if needed. Thus, the presentation design planning is a cooperative process amongst the media-specific generator agents, supervised by the MMP. In comparison with the reference architecture model, the MMP shares with the media-specific generator agents the tasks performed in the design layer of the architecture.

Each media-specific generator agent receives a discourse segment to be realised. It develops the design of this segment closely with the MMP before starting the generation process. These agents use specific knowledge sources (e.g., grammar and lexicon, icons region models, texture models, graphics techniques, etc.). In comparison with the reference architecture model, the media-specific generator agents perform the tasks represented in the realisation layer of the architecture.

[4] We are currently collaborating with UniSA on the design of a Media Selection agent and a Media Presentation agent for these two layers.

[5] In briefing mode, the DM agent generates the information need, while in dialogue mode, the information need comes from the input devices, whose output is sent to the Input Fuser and then to the DM.

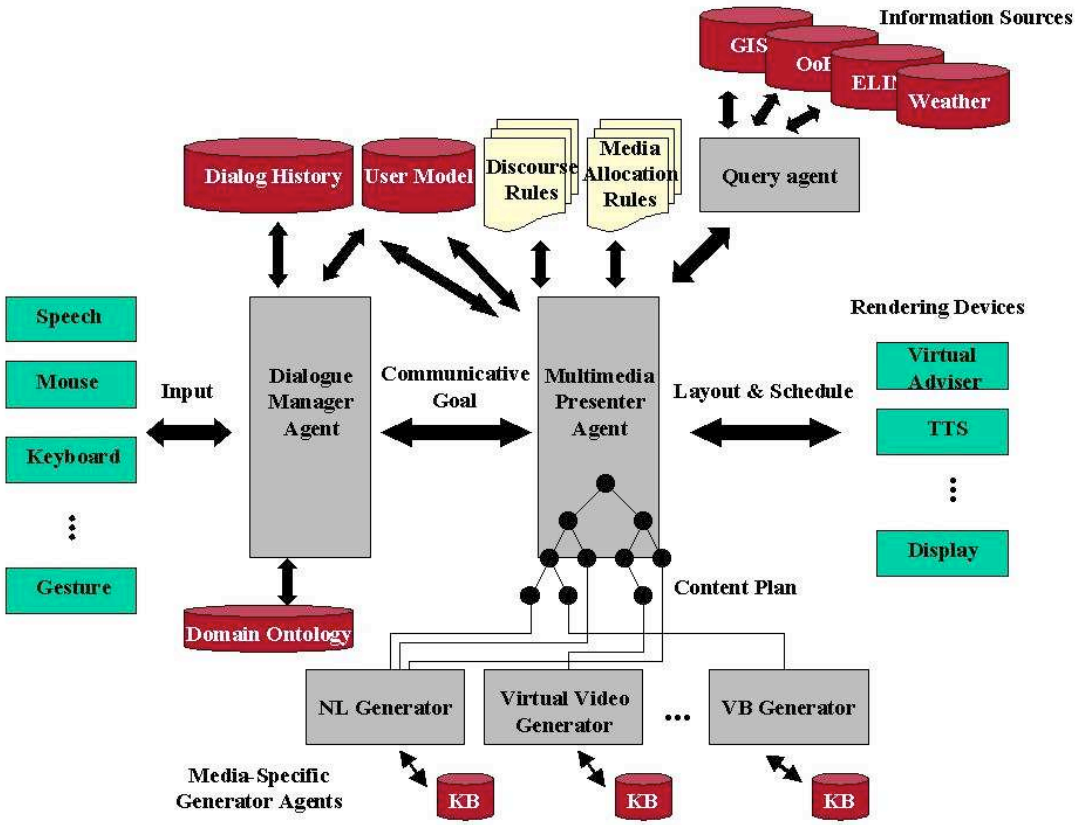


Figure 3: IMMP architecture for FOCAL

When each presentation unit has been realised and appropriately scheduled on a single timeline by the MMP, they are sent to their specific rendering devices to be displayed (cf. the presentation layer of the reference architecture).

The architecture and the interaction process flow described above have been designed to handle an interaction between a user and the FOCAL system. This means that, during a session, a user may interact with several virtual advisers. Virtual advisers can be considered as a means to interact with the system in the same way as a mouse or a pointing device. In this case, one DM and one MMP drive the interaction and provide appropriate answers. However in the case of multiple users interacting simultaneously with the system and in particular with different virtual advisers, the architecture will need to be extended to support parallel interactions. It will be necessary to have one DM and one MMP per interaction stream.

4 Discussion

This work is still in its early stages, and the architecture proposed here has not yet been fully implemented; however the current FOCAL system is very much in line with the IMMP architecture. Although attention has so far been put mainly on the spoken dialogue with the virtual advisers (Estival *et al*., 2003) and on the integration of new input modalities within a unified framework (Wark *et al*., 2004), our intention is to continue the work to produce appropriately integrated presentations. To conclude this paper, we would like to briefly discuss issues of evaluation.

The FOCAL environment may be evaluated at different levels and for different purposes. The aspect which concerns us here is the generation of multimedia information and its integration across several modalities or media. The important questions then are whether users receive enough information, whether the information is relevant for them to accomplish their task, and whether the information has been appropriately represented and integrated. Through evaluation, we would like to be able to answer questions such as:

- Is a particular medium/modality to be preferred for the encoding of specific information in order to facilitate the comprehension and retaining of that material? Which information should be represented under which format?
- Which modalities best complement each other?
- How can we avoid the split-attention effect in multimedia material?
- Does the verbal or visual representation of information have an impact on its processing by users?

5 Acknowledgements

We would like to thank our colleagues in the FOCAL team, Dr Steven Wark, Michael Broughton and Andrew Zschorn for their invaluable contribution to this project. We also wish to acknowledge the support of Nuance for the development of the speech recognition system.

References

André, E. and Rist, T. (1990) Towards a plan-based synthesis of illustrated documents. In *Proc. of 9th ECAI*, Stockholm, Sweden, 25-30.

André, E. and Rist, T. (1993) The design of illustrated documents as a planning task. In M. Marbury (Ed.), *Intelligent Multimedia Interfaces*, ch 4, 94-116. AAAI Press / The MIT Press.

Arens, Y., Hovy, E. and van Mulken, S. (1993). Structure and rules in automated multimedia presentation planning. In *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambery, France.

Austin, J. (1962) *How to do things with words*. Ed. J.O. Urmson. England: Oxford University Press.

Bateman, J., Kamps, T., Kleinz, J. and Reichenberger, K. (1998). Communicative goal-driven NL generation and data-driven graphics generation: an architectural synthesis for multimedia page generation. In *Proc. of the 9th International Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada*.

Bordegoni, M., Faconti, G., Maybury, M.T., Rist, T., Ruggieri, S., Trahanias, P. and Wilson, M. (1997). A standard reference model for intelligent multimedia presentation systems. In *Computer Standards and Interfaces: the International Journal on the Development and Application of standards for computers, Data Communications and Interfaces*, 18(6-7).

Broughton, M., O. Carr, D. Estival, P. Taplin, S. Wark and D.Lambert (2002). "Conversing with Franco, FOCAL's Virtual Adviser". *Human Factors 2002*, Melbourne.

Colineau, N. and Paris, C. (2003) Framework for the Design of Intelligent Multimedia Presentation Systems: *An architecture proposal for FOCAL*. CMIS Technical Report 03/92, CSIRO, May 2003.

Estival, D. Broughton, M., Zschorn, A. and Pronger, E. (2003). Spoken Dialogue for Virtual Advisers in a semi-immersive Command and Control environment. In *5th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, pp 125-134.

Green, N., Carenini, G., Kerpedjiev, S., Roth, S. and Moore, J. (1998). A media-independent content language for integrated text and graphics generation. In *Proc. of the COLING'98/ACL'98 Workshop on Content Visualization and Intermedia Representations* Montréal, Canada.

Grosz, B. and Sidner, C. (1986). Attention, Intentions, and the structure of discourse. In *Computational Linguistics* 12(3), 175-204.

Hovy, E. (1988). Planning coherent multisentential text. In *Proc. of the 26th Conference of the ACL*, Buffalo, NY, 163-169.

Lambert, D. and Relbe, M. (1998). Reasoning with Tolerance. In *Proc. of the 2nd International Conference on Knowledge-Based Intelligent Electronic Systems*. IEEE. pp. 418-427.

Lambert, D. (1999). Advisers With Attitude for Situation Awareness. In *Proc. of the 1999 Workshop on Defence Applications of Signal Processing*. pp.113-118, LaSalle, Illinois.

Mann, W. and Thompson S. (1988). Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *Text*, 8(3), 243-281.

Maybury, M. (1993). Planning multimedia explanations using communicative acts. In M. Marbury (Ed.), *Intelligent Multimedia Interfaces*, chapter 2, 59-74. AAAI Press / The MIT Press.

McKeown, K. R. (1985a). Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1-42.

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

Mittal, V., Moore, J., Carenini, G. and Roth, S. (1998). Describing complex charts in natural language: a caption generation system. In *Computational Linguistics, Special issue on Natural Language Generation. Vol. 24, issue 3*, 431-467.

Moore, J. and Paris, C. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. In *Computational Linguistics*, vol.19, Number 4, 651-694.

Nowak, C, D. Estival and A. Zschorn (2004). "Towards Ontology-based Natural Language Processing". RDF/RDFS and OWL in Language Technology: 4th Workshop on NLP and XML (NLPXML-2004), ACL 2004, Barcelona, Spain. pp-59-66.

Taplin, P., G. Fox, M. Coleman, S. Wark and D. Lambert (2001). "Situation Awareness Using A Virtual Adviser". OzCHI.

Wark, S., A. Zschorn, M. Broughton and D. Lambert. (2004). "FOCAL: A Collaborative Multimodal Multimedia Display Environment". *SimTecT 2004*, Canberra, Australia. pp-298-303.

Selecting Systemic Features for Text Classification

Casey Whitelaw and **Jon Patrick**
Language Technology Research Group
Capital Markets Co-operative Research Centre
School of Information Technologies
University of Sydney
{*casey, jonpat*}*@it.usyd.edu.au*

Abstract

Systemic features use linguistically-derived language models as a basis for text classification. The graph structure of these models allows for feature representations not available with traditional bag-of-words approaches. This paper explores the set of possible representations, and proposes feature selection methods that aim to produce the most compact and effective set of attributes for a given classification problem. We show that small sets of systemic features can outperform larger sets of word-based features in the task of identifying financial scam documents.

1 Introduction

Text classification is among the most widespread applications of computational linguistics. The range of services that offer text classification continue to grow, and include such mainstream applications as email and web content filtering. The classification of documents by machine learning techniques requires a representation of each document as a set of features; almost without exception, these features are based on the presence, absence, or frequency of words in the text. This 'bag-of-words' model is popular both due to its ease of implementation, and its excellent performance on many tasks. Topic-based classification, such as newswire or newsgroup tasks, is well-suited to this automated keyword-spotting approach. These are cases in which the presence of a topic-related word such as 'wheat' is a very strong indicator of a document's class.

The bag-of-words model makes large simplifying assumptions about a document. It assumes that there is no textual structure; no ordering of paragraphs in the text, sentences in a paragraph, clauses in a sentence, or words in a clause. In addition, it assumes that the occurrence of each word is independent of each other word. These assumptions, in providing a much simpler picture of the document, destroy much of the text's meaning. Work has been done to restore this information using semantic resources such as WordNet (Scott and Matwin, 1998) or using syntactic information (Carr and Estival, 2002). There is also growing interest in classifying texts on non-denotational meaning, such as writing style, authorship identification (van Halteren, 2004) and sentiment analysis (Pang and Lee, 2004). These new areas highlight the properties of a document that are currently slipping through the cracks.

This paper takes another approach to providing a better representation than bag-of-words. In line with Systemic Functional Linguistic theory, the words of a text are treated as evidence of semantic choices being made by the author. These choices form systems, and each document is modelled as the set of choices it makes within these systems. This knowledge of the semantic relationships between features allows for more sophisticated rep-

resentations that more accurately capture characteristic linguistic differences. In Section 2 we enumerate these representations and discuss how their semantics differ. Section 3 describes the Scamseek project and the use of systemic features in the identification of financial scams. The results provided in Section 4 show that smaller number of systemic features can outperform larger numbers of word-based features in text classification.

2 Systemic Features

Systemic Functional Linguistics (SFL) is a linguistic theory that approaches language as a social resource for meaning-making (Halliday, 1994). Language is not seen as a collection of discrete phrase production rules working upon a deeper syntactic structure, but as an interwoven collection of systems realising a deeper semantic structure and functional intention. SFL explicitly deals with three types of meaning (metafunctions) in text: the *ideational* (content of the text), the *textual* (organisation of the text), and the *interpersonal* (social positioning of the text) meanings, each of which contribute to the formation of a document.

SFL uses *system networks* as a way to represent the patterns of language choice related to a particular meaning. A system network is defined both graphically and algebraically (Matthiessen, 1995) as a hierarchy of choices: at the most delicate level, these choices result in particular lexical or grammatical artifacts. SF linguists have proposed standard system networks for most aspects of the English language.

SFL has been applied in natural language processing since the 1960s, but has been adopted most widely within the field of text generation (Matthiessen and Bateman, 1991). Most recently, systemic analysis has been used with machine learners in more statistical NLP tasks such as functional clause classification (O'Donnell, 2002). The increased interest in attitude and affect has also seen SFL's theory of appraisal used to augment sentiment classification (Taboada and Grieve, 2004).

Systemic features are a way to describe the usage of a system network within the document as a whole. Systemic features were introduced as a way of identifying the interpersonal distance of documents (Whitelaw et al., 2004), using only a single system network. Features from multiple system networks have been used together to classify different styles of academic writing (Argamon and Dodick, 2004). These types of grammar models have been shown to be well suited to the task of describing the non-denotational or stylistic properties of writing (Whitelaw and Argamon, 2004).

2.1 Types of System Networks

Two types of system network are used in this paper, both constructed using Systemic Functional Linguistic theory. The first, grammar models, are based on the general linguistic descriptions provided in linguistics texts, eg. (Matthiessen, 1995), and are similar to those used previously for stylistic text classification. The specific systems used here include:

- CONJUNCTION: models how clauses expand on their context through elaboration (*that is*), extension (*moreover*), or enhancement (*then, next*).
- PRONOMINAL/DETERMINATION: models the way in which referents are identified in a text. This system has been used to classify texts on the basis of interpersonal distance (Whitelaw et al., 2004)
- COMMENT: describes the status of a clause within the context as eg. evaluative/judging (*sensibly*), desiderative (*unfortunately*), or assertive (*certainly*).
- MODALITY: is a rich system that describes the likelihood (*probably*), frequency (*might*), and necessity (*should*) of events.

Grammatical models such as these provide a general profile of language use within a document and a register. An advantage of these general models is their domain independence; the distinctions made within these systems are based on the manner in which the document was written, rather than its topic. Manual linguistic research has given evidence that scam documents differ from normal documents in their language (Herke-Couchman, 2003), and so it is expected that features from these systems will assist in this classification.

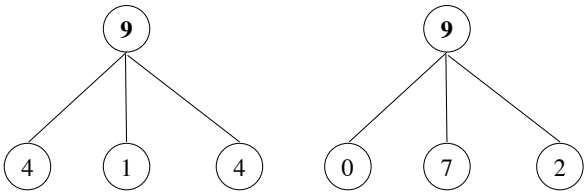


Figure 1: Aggregating counts smooths differences at greater delicacy

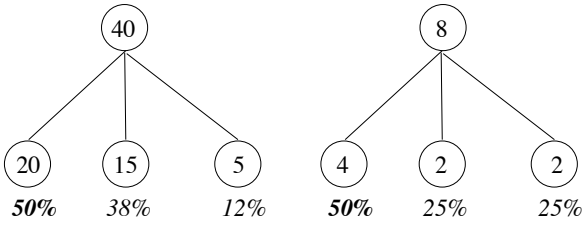


Figure 2: Proportional features are a local and size-independent measure

Register models, in contrast to the general applicability of the grammatical models, describe specific linguistic traits that are characteristics of individual registers. Register models are compiled manually by trained SF linguists based on their analysis of a training corpus. A register, in SFL terminology, is a group of texts whose language selections vary from the general language system in similar ways; a skewing 'of probabilities relative to the general systemic probabilities' (Matthiessen, 1993). In the absence of a fully developed system network for English, register models each define portions of language use that are characteristic and discriminatory within the current classification task.

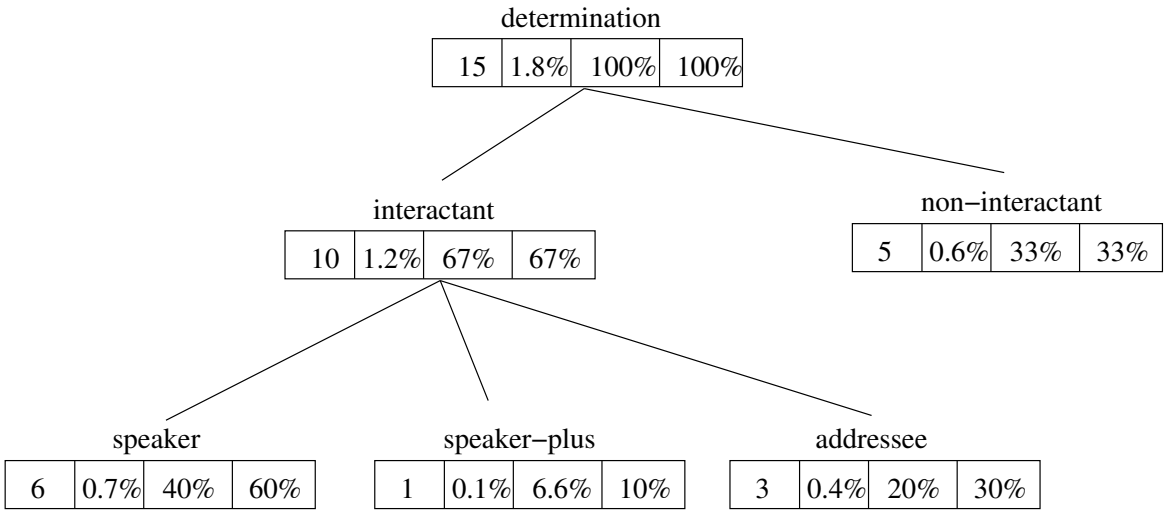
2.2 Leveraging Systemic Structure

In a standard 'bag-of-words' approach, the contribution of a word to a document is given by its relative frequency; how rarely or often that word is used. This implicitly uses a language model in which all words are independent of each other. Crucially, this does not and cannot take into account the choice between words, since there is no representation of this choice. Placing words within a system network provides a basis for richer and more informative feature representation. There are two main advantages to be gained from systemic information.

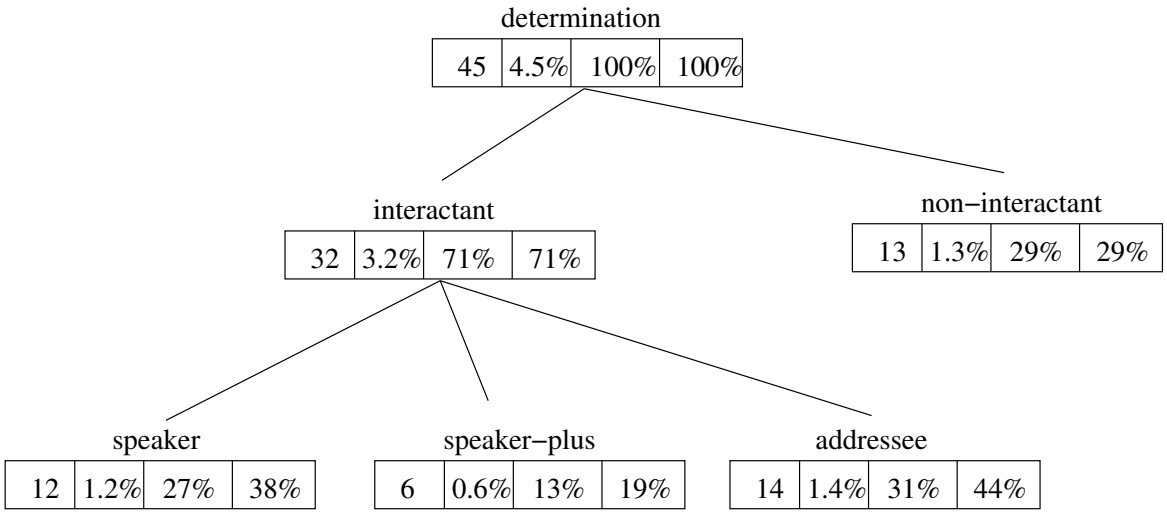
Firstly, it allows for categorical features that are based on semantically-related groups of words, at all levels in the network. By collecting aggregate counts, individual variations within a category are ignored. Figure 1 shows the raw counts of the same system in two documents; at the lower level, closer to lexis, the distributions of counts are highly dissimilar. At the higher level, these differences have been smoothed, and the documents look the same. This aggregation also helps alleviate the problems associated with representations containing large numbers of very sparse features.

For a given register, it may be the case that important and characteristic language choice occurs at a very fine level, distinguishing between usage of individual words. This word-level information is kept intact, as in a bag-of-words approach. In another register, it may be the usage of a category, such as interactant, that is characteristic. The usage of any words within the category may appear random while maintaining consistent category usage. These higher-level features are not available in a traditional bag-of-words approach, hence these patterns may be lost as noise.

The second and more important difference to traditional feature representation is the representation of language choice. SF theory treats language use as a series of selections within systems; at any point in the system network, or tree as it has been modelled here, the selection is restrained to the immediate sub-systems. The choice is not between one word and any other, or even one system and any other, but a series of semantically-driven choices within the system. A bag-of-words model can model only choice between one word and any other; a choice between arbitrary words such as 'dog' and 'elegant'. Comparative features such as these can only be used within an appropriate theory-driven structure, which is provided here through the use of SFL and system networks. Figure 2 shows the potential for comparative features to reveal similarities not immediately apparent in a text. The leftmost node in each system contributes 50% to parent system usage, despite markedly different numbers of occurrences.



Document A: 800 words



Document B: 1000 words

count	*term frequency*	*system %*	*system contribution*

Figure 3: Different feature representations portray a text differently

2.3 Representing Systemic Features

Figure 3 shows a portion of the DETERMINATION system for two documents of different sizes, belonging to the same register. Four possible feature representations are given: from left to right, each node shows the total count, term frequency, system percentage, and system contribution. Each feature representation captures a different aspect of system usage in a document and register.

Raw counts (term frequency) (first column). The summed feature count, shown in the leftmost column, presents these two documents as highly dissimilar. Note also that this is only the top portion of the system, and that multiple levels exist below those shown. Raw term counts are usually not used directly as features, as they are heavily influenced by document length.

Document percentage (second column) is the standard basis for bag-of-words representations; it gives the proportion of the document accounted for by this term. It is commonly used since it normalises for document length; most topic-based document classification rightly assumes that the document length is not important (Sebastiani, 2002). In creating features for each sub-system, this representation can still take advantage of the aggregation and smoothing provided by the system, but does not take further advantage of the known structure.

System percentage (third column) gives the proportion of total system usage made up by this sub-system. In Document A, addressee occurs three times from a total of fifteen occurrences of determination in the document, giving it a system percentage of 20%. Within a document, system percentage is directly proportional to term frequency, but is independent to system *density* in the document. If another 800 words were added to Document A, but no more uses of DETERMINATION, the term frequency for a feature would halve while the system percentage remained constant. This makes it a suitable representation where distinctions are made not on how often a feature occurs, but the manner of its use. The system percentage of *speaker* is higher in Document A than Document B, despite higher term frequency in the latter. System percentage is also useful when the area of interest is a constant-size subsection of a variable-length document.

System contribution (fourth column) shows the ratio of sub-system to super-system occurrence. Again in Document A, speaker occurs six times and its super-system, interactant, occurs ten times, giving a system contribution of 60%. This is a strictly local measure of usage, and captures most directly the systemic notion of choice: once the decision to use a given super-system has been made, how often was this sub-system chosen as the realisation? This is a relative feature, and as such is independent of document length, total system usage, and usage of other portions of the system (see Figure 3). Despite the differences in lower-level choices, and in the raw counts of system usage, the system contribution of interactant in Documents A and B are very similar.

System contribution is not proportional or strongly correlated to document percentage, and the two measures provide useful and complementary information. Within a system instance, document percentage can be used to report the frequency not just of terms but of systems as well. System contribution does not capture how often a system is used, but rather its usage in relation to the other possible choices. In the same way as a register may be characterised by choice, it may also be characterised by frequent usage of a particular system, which will be highlighted by system percentage. The four complementary representations given here may each be useful in discerning characteristic system usage.

In implementing these representations, it is worth noting that not all system contribution features are necessary, and some can be removed. Features from a node which is an only child do not add information since there is no choice. In a system with a binary choice, either one of the features may be discarded since they have unit sum. Both system percentage and system contribution are meaningless at the root level, and system percentage and system contribution are identical at the first level below the root. These feature reductions can be performed deterministically before any further feature selection.

By mapping only the relevant portions of a document's meaning, systemic features also have the potential to increase computational efficiency by

reducing the number of attributes used in machine learning systems, in comparison to broader bag-of-words methods. This should produce smaller feature sets with equal or better performance.

2.4 Selecting Systemic Features

We have presented four potential feature representations for systemic features. Depending on the behaviour of a system network in a particular classification task, the most appropriate representation may vary. In addition, the best feature type may change within a single system. We propose a simple feature selection method for systemic features.

For a given task, the attribute significance of each possible feature representation can be measured using a method such as information gain. By ranking the options for a single node upon an information metric, the best feature type for each node can be selected. This reduces the number of features, reduces the chance of performance loss through correlated features, and should combine the strengths of each feature type.

3 Scamseek: Identifying Financial Scams

We tested this range of possible systemic feature representations using models and data compiled as part of the Scamseek project[1]. Scamseek aims to identify a variety of criminal financial scams on the internet, using a combination of automatically and linguistically derived criteria.

The entire Scamseek corpus, collected and manually classified by ASIC experts, contains 7556 documents in a total of 58 registers. These registers fall into four broader classes which group financial scams, other scams, legitimate financial documents, and all other web pages. This coarser classification is of the most interest to the client, as potential scams are investigated regardless of scam type. For these experiments we used 1896 documents from 22 registers with a minimum of 20 documents per register.

As well as existing grammar models, a register model was developed by SF linguists for each of the Scamseek registers. These were treated in two ways.

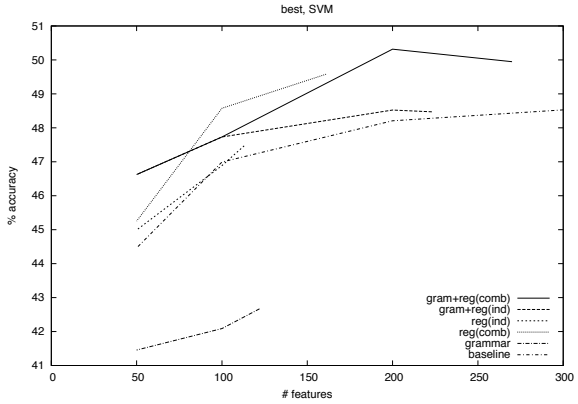


Figure 4: Results for each set of models, selecting the best feature at each node

Each register model can be considered as a selection from a full systemic description of English. Taken individually, a register model aggregates all topic- and genre-specific features, producing an overall picture of the 'topicality' of a text. The register models can also be combined to form a single system network, which is more complete but not topic-specific. In this case, it should function more like a grammar model in that the relative usage of systems should become more important. Both of these options were tested.

As well as testing grammar models and register models independently, the two types of system networks were combined. In all cases, features with no variation or no occurence in the corpus were removed. The systemic features were extracted from documents using an efficient partial parsing method (Whitelaw and Argamon, 2004). Each of the feature representation methods given in Section 2.3 were tested individually and in combination ('all'). The best-feature-per-node feature selection method ('best') was also tested for each feature set.

As a baseline, we used a bag-of-words representation using all of the words and phrases included in all the grammar and register models. Each feature set was tested at various sizes, using information gain to select features. Tests were performed using ten-fold cross-validation and the support vector machine (SVM) (Platt, 1998) im-

[1]Scamseek is a joint project funded by The University of Sydney, the Capital Markets Cooperative Research Centre, the Australian Securities & Investments Commission and Macquarie University.

	gram	reg (ind)	reg (comb)
term frequency	24%	19%	14%
document %	20%	**36%**	**30%**
system %	18%	18%	27%
sys. contribution	**38%**	27%	29%

Table 1: Proportion of each feature type selected for 'best' sets

plementation used in the WEKA machine learning environment (Witten and Eibe, 1999) [2]

4 Results

Figure 4 shows the results from selecting the best combination of feature types. This also shows the best overall result achieved at 50.4%, using 200 features selected from the grammar and combined register models. This outperforms the full baseline result by two percent. The grammar models alone perform much lower than any of the register models, which is to be expected on this register-based classification.

Table 1 shows what types of features were selected in the best-feature-per-node process. As expected, the densely populated grammar models select more system contribution features. When register models are used individually they are very sparse, and there is less benefit from including relative features. In this case, document percentage makes up 36% of the feature set. The combined register model, which forms a single more fully-specified system network, selects equally from all feature types except term frequency. All representations were used by all models, and it is through this corpus- and system-specific selection that the best combination of feature types is found.

The relative performance of each feature type can be seen in Figure 5. As in most text classification, raw counts do not work as well as normalised features. Including all features from all nodes regardless of potential correlations, as shown by the solid line, produces worse results than using only the best combination of features.

As discussed in Section 2.2, features higher in a system network aggregate and smooth the features below it. When it is the use of semantic *categories*

[2]Each experiment was also run using J48 decision trees and Naive Bayes, but produced consistently poorer results.

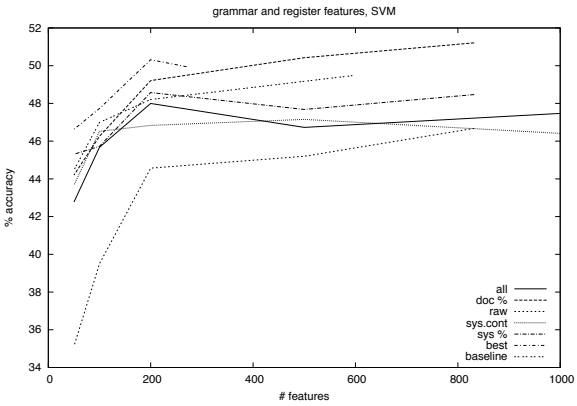


Figure 5: Results for each feature type

	features	accuracy
baseline	594	82.2%
grammar	200	82.4%
combined	200	84.4%
Scamseek	> 5000	$> 90\%$

Table 2: Class-based accuracy results.

of words that is important, these internal features will be favoured over lexis. This is the case for all the models tested: of the top hundred features in grammar models, 83 are internal. Experimental results bear out the advantage, with better performance for systemic features than the lexis-only baseline when both use document percentage.

Table 2 shows the class-based accuracy results for the best feature sets obtained. Registers are more similar to other registers in the same class, resulting in much higher performance than when classifying by register. The best set of 200 systemic features performed 2% better than the baseline bag-of-words system. Grammar models also outperformed the baseline despite poor register accuracy. This is evidence of the stylistic differences between these categories. The full Scamseek system, which combines bag-of-words features with more systemic features and other processing such as entity recognition, uses many more features and achieves much higher performance.

5 Conclusions

A document is more than a bag of words. As the forms of document analysis and classification con-

tinue to expand beyond topic detection, we must move towards a richer representation of a document. SFL provides one such linguistic model, and the representation of system models as features presented here shows the efficacy of a theoretically motivated approach. Systemic features allow for the production of smaller, denser feature sets that contain more sophisticated features than traditional methods. Grammar models can help build stylistic profiles of texts; register models supplement these with genre-specific linguistic phenomena. Through their combination, and a combination of new feature representations such as system contribution and system percentage, we have shown increased performance on the difficult task of identifying financial scams.

The system networks used in this research are still heavily tied to lexical realisations. The systemic feature extraction process can be efficiently expanded to include morphosyntactic and simple grammatical relationships; this will allow for the description of linguistic phenomena related to the logogenesis or unfolding of a text, such as the relative ordering of features. As more system networks are constructed for core sections of English SFL grammar, these models will be beneficial to a wide range of tasks including the classification of style, sentiment, attitude and affect.

References

Shlomo Argamon and Jeff T. Dodick. 2004. Linking rhetoric and methodology in formal scientific writing. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*.

Oliver Carr and Dominique Estival. 2002. Text classification of formatted text documents. In *Proceedings of the 2002 Australian Natural Language Processing Workshop*.

Michael A. K. Halliday. 1994. *Introduction to Functional Grammar*. Edward Arnold, second edition.

M. A. Herke-Couchman. 2003. Arresting the scams: Using systemic functional theory to solve a hi-tech social problem. In *ASFLA03*.

C. M. I. M. Matthiessen and J. A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Frances Pinter Publishers and St. Martin's Press, London and New York.

C. M. I. M. Matthiessen. 1993. Register analysis: theory and practice. In *Register in the round: diversity in a unified theory of register*, pages 221–292. Pinter, London.

Christian Matthiessen. 1995. *Lexico-grammatical cartography: English systems*. International Language Sciences Publishers.

M. O'Donnell. 2002. Automating the coding of semantic patterns: applying machine learning to corpus linguistics. In *Proceedings of the 29th International Systemic Functional Workshop*. University of Liverpool.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL 2004, Main Volume*, pages 271–278, Barcelona, Spain, July.

J. Platt, 1998. *Advances in Kernel Methods - Support Vector Learning*, chapter Fast Training of Support Vector Machines using Sequential Minimal Optimization. MIT Press.

Sam Scott and Stan Matwin. 1998. Text classification using WordNet hypernyms. In Sanda Harabagiu, editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38–44. Association for Computational Linguistics, Somerset, New Jersey.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

M. Taboada and J. Grieve. 2004. Analyzing appraisal automatically. In *AAAI Spring Symposium of Exploring Attitude and Affect in Text*. AAAI.

Hans van Halteren. 2004. Linguistic profiling for authorship recognition and verification. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 199–206, Barcelona, Spain, July.

Casey Whitelaw and Shlomo Argamon. 2004. Systemic functional features in stylistic text classification. In *Proceedings of AAAI Fall Symposium on Style and Meaning in Language, Art, and Music*. AAAI.

Casey Whitelaw, Maria Herke-Couchman, and Jon Patrick. 2004. Identifying interpersonal distance using systemic features. In *Proceedings of AAAI Workshop on Exploring Attitude and Affect in Text: Theories and Applications*. AAAI Press.

Ian H. Witten and Frank Eibe. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

A framework for utterance disambiguation in dialogue

Pont Lurcock, Peter Vlugter, Alistair Knott
Department of Computer Science
University of Otago
New Zealand
{pont,pvlugter,alik}@cs.otago.ac.nz

Abstract

We discuss the data sources available for utterance disambiguation in a bilingual dialogue system, distinguishing global, contextual, and user-specific domains, and syntactic and semantic levels. We propose a framework for combining the available information, and techniques for increasing a stochastic grammar's sensitivity to local context and a speaker's idiolect.

1 Introduction

Resolving the ambiguities present in an incoming utterance is a key task in natural language processing. Interpreting an utterance, whether semantically, syntactically or phonologically, is typically construed as a two-stage process: the first stage involves deriving a set of all possible analyses, using relatively well-defined principles, and the second stage involves selecting between these analyses, using principles that are harder to define and formalize.

This paper considers principles for disambiguating utterances in a human-machine dialogue system. Our main goal is to present a framework for integrating the different sources of information relevant to this task, and the available techniques for making use of this information. Working with a dialogue system highlights the need for such a framework; there are additional types of ambiguity that need to be considered (such as dialogue act ambiguity), and a particularly diverse set of informational resources to consult (many of which relate to the current dialogue context). At the same time, there are additional opportunities available—in particular, the ability to ask the user clarification questions if a given ambiguity cannot be otherwise resolved.

We begin in Section 2 with an analysis of the kinds of disambiguation needed in our dialogue system, and of the information available to the system to perform the task. In Section 3, we outline a general framework for performing disambiguation in a dialogue system. In Sections 4 and 5, we describe how our system implements this framework. We conclude in Section 6 with an account of our current work.

2 Sources of information available for utterance disambiguation

Utterance interpretation is typically modelled as a pipeline process, beginning with phonological interpretation, proceeding with syntactic analysis and semantic analysis, and concluding with discourse or dialogue attachment. We will illustrate by showing the pipeline used in our own bilingual English/Māori dialogue system, Te Kaitito (see e.g. Knott and Wright (2003); Knott et al. (2004)). Our pipeline is shown in Figure 1. Since our input is a written sentence

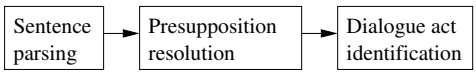


Figure 1: The utterance interpretation pipeline

we begin with sentence parsing, using the LKB system (Copestake and Flickinger, 2000). LKB works with HPSG-like grammars; the grammar we use is bilingual, simultaneously encoding a fragment of English and Māori (Knott et al., 2002). LKB's parser delivers 'flat' semantic representations in a format called Minimal Recursion Semantics (MRS) (Copestake et al., 1999), which are turned into Discourse Representation Structures (DRSs) distinguishing between an assertion and a set of presuppositions (Kamp et al., in preparation). These DRSs are then passed to a presupposition resolution module, which finds referents for anaphora, definite NPs, and other presuppositional constructions. The resolved DRSs are passed to a dialogue act identification module, which determines the dialogue act made by the utterance.

Ambiguities can arise at any point in this pipeline. There can be many parses of the sentence, many semantic interpretations of a parse tree, many ways to resolve the presuppositions in a semantic interpretation, and many ways to interpret the resulting structure as a dialogue act. By the end of the pipeline, there can be a large number of interpretations consistent with the original input sentence. How should one interpretation be chosen?

To answer this question, it is useful to survey the kinds of information available to a system for the resoluton of ambiguities. There are two ways of thinking about this information. Firstly, relevant information can appear at different **levels**, suitable for use at different points in the pipeline. Some information is **syntactic**: as is now well known[1] we can make use of statistics about the relative frequency of syntactic constructions in a corpus to decide between alternative analyses. Other information is **semantic**: for instance, we can specify a set of axioms about what are considered normal circumstances, and use a theorem-prover to determine which candidate interpretation is most consistent with these. Syntactic information can be used to resolve syntactic ambiguities, and semantic information can be used to resolve ambiguities in semantic interpretation and discourse attachment. Secondly, relevant information can come from different **domains**, of which we consider three prominent ones: the **world**, the **speaker model**, and the **dialogue context**. Domains and levels of information are roughly orthogonal; Table 1 summarizes their possible combinations.

		level	
		syntactic	semantic
domain	world	general corpus statistics	world knowledge axioms
	speaker model	language model of the user	axioms about user
	dialogue context	statistics about recent context	context-matching operations

Table 1: Domains and levels of information for use in utterance disambiguation

This division is remeniscent of Menzel and Schröder's (1999) concept of multi-level parsing, although they do not use the orthogonal domain and level classifications shown here.

To illustrate these categorizations, consider a well-known example of syntactic ambiguity:

(1) Fruit flies like a banana.

Under the most intuitive reading (call it Reading 1), the sentence is about what fruit flies like to eat. But it can also be interpreted (Reading 2) as being about the way fruit tends to fly through the air. We can use different sorts of knowledge to help decide between these alternatives.

Firstly, we could use world knowledge—for example, the fact that fruit doesn't typically fly, or that insects often like fruit. This information could take the form of axioms in some suitable logical language, directly encoding such propositions. But it could also take the form of statistics about common syntactic structures in wide-domain corpora, which indicate that the probability of the verb *fly* taking a subject headed by *fruit* is vanishingly low. These statistics can also be seen as a form of world knowledge, albeit one encoded in a far less explicit way than a set of logical propositions.

Knowledge about the speaker of the sentence is also of use in helping to disambiguate. For instance, if the speaker is known to be a geneticist, this would support Reading 1, while if she is an aerodynamicist, this might generate a weak preference for Reading 2. Again, knowledge of the speaker can either be semantic (e.g. taking the form of logical axioms) or syntactic (e.g. statistics derived from a set of her previous utterances).

Finally, an utterance's context provides strong constraints on disambiguation. Again, this information can be syntactic (a recent usage of *fly* as a verb or noun would lend support to the corresponding interpretation) or semantic, involving use of a logical representation of the dialogue context: an utterance is scored by the ease with which it can be incorporated into the context (Knott and Vlugter, 2003). For instance, if the utterance answers the question *How does fruit fly?*, we may assume Reading 2; if it answers the question *What do fruit flies like?*, we have very strong evidence for Reading 1.

In summary: we can distinguish six broad categories of information relevant to utterance disambiguation, classifying orthogonally by do-

[1] See e.g. Manning and Schütze (1999, Chapter 12).

main and level.

3 A framework for utterance disambiguation

The central question in this paper is: how can we incorporate disambiguation routines into our pipeline so as best to integrate these disparate categories of information? A simple approach would be to use a 'greedy' algorithm: choose the best syntactic analysis, using all the available syntactic data, then pick the best semantic interpretation that can be derived from this winning syntactic analysis, and so on. But this approach can throw out plausible interpretations: semantic information often *overrides* statistical information about frequency of syntactic constructions. Consider the following dialogue:

(2) System: I keep my pig in a pigpen.
User: Where is the pen?

If the system's grammar has two lexical entries for *pen*, one meaning 'writing pen' and one meaning 'pigpen', the user's utterance will be syntactically ambiguous. The intended interpretation is clearly 'pigpen', but corpus statistics are unlikely to support this; if anything, writing pens will be more common in a general corpus. (Even looking at syntactic constructions in the recent context will not help, since the word *pen* was not used in the first utterance.) However, at a higher level, presupposition resolution information can give us the right reading. The intended reading carries a presupposition that there is a pigpen, which can be successfully resolved, while the other reading presupposes a writing pen, which will not be found, and must be accommodated.

In summary, we need a way to *combine* information from different levels (syntactic and semantic) and domains (world, user and local). Two questions now arise. Firstly, when disambiguating an utterance at some point in the interpretation pipeline, how far should we **look ahead** along the pipeline? Secondly, how should we combine evaluations made at different points in the pipeline using different types of information?

3.1 Proposal for a disambiguation procedure

As to the question of how far to look ahead: we propose that we should always look to the very end of the pipeline. For instance, when performing syntactic disambiguation, we should take each possible reading, and perform semantic interpretation, presupposition resolution and dialogue attachment. Of course, each of these might themselves generate alternative possibilities. The resulting space of possibilities is rather like a conventional AI search graph, in which leaves are complete interpretations of the utterance. At each stage, the procedure which generates the ambiguities at that stage can assign each alternative a **local score**, using the information appropriate to that stage. When we have created the full set of complete interpretations, we can combine these local scores somehow (see Section 3.2) to create a **global score** for each complete interpretation.

We now need to use these scores to decide on the best interpretation. Local and global scores provide only a heuristic measure of which interpretation is best, so we can only use them as a rough indicator of which is the best reading. If one interpretation has a global score far exceeding those of all other interpretations, we can safely choose it. But if there are several alternatives with roughly the same global scores, we need to ask a clarification question, so that the user can disambiguate overtly. The selection of an appropriate clarification question is problematic in its own right, since we may be trying to distinguish between several interpretations whose high global scores originate in different stages of the pipeline. Section 3.3 discusses this topic.

3.2 Combining information types

How do we combine local scores due to statistical parsing, presupposition resolution and dialogue attachment? To date, we have considered two approaches. We first considered a **weighting formula**. In this method, all local scores are numerical, and the global score is a weighted sum of the local scores, with weights chosen so as to reflect the importance of different sources of information. However, this fails to capture potential interactions between different information sources. We now use a **conditional formula**—basically a simple procedural algorithm. In this scheme, we can specify, for instance, that a certain local score is only used if there is a tie as regards some other local score. Our general observation is that higher levels of information tend to trump lower levels; for instance, if there are two alternative interpretations of an utterance at the dialogue act level, and only one of these is consistent with a co-

herent dialogue, then it shouldn't matter if the other interpretation scores more highly in the syntactic or presupposition-resolution domains. Section 4 discusses this algorithm further.

3.3 How to generate clarification questions

A key component in a dialogue-based disambiguation system should be the ability to ask the user for clarification. There are well-known dialogue strategies for doing this; the concept of a **clarification subdialogue** is well-established as a dialogue structure—see, for example, Schmitz (1997).

It seems that different clarification questions target ambiguities at different points in the interpretation pipeline. At the syntactic level, clarification questions tend to have a **multiple-choice** structure, in which the alternative syntactic possibilities are disambiguated by rephrasing. For instance:

(3)
User: Fruit flies like a banana.
System: Do you mean (1) 'A banana is liked by fruit flies,' or (2) 'Fruit flies just as a banana does'?

At the presupposition resolution level, clarification questions tend to take the form of *wh*-questions. For instance:

(4)
User: The dog barked.
System: *Which* dog barked?

These questions are sometimes termed **echo questions** (c.f. Ginzburg (1996)). They are syntactically different from ordinary questions: for instance, the *wh* element receives a certain kind of stress, and interestingly, they can be embedded inside questions:

(5)
User: Who did the dog chase?
System: Who did *which* dog chase?

Questions about dialogue act assignment are much less common. Questions of this sort would probably include **meta-level questions** such as the following:

(6) System: Are you talking to me?

(7) System: Are you asking me, or telling me?

Recall from Section 3.1 that a clarification question will be asked if the highest-ranked interpretation is close enough in score to one or more lower-ranked interpretations. What sort of clarification question should we then generate?

Our key suggestion here is that we need to nail ambiguities *in the order they arise in the interpretation pipeline*. We do not want to ask a question about an ambiguity at one stage in the pipeline if there are still ambiguities remaining at an earlier stage. We therefore propose traversing the interpretation space a second time for the interpretations to be clarified; as soon as an ambiguity is generated, we should ask a question to resolve it. If ambiguities still remain after this question has been answered, we continue to traverse the search space for the remaining interpretations, and ask further clarification questions about points further on in the pipeline. For example, we might have three potential interpretations A, B and C. If B and C have the same syntactic analysis, but A's analysis differs, we ask a multiple-choice question about these two syntactic possibilities. If the user answers that the syntax is that of B/C, we need to look further in the pipeline. If we find that B and C have a presupposition resolved in different ways, we then ask an echo question to nail this remaining ambiguity.

The ambiguities that we find on this second pass can be thought of as those that we find to be ambiguous *in hindsight*, in the light of processing further on in the pipeline. Even though we are asking a question about syntactic ambiguity, we have actually worked out one or more full interpretations for each possibility. To be helpful to the user, the system could perhaps be configured to include information with a 'parenthetical' flavour in a clarification question, indicating what subsequent interpretation decisions follow as a corollary to the alternatives she is currently being asked about. For example:

(8)
User: The fruit flies like a banana.
System: Do you mean (1) 'The (fresh) fruit (in the bowl) flies like a banana,' or (2) 'The (mutant) fruit flies (in the genetics lab) like a banana'?

As described in section 5.1, we cannot be sure that the system's grammar will always be sufficient to form a syntactically unambiguous rephrasing such as *The drosophila are fond of a banana*. In these cases, parenthetical annotations of semantic information could be very useful.

Parse	Probability	Attachment	Saliency	Presuppositions	Accommodations	Dialogue act
P1		P1.A1				
		P1.A2				
P2		P2.A1				
P3		P3.A1				
		P3.A2				
		P3.A3				
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 2: Structure for aggregation of utterance disambiguation data in Te Kaitito

4 Disambiguation in Te Kaitito

Te Kaitito provides a variety of sources for disambiguation data: a DRS representation of the current discourse context, a saliency list of referents ranked by how recently they have been mentioned, and records of recent utterances and preferred parses. We also have a corpus of sentences hand-annotated with their correct parses.

As an utterance travels along Te Kaitito's processing pipeline, a **disambiguation table** (See Table 2) is progressively filled with all the information necessary for a disambiguation decision. When the utterance reaches the end of the pipeline, a disambiguation module uses this information to make a decision—either selecting an interpretation outright, or initiating generation of clarification questions if there is insufficient information for a clear decision.

The structure of the disambiguator is, broadly speaking, an iterative pruning process traversing the table from right to left. Each column is consulted in turn and sufficiently implausible parse/attachment combinations discarded. If at any stage only a single interpretation remains, it is chosen as the correct one; otherwise the next column to the left is used to prune the remaining parses. If multiple interpretations remain after the final stage (the stochastic grammar), clarification questions are generated (see Section 5).

This model fits the observation, made in Section 3.2, of prioritising information from higher semantic levels: the lower levels are only consulted as tie-breakers for the higher levels.

The exact mechanism for combining disambiguation is subject to some experimentation: in particular, it is not clear that presuppositional weight is always a more reliable indicator than saliency. However, the self-contained nature of the disambiguator lets us modify it without affecting the rest of the system.

4.1 Syntactic disambiguation using statistics

Probabilistic rule annotation using statistical data is an established technique for resolving syntactic ambiguity (Manning and Schütze, 1999, Chapters 11 and 12). Augmenting a context-free grammar with rule probabilities is relatively straightforward; the application of similar techniques to a Head-Driven Phrase Structure Grammar such as that used in Te Kaitito is a more complex issue (Brew, 1995).

We use the techniques discussed by Toutanova et al. (2002) for stochastic HPSG parsing: augmentation of the derivation trees with probabilities in the manner of a probabilistic CFG, and an expectation-maximization technique on selected features of the derivation tree.

To construct a stochastic grammar, we require a source of statistical data. In our case this takes the form of an annotated treebank. Using the LKB parser and the [incr tsdb()] package[2], we can parse a corpus of test sentences and manually select preferred parses. [incr tsdb()] stores the human annotator preferences as first-class data, making it relatively immune both to changes in the nature of the statistics extracted and in the underlying grammar.

4.1.1 Contextually augmented probabilities

Commonly, probabilities in stochastic grammars are static: they are inferred off-line from a corpus and remain fixed thereafter. It would clearly be desirable to adapt the probabilities to the current dialogue context. Consider ex-

[2][incr tsdb()] provides an integrated grammar development environment with a range of facilities for diagnostics, evaluation and benchmarking (see Oepen (1999) for details). Here we are mainly concerned with its facility for maintaining a database of test items, parses, and human correctness annotations.

ample 1: If *fruit flies like bananas* follows hard upon *meat flies rather inelegantly* and *vegetables fly like a dream*, then we would like to assign greater weight to the *fly-as-verb* reading. This can be done by treating the dialogue as an additional corpus, albeit with different learning parameters.

We propose augmenting the usual probability of a rule or feature with another value representing its 'weight' in the immediately preceding context. This value is then combined with the corpus-derived probability to give the overall probability used in disambiguation.

The contextual 'weight' of a feature is simply a counter which is incremented whenever the feature appears in an utterance by the user or the system. Heavy damping is applied at each dialogue turn so that features which stop appearing in the context soon regain their base probabilities. Careful tuning of the damping factor will be necessary.

It is not at present clear how best to combine the contextual weight of a feature with its base probability. Our current proposal is simple addition after scaling by an empirically tuned factor, but further evaluation will be needed.

4.1.2 The user model

The distribution of grammatical features varies with the speaker as well as with the context, both in vocabulary and higher-level constructs. Our dialogue system functions in a language learning environment, where users are learners of Māori. In this environment, there is even greater variation between users' idiolects, as learners often have widely differing skill levels.

For this reason it makes sense to augment the probabilistic model with per-user information. Again, the dialogue itself is used as an ancillary corpus, but in this case statistics are only extracted from the user's utterances, not those of the system. The results of this on-line learning can then be combined with the corpus-derived probabilities in the same way as the contextual counts. Again, some damping is desirable. Personal usage patterns don't change as fast as conversational topics, but they are subject to gradual variation, particularly when the speaker is learning a new language.

There are additional benefits to keeping track of a learner's usage patterns: they can be compared with a corpus of sentences at the learner's intended level to find gaps in their knowledge of a language. It may be useful to bias the system's generation system in favour of features that a learner's speech *lacks*, in order to give them more exposure to constructs that they find more difficult.

4.1.3 Question-answering and priming

In human conversation, even the most unlikely parse can become plausible when primed by a question with the appropriate syntactic structure. For example, a hugely improbable interpretation of the sentence *Matt cooks lunch* can be primed by prepending the question *What do matt cooks do when they get hungry in the middle of the day?*. We would like to be able to prime our system similarly.

This kind of priming can be incorporated in similar fashion to the context and user models, using a very short-term skewing of feature probabilities. When a question is asked, the probabilities of its features are substantially increased for the next dialogue turn only. In this case, *matt-as-adjective* and *cook-as-noun* would receive greater than normal weight.

This problem can also be considered at other levels of processing: at a semantic level, Te Kaitito will prefer the *cook-as-noun* interpretation if there are cooks in the current discourse context. At a dialogue act level, the uncommon interpretation can be preferred because it is the only one which answers the question. One advantage of incorporating this kind of priming at the syntactic level is efficiency: with a large grammar it might be necessary to prune less likely parses before the semantic stage of processing, even though this runs counter to our disambiguation technique. In this case it's vital to incorporate priming at the syntactic level, or the correct parse may be removed before the semantic and dialogue-act layers can perform disambiguation.

4.2 Semantic disambiguation using presupposition resolution

We have already described how our system uses information about the presupposition resolution process to generate preferences between interpretations: see Knott and Vlugter (2003). Here is a quick summary.

There are three possibilities to consider when resolving a presupposition: we may find no antecedents in the discourse context that match the presupposition, we may find exactly one antecedent that matches, or we may find more than one antecedent. If we cannot find any discourse entities to bind a presupposition to we can be generous and resolve this presupposition

by accommodating the information provided. If there is exactly one possible binding the presupposition is simply resolved. If there is more than one possible binding then we add further ambiguity to the intended meaning of an utterance as each possible binding can be considered a separate interpretation. After presupposition resolution each interpretation has its presuppositions resolved either through binding to some entity in the discourse context, or by accommodating the content of the presupposition, and the number of possible interpretations may have increased.

In disambiguation through presupposition resolution we keep to three principles. Firstly, we prefer interpretations that resolve through binding over those that resolve through accommodation. Secondly, when presuppositions are resolved through binding we prefer those with greater presuppositional content. Thirdly, we prefer interpretations where presuppositions are resolved to more salient entities in the discourse.

4.3 Dialogue act disambiguation

Dialogue act disambiguation is currently done procedurally. As noted in Section 3.2, it seems unlikely that a dispreferred dialogue act interpretation could ever be redeemed by high local scores on syntactic or presupposition-resolution grounds. In the context of a question, an assertion is checked to see if it can be interpreted as an answer to the question. If not, it is considered to be a new assertion (with the question being ignored). Note that it could be an answer that the system cannot interpret as such, because of some misunderstanding or the limitations of matching questions and answers. In the context of an ungrounded assertion, a question is first considered as a possible clarification question and then, if this fails, as a new query (with implicit grounding of the previous assertion). In dialogue act disambiguation we prefer interpretations that answer a question over those that ignore the question and assert new information, and so on.

5 Clarification questions in Te Kaitito

Te Kaitito currently generates two kinds of clarification question. These will be described in turn.

5.1 Multiple-choice questions

We generate all possible sentences realizing each remaining candidate interpretation, and then search for sentences that unambiguously present one interpretation. Here is an example:

(9)
User: The dog chased the sheep.
System: Do you mean (1) 'The sheep were chased by the dog,' or (2) 'The sheep was chased by the dog'?

There's no guarantee that we will find such a sentence for each interpretation. However, we can take certain measures to improve the chances of doing so. Firstly, we can adapt our grammar to include devices for resolving common ambiguities. For instance, we can include a lexical item *you (by yourself)*, to unambiguously signal a singular second-person pronoun. Naturally we do not want to generate this term *except* when we are nailing ambiguities. But if we use the probabilistic grammar to rank alternative sentences when generating, and we ensure that special constructions are rare in the test suite, we should ensure that they are only used in clarification questions. Secondly, our grammar is bilingual, so we can generate sentences that realize an interpretation in another language. For example:

(10)
User: Kia ora, e hoa mā.
System: Do you mean (1) 'Hello friends,' or (2) 'Hello, O white friend'?

This also increases the likelihood of paraphrases that successfully nail syntactic ambiguities.

5.2 Echo questions

We ask echo questions to resolve referential ambiguities generated during presupposition resolution. Example 4 demonstrates this. To be a bit more useful, the system can also generate multiple-choice alternatives:

(11)
User: The dog barked.
System: *Which* dog barked? The black dog? Or the white dog?

Finally, if ambiguities are encountered during the processing of questions, rather than produce a nested echo-question, the system simply provides an answer to all of the questions the user can be understood as asking.

(12)
User: Which dog chased the cat?
System: The black dog chased the white cat. The white dog chased the black cat.

6 Summary and further work

We have created a general framework within which to combine dialogue information from different domains and semantic levels, for the purpose of disambiguating user utterances in a dialogue system. We propose that disambiguation information be roughly prioritised according to its position in the processing pipeline: high-level semantic information carries more weight than low-level syntactic information. Clarification questions, if required, are generated in increasing order of semantic level. We also discuss a technique for augmenting a stochastic grammar with statistics drawn from the current dialogue context and from a particular user's dialogue history, giving it a better chance of selecting the correct parse in a given context.

The framework we describe is already in place, but many of the variables in the process (for example, the weighting and damping factors used to augment the stochastic grammar) still require some careful tuning. Further testing may also expose unforeseen interactions between levels, which may complicate the current straightforward iterative pruning algorithm; however, no changes to the framework itself should be necessary.

References

Chris Brew. 1995. Stochastic HPSG. In *Proceedings of EACL-95*.

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

J Ginzburg. 1996. Interrogatives: Questions, facts and dialogue. In S Lappin, editor, *The handbook of contemporary semantic theory*, pages 385–422. Blackwell, Oxford.

H Kamp, J van Genabith, and U Reyle. in preparation. Discourse representation theory. In *Handbook of Philosophical Logic*. Springer-Verlag.

A Knott and P Vlugter. 2003. Syntactic disambiguation using presupposition resolution. In *Proceedings of the 4th Australasian Language Technology Workshop (ALTW2003)*, Melbourne.

A Knott and N Wright. 2003. A dialogue-based knowledge authoring system for text generation. In *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, Stanford, CA.

A Knott, I Bayard, S de Jager, L Smith, J Moorfield, and R O'Keefe. 2002. Syntax and semantics for sentence processing in English and Māori. In *Proceedings of the 2nd Australasian Natural Language Processing Workshop*, pages 33–40, Canberra, Australia.

A Knott, I Bayard, and P Vlugter. 2004. Multiagent human-machine dialogue: issues in dialogue management and referring expression semantics. In *Proceedings of the 8th Pacific Rim Conference on Artificial Intelligence (PRICAI 2004)*, Auckland.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Wolfgang Menzel and Ingo Schröder. 1999. Error diagnosis for language learning systems. *ReCALL*.

Stephan Oepen. 1999. [incr tsdb()] competence and performance laboratory. user and reference manual. http://citeseer.ist.psu.edu/457852.html.

Birte Schmitz. 1997. Collaboration in automatic dialogue interpreting. In *Proceedings of the Workshop "Collaboration, Cooperation and Conflict in Dialogue Systems", IJCAI-97*, pages 79–88.

Kristina Toutanova, Christopher D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263.

Maximum Entropy Markov Models for Semantic Role Labelling

Phil Blunsom
Department of Computer Science and Software Engineering
University of Melbourne, Vic 3010, Australia
email: pcbl@cs.mu.oz.au

Abstract

This paper investigates the application of Maximum Entropy Markov Models to semantic role labelling. Syntactic chunks are labelled according to the semantic role they fill for sentence verb predicates. The model is trained on the subset of Propbank data provided for the Conference on Computational Natural Language Learning 2004. Good precision is achieved, which is of key importance for information extraction from large corpora containing redundant data, and for generalising systems beyond task specific, hand coded template methods.

1 Introduction

In recent years, much progress has been made in the fields of information extraction and question answering. Research systems developed for conference competitions have graduated into the commercial world in such applications as air travel information and booking, call handling, and banking. Although these systems perform well on their chosen tasks, they are generally based on a frame–and–slot approach. This approach uses application–dependent frames defined for propositions and then attempts to fill slots from words surrounding the proposition that triggered the frame. For example, in a financial system we may be interested in extracting company mergers from newswires. We could define a frame for the verb stem *merge*, and slots in that frame for the companies involved in the merger. Although this could be effective for the chosen domain, each time we want to develop a system for a new domain we need to start from scratch. In order to build broad coverage systems capable of generalising, we need a way of defining and labelling propositions and their arguments that is not tied to a particular application.

Systems have been developed to address the task of semantic role labelling (SRL) using a variety of machine learning techniques and features ranging from simple lexical information to those derived from parse trees. These semantic roles identify arguments of verb predicates and can be general, such as *agent*, *theme*, or verb specific, such as *A0*, *A1*.

```
<!DOCTYPE frameset SYSTEM "frameset.dtd">
<frameset>
<predicate lemma="begin">
<roleset id="begin.01" name="start" vncls="55.1">
<roles>
  <role descr="Agent" n="0">
    <vnrole vncls="55.1" vntheta="Agent"/></role>
  <role descr="Theme(-Creation)" n="1">
    <vnrole vncls="55.1" vntheta="Theme"/></role>
  <role descr="Instrument" n="2"/>
</roles>
```

Figure 1: Propbank XML frame for *begin*

This paper presents a discriminative Markov model for the semantic role labelling task proposed for the Conference on Computational Natural Language Learning 2004 (CoNLL). Firstly, recent work on SRL is examined and the data available for this task is presented. A discriminative maximum entropy framework is defined and a SRL model described. This model trains maximum entropy classifiers for each state in the Markov model, representing the probability distribution for transitions from that state for a given observation feature vector. This approach combines the advantages of transition sequence based models with the ability of maximum entropy classifiers to handle a diverse range of overlapping features. The results obtained achieve precision comparable to the best performing support vector model, while requiring significantly less time to train. However, recall is not as high as other approaches and provides an area for further improvement.

2 Background

Many researchers have tackled semantic role labelling. Traditional parsing systems have performed tasks that incorporate a level of semantic labelling and many information extraction systems attempt to solve the labelling problem for a single,

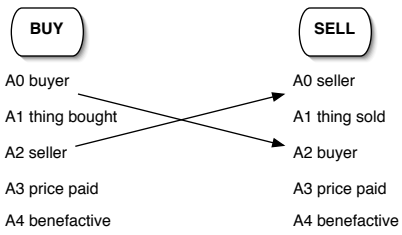


Figure 2: Propbank frames for two related predicates

or small group of propositions, e.g. the biomedical domain. Recently, the development of semantically labelled corpora has led to a number of statistical systems being developed.

Following the pattern of other areas of NLP, it was not until the development of projects to systematically label propositions and their arguments in a corpus, that researchers began to attack the problem of developing generalised statistical systems for SRL. FrameNet was one of the first such projects and aimed to create a hierarchy of semantic frames describing predicates and the roles they accept. Sentences from the British National Corpus (BNC) where annotated with roles derived from these frames.

Gildea and Jurafsky (2002) used FrameNet to train networks of statistical classifiers based on a variety of lexical and syntactic features. They used the statistical parser of Collins (1997) to create the parse trees upon which their syntactic features were based. As the BNC doesn't contain gold standard parse trees, they were unable to quantify the affect of an imperfect parse on the classifiers. However, this work served to highlight the two distinct tasks in SRL: segmenting argument constituents and identifying their semantic role. On the full task of segmenting and labelling arguments, Gildea and Jurafsky achieved a performance of 65% recall and 61% precision, but it must be noted that their system assumes knowledge of the proposition arguments being labelled. Fleischman et al. (2003) applied maximum entropy techniques to the problem defined by Gildea and Jurafsky and achieved a slight increase in performance. This work cast the labelling task as one of tagging, using a maximum entropy formulation over eleven feature sets, and the Viterbi algorithm to search for the best tag sequence. Thompson et al. (2003) applied a generative model to labelling semantic roles in FrameNet data which was somewhat similar to a hidden Markov model (HMM) approach. Their model calculated the probability that for a particular frame a role sequence would generate an observed constituent sequence and assumed that roles correspond to sentence constituents. On their described task they record approximately 70% accuracy, but it is not clear how their system could be applied to the case of general SRL given a raw sentence with no additional information.

The Proposition Bank is a project to add propositional annotations to the Penn Treebank corpus (Kingsbury et al., 2002). In comparison to FrameBank, Propbank labels roles with a generic set of tags: *A0*, *A1*, *A2* etc. Frame entries map these generic labels to proposition specific semantic roles, such as theme and agent (figure 1). Although the semantic roles are not strictly comparable across propositions, annotators were instructed to be consistent in naming and numbering semantic related verb roles (figure 2). The more statistically representative approach and increased coverage, as well as the availability of the treebank parse trees, has made Propbank the most popular resource for current SRL research. Pradhan et al. (2003) developed a support vector machine that chunked semantic arguments from sentences in the Propbank corpus. Their system used similar features as previous systems, including parse trees, and explored using both word-by-word and chunk-by-chunk instances for the classifier. Of interest in this work is an investigation into the effect of not using features derived from a parse tree and relying only on shallow syntactic information, they found that this reduced performance by around 20%.

In an attempt to motivate the development of SRL systems that have direct application to current problems in information extraction and question answering, CoNLL made SRL the topic of their 2004 shared task (Carreras and Marquez, 2004). The challenge was to come up with "machine learning strategies which address the SRL problem on the basis of only partial syntactic information, avoiding the use of full parsers and external lexico-semantic knowledge bases," thus recognising the importance of SRL techniques that are fast and domain independent. The data provided for the conference was based on Propbank and didn't contain parse trees. Of the systems developed for CoNLL, two used maximum entropy techniques. Both these systems used a single instance–by–instance maximum entropy classifier. Lim et al. (2004) achieved the highest performance of the two with an F score of 64.76 on the test data. In this paper we aim to improve precision over previous maximum entropy techniques by optimising the whole sentence tag sequence and thus reducing sequence errors such

as repeated arguments and unlikely argument orderings.

3 Data & Evaluation

The CoNLL shared task supplied training, development and testing data created from Propbank annotations on six sections of the Wall Street Journal component of the Penn Treebank. The standard semantic labels from Propbank were used:

Verb specific arguments Arguments with a specific semantic meaning for a verb are labelled *A0-A5*. The semantics of the roles corresponding to these numbered arguments are defined in the Propbank frame for the predicate verb, but in general *A0* maps to agent and *A1* to patient or theme.

Adjunctive arguments General arguments that any verb may take. These include *AM-LOC*, for locative and *AM-TMP*, for temporal. For the complete list see (Carreras and Marquez, 2004).

Argument references Predicate arguments that reference other predicate arguments. Labelled *R-A?* with the argument referenced as the suffix, eg. *R-A1* is a reference to the *A1* defined elsewhere.

Predicate verb The predicate verb that defines the proposition being labelled is tagged *V*.

The data contained annotations of part-of-speech (PoS), base-phrase chunks, clause embedding and named entities. An example of the data is shown in figure 3. For each of the target verbs a column is provided with the argument labelling for that verb.

PoS and base–phrase chunks are annotated in IOB2(inside, outside, begin) format (Ramshaw and Marcus, 1994) and don't allow embedding. Clauses and arguments are annotated in begin, end format, with clauses allowing overlapping and arguments not. So *(A0** represents the start of the *A0* argument and **A0)* represents the end, while *(A0*A0)* labels an argument spanning a single word.

CoNLL also developed the evaluation script *srl-eval.pl* that ranks systems on the standard criteria of *precision*, *recall* and F1 score.

4 Maximum Entropy Markov Models

The system described in this paper is based on a discriminative Markov model, allowing both the optimisation of the tag sequence and the incorporation of multiple features over observations. A limitation of HMMs is that it is hard to extend them to allow multiple features of observations, rather than atomic observations themselves. An alternative to the HMM was proposed by McCallum et al. (2000) in which the transition and observation probability matrices are replaced by maximum entropy classifiers for each state. These classifiers encode the probability distribution $P_{s'}(s|o)$, the probability of making the transition to s from s' and observing o.

4.1 Conditional Exponential Transition Model

The maximum entropy framework, as presented by Berger et al. (1996), aims to "model all that is known and assume nothing about that which is unknown." This is achieved by choosing the model that fits all the constraints expressed by the training data and is the most uniform, i.e. the one with the highest entropy.

Many classification tasks are most naturally handled by representing the instance to be classified as a vector of features. By combining the state and observation transition functions into a single maximum entropy model for each state we can condition the tag sequence assigned to a sentence on such things as part–of–speech tags, phrasal tags, predicate verbs, etc.

For this work we represent features as binary functions of two arguments, $f_i(o, s)$, were o is the observation and s is the possible next state. In order to encode properties of instances which are not binary, such as part–of–speech, a binary feature function is defined for each possible value of the property, i.e. $f_0(o, s)$ is true *iff* o contains the tag *NN* and a transition to s is observed, $f_1(o, s)$ is true *iff* o contains the tag *VB* and a transition to s is observed, and so on.

Given our set of feature functions, and a set of labelled training instances, we can formulate the following constraint equation:

$$expected[f_i] = empirical[f_i] \qquad (1)$$

i.e. we should aim to have the expected value for the feature function i in the predicted distribution equal to its average on the empirical training sequence. The maximum entropy distribution is a conditional exponential model of the form:

$$P_{s'}(s|o) = \frac{1}{Z(o, s')} \exp(\sum_i \lambda_i f_i(o, s)) \qquad (2)$$

where λ_i are the feature weights that need to be estimated from the training data, and $Z(o, s')$ is a normalisation factor to ensure P is a probability distribution.

```
That            DT    B-NP    (S*         -              (A0*              *
settlement      NN    I-NP      *         -                 *A0)           *
represented     VBD   B-VP      *         represent        (V*V)           *
the             DT    B-NP      *         -              (A1*              *
first           JJ    I-NP      *         -                 *              *
time            NN    I-NP      *         -                 *              *
shareholders    NNS   B-NP    (S*         -                 *          (A2*A2)
were            VBD   B-VP      *         -                 *              *
granted         VBN   I-VP      *         grant             *           (V*V)
a               DT    B-NP      *         -                 *          (A1*
major           JJ    I-NP      *         -                 *              *
payment         NN    I-NP      *         -                 *              *A1)
in              IN    B-PP      *         -                 *       (AM-LOC*
a               DT    B-NP      *         -                 *              *
greenmail       NN    I-NP      *         -                 *              *
case            NN    I-NP      *S)       -                 *A1)           *AM-LOC)
.               .     O         *S)       -                 *              *
```

Figure 3: CoNLL data format

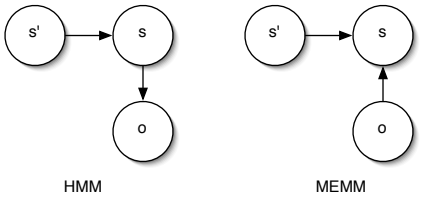


Figure 4: In a MEMM states are conditioned on the previous state and the observation.

4.2 Training and Evaluating the Model

In order to train the MEMM we first need to split the global training data into subsets which will be used to train each individual maximum entropy model. The list of training instances for each state represents the transitions and observations made from that state in the training data. Once all the training data has been processed into sub training lists, the Generalized Iterative Scaling (GIS) (Darroch and Ratcliff, 1972) algorithm is used to train each maximum entropy classifier.

A requirement of natural language systems, especially those based on tagging sentences, is that the sequence of classifications produced by a model should be coherent, this is what the HMM approach and the Viterbi dynamic programming algorithm facilitate. The recursive Viterbi step for the MEMM is defined as:

$$\delta_t(s) = \max_{0 \leq s \leq N} [\delta_{t-1}(s') \times P_{s'}(s|o_t)] \qquad (3)$$

where $\delta_t(s)$ is the probability of seeing the observation sequence up until time t and being in state s, having followed the most probable sequence of state transitions into s. Figure 4 illustrates that both the previous state and the observation determine the next state probability distribution.

4.3 Limitations

The MEMM approach has limitations that must be kept in mind for any implementation. By splitting the training data on the basis of state transitions we are removing the ability of the model to make global generalisations over particular semantic role properties. For example, the properties that mark the start of a particular role will be dispersed among all the states that transition into that role. Therefore the fact that a role is often realised as a prepositional phrase with its first word *on* will be diluted in the split training data. As an added problem, some roles will only appear very rarely in the training data and thus their state transition functions will need to be estimated from a very small amount of data, without the ability to represent general role properties represented in the global data. In order to partially address the problem of data sparseness, a gaussian prior is used to smooth the individual transition functions which have less than 10,000 [1] supporting training examples.

Lafferty et al. (2001) identified the label bias problem as a potential concern for MEMMs. If a particular state has a low–entropy next state distribution, with the extreme case being a single next state, then the observation that the transition is conditioned on will effectively be ignored. Thus the Viterbi path will be biased towards state transitions with low entropy that may be supported by very little training data, over other transitions which are much more supported in the training data. As the MEMM developed in this paper was trained assuming a fully connected initial structure, the transition functions were unlikely to contain a single next state with a nonzero probability. Thus, it was assumed that the label bias problem had little effect, but this could be verified experimentally in further work.

[1] Determined experimentally. A discussion of smoothing maximum entropy models can be found in (Chen and Rosenfeld, 1999).

5 A MEMM for SRL

Section 3 described the framing of SRL as a tagging task. As MEMMs allow the determination of an optimum tagging sequence for a sentence, and allow the modelling of the data as multiple features, it is of interest to investigate their performance on the CoNLL SRL labelling task.

The MEMM described in this paper is a phrase-by-phrase model that maps argument labels to states, and feature vectors over phrases to observations. The original data format displayed in figure 3 was mapped into a phrase based format as shown in figure 5 by converting role boundaries to IOB2 representation (Ramshaw and Marcus, 1994) and collapsing each phrase into a single instance represented by its head word.

Tagging phrase–by–phrase leads to the loss of some information as it is no longer possible to represent roles that have sub-phrasal boundaries. However, as such roles are rare, or easy to handle with post-processing, the loss is acceptable. The advantage of this representation is that it compresses the data and thus decreases the processing required by the model.

A further verb specific compression is performed on the data clauses. It was observed that phrases in clauses below the target verb clause participate completely in roles, therefore these clauses were collapsed with only the first phrase retained for lexical and syntactic information, as shown in figures 6 and 7.

The following features were calculated for each of the phrase/clause instances presented to the model as an observation vector:

Syntactic features Phrase type, head word, head PoS were used. Also, for context, the two preceding and following phrase types and head PoS.

Clause delta The difference of the clause depth of the instance phrase and the clause containing the target verb. If the phrase is lower in the clause tree than the target verb, this value is negative.

Predicate clause A boolean feature that is *true* if the instance clause depth is equal to the target verb phrase depth, *false* otherwise.

Position relative to the verb A feature to indicate whether the phrase instance is before, in, or after the target verb phrase.

Target verb stem The stemmed target verb, as supplied in the CoNLL input data.

Predicate verb suffix Regular expression suffix matching is performed on the target verb. Suffixes matched are: *ing*, *ogy*, *ed*, *s*, *ly*, *ion* and *ies*.

Most occured frame The frame of a verb is defined as the role sequence that it appears with in a sentence, excluding adjunctive roles, for example: *(A0, V, A1, A2)*. A frequency distribution of frames that a target verb appears in is created from the training data, and the most frequent frame for a particular target verb presented as a feature. This aims to represent a verbs preferred sentential structure.

Number of NPs from the target verb A count of the number of NP chunks between the instance phrase and the target verb phrase. If the instance is after the verb, this is a negative count.

Number of base-phrases to the target verb A count of the number of base-phrase chunks between the instance phrase and the target verb phrase. If the instance is after the verb, this value is negative.

Target verb voice A heuristic is used to estimate the voice of the verb as either *active* or *passive*. If the verb phrase contains a form of "to be" and the verb is not gerundive, it is labelled *passive*, otherwise it's labelled *active*.

Prepositional head All noun phrases following a prepositional phrase used a feature encoding the prepositional phrase's head word. This improved the handling of adjunctive arguments realised as prepositional phrases.

Clause As described previously, clauses below the verb are compressed to a single instance. If this instance phrase is a compressed clause, this feature is *true*.

Feature pairs The following features were paired to provide an indication of dependency: *verb stem + head*, *verb voice + position*, *verb stem + phrase type* and *verb stem + phrase path*.

6 Implementation

The model was implemented in the Python scripting language using the Natural Language Toolkit (NLTK) (Bird and Loper, 2004). The system was implemented as a pipeline of processes:

1. First the training data was tokenized and global statistics calculated, such as verb frames etc.

```
settlement      NN    NP        -               B-A0          O
represented     VBD   VP        represent       B-V           O
time            NN    NP        -               B-A1          O
shareholders    NNS   NP        -               I-A1          B-A2
granted         VBN   VP        grant           I-A1          B-V
payment         NN    NP        -               I-A1          B-A1
in              IN    PP        -               I-A1          B-AM-LOC
case            NN    NP        -               I-A1          I-AM-LOC
.               .     O         -               O             O
```

Figure 5: Phrase-by-phrase compressed data format

```
settlement      NN    NP        -               B-A0
represented     VBD   VP        represent       B-V
time            NN    NP        -               B-A1
shareholders    NNS   NP        -               I-A1   (a compressed clause)
.               .     O         -               O
```

Figure 6: Data representation with a clause compressed for the verb *represent*

2. The data was then separated into phrase/clause chunks and the features determined. It was then in the form of training instances, an instance corresponding to a sentence and target verb pair, that could be used as training data.

3. Individual state transition function training lists were then created as described in section 4.2

4. The GIS algorithm with a Gaussian prior was then used to train each state transition function

Testing instances were labelled by the model using the Viterbi algorithm described in section 4.2 to determine the optimal tagging. A post processing step was used to tag *AM-MOD* and *AM-NEG* adjuncts as these occur at a lexical level and thus can't be correctly classified by a phrase-by-phrase system. All words in the target verb phrase that had PoS tags *MD* (modal auxiliary) were tagged as *AM-MOD*, while occurrences of the words *n't* or *not* were tagged as *AM-NEG*.

Table 1 shows the output of the *srl-eval.pl* script when the model was used to tag the test data. The overall F_1 score achieved by the system was 59.09. This score corresponds to 24% of the sentences being perfectly labelled.

Due to the large amount of time required to tag the development and test sets using a full set of role tags, the tagging was performed with a reduced set of tags. The tags used were: *A0*, *A1*, *A2*, *A3*, *A4*, *A5*, *R-A0*, *R-A1*, *R-A2*, *C-A1*, *AM-TMP*, *R-AM-TMP*, *AM-ADV*, *AM-LOC*, *AM-MNR*, *AM-MOD*, *AM-DIR*, *AM-NEG*, *AM-DIS*, *AM-CAU*, *AM-EXT*, *AM-PNC*, and *V*. This accounts for some of the reduced performance on the test set as the the reduced tag set was chosen on the basis of the development data, and thus had less coverage on the test data.

7 Discussion

The motivation for using a MEMM approach was to produce more coherent output by finding the optimal Markov sequence, and this is borne out in the results. As is to be expected when not using a parse, prepositional phrase attachment and other syntactic ambiguity, such as propositions late in long sentences, contribute to a number of errors.

Figure 8 shows two sentences that were correctly tagged by the system, and three that contained errors. We can see from the correct sentence that the model is capable of recognising some of the more complex roles such as *discourse (DIS)* and *purpose (PNC)*. In the third sentence it is apparent that the system has not recognised *January* as temporal, instead labelling the preposition as *locative (LOC)*. This is a common error, *LOC* and *TMP* arguments are both frequent and occur in similar prepositional syntax and the model has not been able to capture the lexical distinctions between the two. Sentences four and five show that the system struggles with the semantic ambiguity of the *LOC* role. In four, the gold standard identifies *'in the Reagan administration'* as the location where the defense secretary served, whereas the model tags the preposition as part of the theme of *served*, a tagging which seems quite reasonable. The opposite occurs in sentence five where the model has tagged *'in real estate'* as a location. In this case the gold standard is more semantically consistent. Overall these errors support the argument that even when the model errs, the output is still often consistent.

The CoNLL data requires the labelling of predicate verbs that occur within noun phrases. The model developed generally fails to handle these predicates correctly. This is to be expected as these predicates have a completely different structure and behavior to the standard verb phrase predicates and

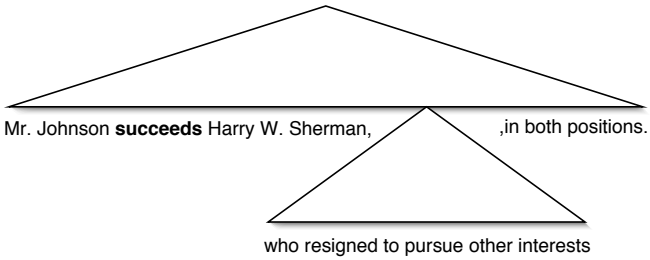


Figure 7: A clause tree showing a clause that would be collapsed to a single instance

1. [Of course]$_{DIS}$, [Mr. Wolf, 48 years old,]$_{A0}$ [has]$_{V}$ [some savings]$_{A1}$.
2. Mr Johnsom succeeds [Harry W. Sherman]$_{A0}$, [who]$_{R-A0}$ [resigned]$_{V}$ [to pursue other interests]$_{PNC}$, in both positions.
3. ([In January]$_{LOC}$)^{TMP}, [he]$_{A0}$ [accepted]$_{V}$ [the position ...]$_{A1}$.
4. [Mr. Carlucci, 59 years old,]$_{A0}$ [served]$_{V}$ as [defense secretary (in the Reagan administration)^{LOC}]$_{A1}$.
5. [Balcor]$_{A0}$, [which]$_{R-A0}$ [has]$_{V}$ ([interests]$_{A1}$[in real estate]$_{LOC}$)^{A1}, said the position is newly created.

Figure 8: Example model output. Square brackets indicate tags applied by the system, while round brackets and indicate the tagging from the gold standard

	Precision	Recall	$F_{\beta=1}$
Overall	71.29%	50.45%	59.09
A0	84.02%	66.11%	74.00
A1	65.94%	51.81%	58.03
A2	58.96%	42.92%	49.68
A3	65.15%	28.86%	40.00
A4	70.00%	42.00%	52.50
A5	0.00%	0.00%	0.00
AM-ADV	46.75%	11.76%	18.80
AM-CAU	66.67%	4.08%	7.69
AM-DIR	45.16%	28.00%	34.57
AM-DIS	66.99%	32.39%	43.67
AM-EXT	70.00%	50.00%	58.33
AM-LOC	37.41%	22.81%	28.34
AM-MNR	47.96%	18.43%	26.63
AM-MOD	87.57%	92.26%	89.86
AM-NEG	82.96%	88.19%	85.50
AM-PNC	35.29%	7.06%	11.76
AM-PRD	0.00%	0.00%	0.00
AM-TMP	61.23%	26.64%	37.13
R-A0	90.65%	61.01%	72.93
R-A1	73.91%	48.57%	58.62
R-A2	75.00%	33.33%	46.15
R-A3	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-PNC	0.00%	0.00%	0.00
R-AM-TMP	100.00%	7.14%	13.33
V	96.63%	95.69%	96.16

Table 1: Model performance on the test set

are also in the minority in the training data. A separately trained model could solve this problem.

There are a number of possible improvements to the MEMM model. Prepositional phrase attachment could be handled explicitly in a subsystem. Each state transition function could induce the n most informative features and discard the rest, enabling the model to be more able to avoid over–training problems. Another obvious path to explore would be the application of conditional random fields (Lafferty et al., 2001) to the SRL task. These models have the advantage of solving the label bias problem, as well as allowing more flexibility in modelling, and by using an MEMM for initial weights they converge more quickly.

The advantage of the approach described lies in the increase in precision and the improved coherence of the sentence tag sequence. However, the nature of the SRL task as one that requires both local phrase level feature information and global sentence information, would indicate that both need to be modelled for maximal performance. Most current approaches, including the approach presented in this paper, use word or phrase level classifiers and then try to capture sentence level information through features such as phrase paths and counts.

The evaluation metrics chosen for the CoNLL shared task appear to be brittle, giving no recognition to systems that come close to predicting arguments but fail to find exact matches. The main motivation for SRL techniques is to improve information

extraction systems, but it is possible that systems that are able to accurately label the head–words that fill roles of predicate verbs, but miss prepositions and additional noun-phrases, would be very useful also. This is less of a concern for systems employing parse trees as phrase attachment information is known.

8 Conclusion

A maximum entropy Markov model was developed for the SRL task defined at CoNLL 2004. The model managed good performance on the test data, achieving a precision of 71.29 and an F_1 score of 59.09. This result suggests that discriminative sequence models are worth further investigation for the semantic role labelling task. Many errors in tagging made by the model can be attributed to lack of information about the syntactic relationships among phrase chunks, providing an argument for the use of full parse trees when labelling semantic roles.

Acknowledgements

The author would like to thank his research supervisor Steven Bird for his comments and feedback on this work and also the reviewers for their helpful comments. The research undertaken for this paper was supported by an Australian Postgraduate Award scholarship.

References

A. Berger, S. Della Pietra, and V. Della Pietra 1996 A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

S. Bird and E. Loper 2004 NLTK: The Natural Language Toolkit *Proceedings of the ACL demonstration session*, Barcelona, Spain.

X. Carreras and L. Marquez. 2004. Introduction to the CoNLL–2004 Shared Task: Semantic Role Labeling *Proceedings of CoNLL–2004 Shared Task*.

S. Chen and R. Rosenfeld 1999 A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing *Proceedings of 35th Annual Meeting of the ACL*, Madrid, Spain.

J. Darroch and D. Ratcliff 1972 Generalized Iterative Scaling for Log-linear Models. *Annals of Mathematical Statistics*, no. 43, 1470–1480.

M. Fleischman, N. Kwon and E. Hovy. 2003. Maximum Entropy Models for FrameNet Classification *Proceedings of EMNLP–2003*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles *Computational Linguistics*, 28(3):245–288.

P. Kingsbury, M. Palmer and M. Marcus. 2002. Adding semantic annotation to the Penn Treebank *Proceedings of the Human Language Technology Conference*, San Diego, CA.

J. Lafferty, A. McCallum, and F. Pereira. 2001 Conditional random fields: probabilistic models for segmenting and labeling sequence data. *Proceedings of the International Conference on Machine Learning*.

J. Lim, Y. Hwang, S. Park, and H. Rim. 2004 Semantic role labeling using maximum entropy model. *Proceedings of CoNLL-2004*.

A. McCallum, D. Freitag and F. Pereira 2000 Maximum entropy Markov models for information extraction and segmentation. *Proceedings of ICML 2000* , 591–598. Stanford, California.

S. Pradhan, K. Hacioglu, W. Ward, J. Martin and D. Jurafsky Oct 2003 TR-CSLR-2003-03: Shallow semantic parsing using support vector machines *Center for Spoken Language Research, University of Colorado*.

L. Ramshaw and M. Marcus 1994 Text Chunking Transformation Based Learning *Proceedings of the 3rd ACL Workshop on Very Large Corpora*.

C. Thompson, R. Levy and C. Manning. 2003. A Generative Model for Semantic Role Labeling *Proceedings of ECML 2003*.

A New Measure for Extracting Semantically Related Words

Yuanyong Wang and **Achim Hoffmann**
wyy@cse.unsw.edu.au
achim@cse.unsw.edu.au
School of Computer Science and Engineering
University of New South Wales
Sydney, 2052, NSW
Australia

Abstract

The identification of semantically related terms for a given word is an important problem. A number of statistical approaches have been proposed to address this problem. Most approaches draw their statistics from a large general corpus. In this paper, we propose to use specialized corpora which focus strongly on the individual words of interest. We propose to collect such corpora through targeted queries to Internet search engines. Furthermore, we introduce a new statistical measure, *Relative Frequency Ratio*,tailored specifically for such specialized corpora. We evaluated our approach by using the extracted related terms to attack the target word selection problem in machine translation. This type of indirect evaluation is conducted because a direct evaluation on the set of related terms thus extracted relies heavily on direct human involvement and is not quantitatively comparable to others' results. Our experimental results so far are very encouraging.

1 Introduction

The identification of semantically related words from texts is an important problem in natural language processing. If successfully identified, they could be used in query expansion, word sense disambiguation, as well as document classification (Tomohiko Sugimachi and Matsuo, 2003). Another application concerns the identification of new word senses in specialized languages, which are constantly evolving and, hence, no up-to-date dictionaries exist that could cover all those word senses. Many approaches, such as co-occurrence statistics based on mutual information (Church and Hanks, 1990), the Z-score (Tomohiko Sugimachi and Matsuo, 2003), have been used in the past to tackle this problem. These approaches are limited to be used only on general corpora in which large amounts of texts are collected from sources as diverse as possible. In this paper, we call this kind of corpus a General Corpus (GC). The nature of these measures (rate high co-occurrence high and rate high frequency words low) requires generality of the corpus. Generality is defined in our paper as for a corpus not being biased toward any particular domain or particular word. Mutual information is defined as follows:

$$I(w_1, w_2) = log\frac{P(w_1, w_2)}{P(w_1)P(w_2)},$$

where $P(w_i)$ is the probability of word w_i to occur in a document and $P(w_i, w_j)$ is the probability of both words w_i and w_j to occur in a document.

If a good corpus with high generality could be obtained, the co-occurrence statistics collected from the corpus could be very good and reflect accurately the tendency of semantic association among words. But there are several innate drawbacks to these GC based approaches. Firstly, generality is often very difficult to define. Secondly, acquiring a GC with good generality is a even more difficult problem. Thirdly, given that these above two problems are properly addressed the set of related terms extracted from the corpora are still limited in number. The main reason behind the third drawback is actually the generality required by these approaches. Since no words have a particularly high frequency in the corpus. It is not difficult to prove that the number of semantically related terms extracted for a given word is usually very low, especailly if we take away function words (which co-occur with any word indiscriminately). These sets of extracted related terms, if used to match the new context of the word in question, only provide very limited disambiguating power.

Based on the above analysis, another type of corpus is needed to obtain a sufficiently large set of related terms(so that they are practically useful) for any particular word. We call this type of corpus a Word Specific Corpus (WSC). It is constructed by collecting only the texts where the particular word of interest is present. We call this word the *seed word* and in formal contexts we refer to a WSC with seed word s, by $wsc(s)$. In such corpora words which occur with low frequency in a GC may

well occur with high frequency. We call this phenomenon frequency jump. Note, frequency jump of a word is a concept based on its frequency difference between a GC and a WSC where the word occurs. For example, the word "cell" is a low frequency word in GCs, but in a WSC with the seed word "tissue", it becomes a word with very high frequency. Frequency jump is very common to WSCs. In situations like this mutual information or similar measures would not properly reflect the semantic closeness between strongly related terms. Words like "cell" and the word "tissue" in the above corpus would be assigned dramatically lower mutual information value because of their high frequency. "Mutual informtion is widely known biased towards the word frequency. The tendency of mutual information does not depend on word semantics and the kinds of corpora but only on word frequency. This causes a problem in extracting the related words of a given word using an appropriate threshold value. Most of the extracted words are low frequency words and middle frequency words are rarely extracted" (Tomohiko Sugimachi and Matsuo, 2003). The Z-score measure is then proposed in the work to help extracting more middle frequency words. But from WSCs it is those high frequency words (apart from function words) that are supposed to be extracted, which makes the applicability of mutual information even Z-score worse on WSCs. Our proposed approach is designed to better measure the semantic closeness between words in WSCs and the seed word.

In summary of the above: for the set of extracted related terms to be practically useful, it has to be sufficiently large; GCs could not provide these sufficiently large sets of terms; WSCs are thus required; Mutual information and the like measures do not work well for WSCs; We propose a new approach that works well with WSCs.

The paper is organised as follows. In section 2 we theoretically present our new method for extracting related words. This is followed in section 3 by putting it into practice and showing some of the words thus extracted. In section 4, we evaluate the measure indirectly by using the extracted related words to attack the target word selection problem in machine translation. In section 5 we compare our results to those of related works. Section 6 contains the conclusions.

2 The Relative Frequency Ratio Measure

In this paper, a new measure called Relative Frequency Ratio (RFR) is proposed to extract semantically related words from WSCs. It is based on the idea that in a given context (e.g. a sentence) surrounding a seed word some words are semantically close to the seed word, others are not. For example "information" in the sentence "We have full information on all sorts of tissue paper." is not semantically as close to "tissue" as it is to "algorithm" in the sentence "The information gain is made as large as possible by this machine learning algorithm."

It is observed that in WSCs the closer a word is semantically to the seed word the larger its frequency jump (frequency increase) would be. This provides a natural measure for the semantic closeness between an arbitrary word and the seed word. The Rative Frequency Ratio, despite being based on the same spirit, is defined in a more formal way. First of all, relative frequencies for a word in both a GC and a WSC are computed. The ratio is then computed by dividing the GC relative frequency by the WSC relative frequency. In essence relative frequency ratio is a normalized version of frequency jump.

The relative frequency ratio (RFR) for a word is given by:

$$RFR_{w(wsc(s))} = \frac{f_{w(wsc(s))}/t_{wsc(s)}}{f_{w(gc)}/t_{gc}}$$

$RFR_{w(wsc(s))}$ is the ***relative frequency ratio***. $t_{wsc(s)}$ is the ***total number of word tokens*** in the corpus ***wsc(s)***. t_{gc} is the ***total number of word tokens*** in the corpus ***gc***.

For example, one of the seed words that have been tested with this approach is "tissue". A large GC in English is compiled. A WSC is also compiled with the seed word "tissue". The relative frequency for "paper" in the GC is 0.000477 and is 0.00322 in the WSC . The RFR value is 6.75. Another two words "end" and "open" that have almost same GC relative frequency as "paper" have drastically lower RFR values with respect to the WSC. The GC relative frequency and WSC relative frequency for "end" are 0.00048 and 0.000215 respectively. Its RFR value is 0.45. Similarly, the word "open" has 0.000477 and 0.000136 as its GC relative frequency and WSC relative frequency. The RFR value is 0.29. While "Paper" occurs almost 7 times more frequently in the WSC than in the GC, both "end" and "open" occur a lot less in the WSC. In this particular case, by setting a threshold of 1 for the RFR value would easily rule out the two words "open" and "end" and keep only the word "paper" as semantically close to the seed word "tissue".

	English	German
Number of Word Tokens	1,538,152	896,413
Number of Word Types	38,508	43,449

Table 1: General corpora for English and German.

3 Extraction of Semantically Related Words

3.1 Corpora

Two GCs, one English and one German general corpus, have been compiled to provide the base relative frequency statistics. GCs are compiled by issuing a set of most frequent function words as queries and extracting all the texts from the search results. This helps to avoid any possible domain bias being introduced by the compiling process because those most frequent function words are themselves not biased toward any domain. A number of WSCs in English and German have been compiled to provide the specialized relative frequency statistics for the English seed words and their German translations. Training data(WSCs) as well as testing data are collected for three English seed words "tissue", "apron" and "attack" respectively from the top n retrievals of Google and other search engines with the seed words as queries. From each document retrieved by the search engine only three sentences surrounding or containing the first occurrence of the seed word are extracted. This is mainly based on the assumption that the publisher usually tries to give as much as possible semantic information at the word's first occurrence to restrict its sense. This assumption has shown to be valid in our experiments. The data collection is summarised in Table 1 and Figure 1. The last column in Figure 1 provides some extra information about experiment results, which is better to be read together with final results summarized in Table 2 when they are discussed in Section 5.

3.2 Identifying related words

All three words have been analyzed with the RFR measure. The threshold on the measure is experimentally set to 1, which means any word with a RFR value higher than 1 would be extracted as semantically related to the seed word. In reality it might not be optimal, the threshold could vary when the sizes of the GC and the WSC are changed, or when the generality of the GC and the desired skewness of the WSC are changed. However the basic trend where semantically closer words have higher RFR values prevails.

A sample of the extracted words (word stems actually) is shown in Figure 2 for the seed word "tissue".

Word	English		German				
	Word tokens	Word types	Translations	Tokens	Word types	Sense cases (test data)	Correctly identify
Tissue	227,959	16,293	Gewebe	32,414	8,866	535	467
			Papiertaschentuch	36,842	8,634	166	132
Apron	70,968	8,102	Vorfeld	11,774	3,860	103	80
			Schutzblech	7,683	2,423	4	0
			Vorbuhne	12,696	4,122	71	0
			Schurze	7,560	2,410	137	70
Attack	103,633	9,956	Anfall	17,235	4,086	105	93
			Ubefallen	21,536	5,422	0	0
			Angriff	12,746	3,771	357	161
			attackieren	16,713	4,621	357	161
			In Angriff Nehmen	31,474	7,489	357	161

Figure 1: The table shows the statistics of our word specific corpora for the English words 'tissue', 'apron', and 'attack' and their possible German translations. Our method does not always provide a judgement for the respective word sense (target word selection). This lets the numbers of correctly identified senses appear lower than they actually are, especially when the applicability is low. A summary of the final experiment results is found in Table 2.

Word	Frequency	Word	Frequency	Word	Frequency
cell	729	study	198	cover	138
paper	442	muscle	190	bone	136
engineer	308	disease	184	develop	131
body	300	blood	181	lung	121
soft	286	structure	180	cause	117
human	278	animal	179	cancer	115
connective	259	culture	177	handkerchief	105
organ	248	toilet	169	normal	100
function	219	layer	151	bathroom	94

Figure 2: Extracted word stems of words related to the seed word 'tissue' along with their respective occurrence frequency in the word specific corpus.

An indirect evaluation approach in next section is adopted to evaluate the quality of the sets of semantically related words extracted with this measure. Semantically related words of several seed words are used to do word sense disambiguation of those seed words for machine translation between English and German.

4 Evaluation by Machine Translation

4.1 Target word selection as word sense disambiguation

In machine translation choosing the correct translation for a word is called target word selection problem. It is also a word sense disambiguation problem. In this case, word senses are defined by their distinctive translations into another language. To attack the problem some approaches have been proposed including knowledge based approach and corpus based statistical approach (Ide and Veronis, 1998; S.Sekine and J.I.Tsujii, 1995; N.Uramoto, 1995; H.A.Lee and G.C.Kim, 2002).

To evaluate the set of semantically related words extracted with RFR measure, we adopted the corpus based approach due to increasing availability of text data and the strong performance of recent corpus based statistical approaches. The experiment is conducted between English and German. We describe the experiments with respect to word sense disambiguation first rather than directly to the target word selection problem. This is because word sense disambiguation is a broader problem and our measure could be applied to it in general. So, we put this general problem before the target word selection problem.

Semantically related words (both English and German) are extracted for all three English seed words and their German translations. German words extracted for a German translation are used later as the join context of that German word. The set of semantically related English words, however, could not be used directly for word sense disambiguation. It is unknown as to which of them suggest one sense of the seed word, which suggest another. We need to convert a set of semantically related words to several sets of sense specific words. These data could then be used to match the new context of the seed word to disambiguate it.

A clustering algorithm is used to find sense specific clusters of words from the set of semantically related words. The algorithm is essentially the same as other clustering algorithms in that it attempts to find word clusters that have the strongest internal connection and to minimize the inter-cluster connections. The difference between such algorithms is often reflected by how the algorithm defines a connection. In our algorithm the connection is defined as word co-occurrence. If two words co-occur frequently enough (i.e. beyond coincidence) a connection is said to exist between them.

Surprisingly, during our experiments, we observed that the clusters found are not really sense clusters as many such clusters correspond to one sense of the seed word and many correspond another. Eventually, we come to realize that the co-occurrence statistics based clustering algorithm only goes half way to obtain sense specific clusters. Each cluster obtained should be, instead, called usage cluster. An concrete example would better explain the situation. In this example the word "tissue" is used as the seed word. A set of semantically related words are extracted with the RFR measure. From the words several clusters are obtained by the clustering algorithm. One cluster contains the words like "toilet","bathroom","roll","dispenser" etc, which clearly indicate the word used in the context of bathroom in the sense of toilet tissue. Another cluster contains the words like "flower","scissors","glue","colour","fold","cut" etc, which indicate the word being used in the context of handcrafts making in the sense of soft tissue paper as a material. These two clusters are difficult to be joined together based on co-occurrence because these two contexts rarely co-occur. In English we could say they represent different senses of the word, but in German they only have one translation. Or even in English if we take a broader view, we could say that they represent the same sense as a type of paper (in contrast to body tissues like organs). So they really correspond to word usage rather than word senses.

Unambiguously, the next step would be to join the usage clusters to form sense clusters, which is not an easy task. Different contexts (usage) rarely co-occur within a close vicinity. One fact, however, simplifies the process. Since senses of a English seed word is defined as the word's German translations. We could bypass the English sense cluster and directly join English usage clusters under German translations of the seed word. This could be done easily with the help of a bilingual dictionary. For example, the English word "tissue" has two usage clusters aforementioned. They are used in two different contexts. In German, however, the word "Papiertaschentuch" as a translation to "tissue" is used in both contexts. If we look up words from the two English usage clusters in a bilingual dictionary, naturally many of their German translations would all occur in the German contexts of "Papiertaschentuch". Thus we could join two English usage clusters under a German translation whenever we could match German translations of English words from both clusters to the context of that German word. The context of a German word is conveniently provided by the set of semantically related words extracted for it.

In summary, a set of semantically related words

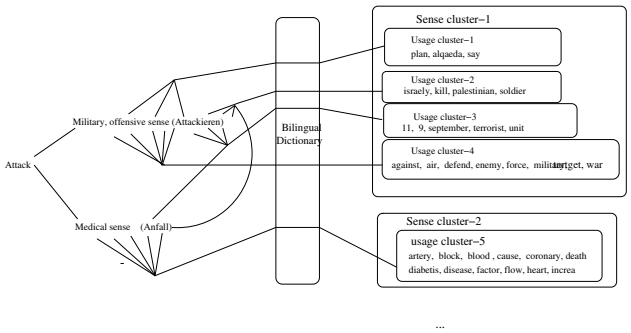


Figure 3: Usage clusters in English words are grouped around word senses based also on the German word-specific corpora.

are extracted for an English seed word X; a set of usage clusters are formed from this set; these usage clusters have to be joined under German translations of X; each usage cluster has a German representation which is the set of all possible enumerations of German translations to all the words in the cluster; this German representation overlaps with contexts of all German translations of X to different extent. The usage cluster Y would be assigned to one German translation context that has the biggest overlap with one of Y's German representations. Figure 3 shows how multiple English usage clusters could be joined under a single German translation (word). In the diagram for example, the English usage cluster2 passes through the "glass bar" of a bilingual dictionary, its German representation coming out of the dictionary has three matches with the first German translation's context, but only one match with the second German translation's context. It should be joined under the first German translation. Usage clusters thus joined under one German translation form a sense cluster that could be used to match new contexts of the seed word to disambiguate it. One sense cluster here corresponds to one German translation (word).

4.2 Testing results

Test data are collected from the Internet and are different from the training data. The sentences containing the English seed word are then labelled manually with the proper German translations of the seed word. Each test data set contains several hundred of such sentences. The sense clusters obtained with above approach are used to provide evidence for sense disambiguation alone and no other types of knowledge is used. The sense cluster that has the biggest overlap (words matched) with the new context assigns its corresponding German translation to the test sentence. This translation is compared to the correct translation manually tagged to the sentence. The results are summarized in Table 2. The 'Precision'(i.e. accuracy) column shows how often an assigned selection is correct. "Recall" indicates the percentage of cases where a judgment is made by the process.

Word	Testing data	Precision	Applicability
tissue	703 instances	97.5%	87.2%
apron	315 instances	69%	67.3%
attack	471 instances	93.7%	57.4%

Table 2: Summary of the disambiguation results. Precision is defined as the portion of correct judgements in the total number of judgments made. Applicability is defined as the portion of cases where a judgement is made in all tested cases.

5 Comparison with Related Works

There are other works that address the same problem of target word selection with different approaches. Sugimachi et al. in (Tomohiko Sugimachi and Matsuo, 2003) have used the Z-score (a refined derivative of mutual information) to extract semantically related words and form clusters from word graphs that resulted from the extraction. Their approach to the word sense disambiguation problem was evaluated qualitatively. Marquez in (Lluís Màrquez, 2000) compared five different supervised statistical approaches for WSD. They are Naive Bayes, Example Based Classifier, Winnow-based Classifier. They also investigated the effect of Boosting and Lazy Boosting. Their Lazy Boosting approach performed the best at an average of 71% accuracy on 21 selected words.

McDonald in (McDonald, 1998) used a vector distance calculation based multidimensional semantic space to calculate the closeness between alternative translations and the local context vector. Experimental results showed an accuracy around 58% at 100% recall, i.e. a judgment is made in every case. Khoen & Knight in (Koehn and Knight, 2000) used unrelated monolingual corpora in both languages together with a bilingual lexicon to build a translation model for 3830 German and 6147 English noun tokens. The probability distribution of different translations were estimated. They showed that the accuracy of their approach lies around 70% on average for a large collection of words.

Compared to these results our results are very encouraging, as our average accuracy is significantly higher. In particular, if we had used default

decisions provided in (Koehn and Knight, 2000), the recall would be much higher without substantially reducing the precision. What's important is that this machine translation application uses as its main knowledge only the set of semantically related words extracted with RFR. This (although indirectly) is sufficient as a proof to the effectiveness of the RFR measure we propose in this paper. The last thing worth mentioning is that mutual information has been used in place of RFR at early stages of the experiment but the precision rate stays at around 75 to 80% on average. Simply replacing mutual information with RFR under the exactly same framework pushes the rate up to 87% on average without compromising the applicability. One major difference made by RFR in comparison to mutual information is extraction of semantically related words with very high frequency. These high frequency words from WSCs all play vital roles in constraining the sense usage of the seed words.

6 Discussion and Conclusion

In this paper we introduced a new statistical measure RFR to extract semantically related words for a given word. The method can be applied to word sense disambiguation in general although we only showed how it could be applied to target word selection. Our experiments showed encouraging results, but because of time and resource limits it is only conducted for a small number of words so far.

The RFR measure could be used to obtain lists of domain specific words, topic specific words and basically, as long as a biased corpus could be obtained the list of words that are related to the biase could be extracted by the RFR measure. In our paper the WSCs are such corpora biased toward single words.

Some of the challenges that this measure faces are the same to those of current co-occurence based statistical approaches. One is to obtain a GC that is large enough and with good generality to provide good base statistics. But the seriousness of the problem is reduced by the fact that RFR measure is not overall sensitive to the bias unlike mutual information. If the GC is biased toward one domain, it will only affect the extraction of semantically related words for seed words in this domain. How could we better utilize the set of semantically related words is also an challenging problem. A general impression developed during the experiment of using the extracted words for WSD is that the measure often performs strongly in extracting domain or topic specific words. But word sense division does not often coincide with domain differences of the divided senses. Quite many sense divisions are based on local syntactic interaction of the word with surrounding words. This type of sense division is typical to verbs, nouns that originate from verbs and sometimes nouns with fine sense divisions. The extracted and clustered words usually do not perform well in this case. The core of the difficulty could just be the simple use of only word form co-occurrence information during the extraction and clustering. Future development of the work would be likely to focus on integrating other types of knowledge beyond word forms into the measure as well as finding of less demanding applications compared to WSD.

References

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

H.A.Lee and G.C.Kim. 2002. Translation selection through source word sense disambiguation and target word selection. In *In COLONG-2002*.

Nancy Ide and Jean Veronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. In *Computational Linguistics. Special Issue on Word Sense Disambiguation*.

Philipp Koehn and Kevin Knight. 2000. Estimating word translation probabilities from unrelated monolingual corpora using the em algorithm. In *Proceddings of the AAAI/IAAI 2000*. AAAI, Austin, Texas, USA.

Lluís Màrquez. 2000. Machine learning and natural language processing. Technical Report LSI-00-45-R, Departament de Llenguatges i Sistemes Informàtics (LSI), Universitat Politecnica de Catalunya (UPC), Barcelona, Spain.

Scott McDonald. 1998. Target word selection as proximity in semantic space. In *COLING-ACL*, pages 1496–1498.

N.Uramoto. 1995. Automatic learning of knowledge for example-based disambiguation of attachment. In *In Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 295–302, KU Leuven, Belgium.

S.Sekine and J.I.Tsujii. 1995. Automatic acquisition of semantic collocation from corpora. *Machine Translation*, 10(3):218–258.

Masayuki Takeda Tomohiko Sugimachi, Akira Ishino and Fumihiro Matsuo. 2003. A method of extracting related words using standardized mutual information. In *Proceedings of 6th Discovery Science Conference*, Sapporo, Japan, October. Springer-Verlag.

Representing and Rendering Linguistic Paradigms

David Penton and Steven Bird
Department of Computer Science and Software Engineering
The University of Melbourne, Victoria 3010, Australia
{djpenton,sb}@csse.unimelb.edu.au

Abstract

Linguistic forms are inherently multi-dimensional. They exhibit a variety of phonological, orthographic, morphosyntactic, semantic and pragmatic properties. Accordingly, linguistic analysis involves multi-dimensional exploration, a process in which the same collection of forms is laid out in many ways until clear patterns emerge. Equally, language documentation usually contains tabulations of linguistic forms to illustrate systematic patterns and variations. In all such cases, multi-dimensional data is projected onto a two-dimensional table known as a linguistic paradigm, the most widespread format for linguistic data presentation. In this paper we develop an XML data model for linguistic paradigms, and show how XSL transforms can render them. We describe a high-level interface which gives linguists flexible, high-level control of paradigm layout. The work provides a simple, general, and extensible model for the preservation and access of linguistic data.

1 Introduction

A linguistic paradigm is any kind of rational tabulation of linguistic forms, such as phrases, words, or phonemes, intended to illustrate contrasts and systematic variation (Bird, 1999). A characteristic property of paradigms is that interchanging entire rows or columns does not change the interpretation of the information. We view paradigms as a two-dimensional arrangement of elements and attributes, with optional row and column labels. An example of a paradigm for the German definite article is shown in Figure 1, with the labelling of number and gender at the top, and case on the left.

This paper describes a relational data model for linguistic paradigms, together with an XML based approach for representing and rendering them. An XSLT implementation provides proof of concept.[1] This work presents a simple and general model for the preservation and access of linguistic paradigms, and can generate an extensive range of useful visualisations.

PARADIGM FOR GERMAN DEFINITE ARTICLE

	SINGULAR			PLURAL
	MASCULINE	FEMININE	NEUTER	ALL GENDERS
NOMINATIVE	der	die	das	die
ACCUSATIVE	den	die	das	die
GENITIVE	des	der	des	der
DATIVE	dem	der	dem	den

Figure 1: Paradigm for German definite article (Finegan, 1999, 60)

This paper is organised as follows. In §2 we discuss the existing computational models in linguistic paradigms and the lack of an existing formalism. In §3 we discuss the data model, while §4 and §5 we provide an example of how to generate and visualise a linguistic paradigm. In §6 we discuss the query engine implementation and motivate each operation. In §7 we describe the transformation of the paradigm data into an XHTML presentation form. Finally, §8 discusses the significance of the work and outlines several areas for further investigation.

2 Background

Traditionally, there have been two sources for computational representations for linguistic paradigms, descriptive linguistic tools and technologies for languages having complex morphology. There has been little formal work in either area concerning the best form for this linguistic data type. Here we examine some of the more widely-used models and their drawbacks.

Of the descriptive linguistic tools, perhaps the foremost are Shoebox[2] and CHILDES.[3] Shoebox is an interlinear text editor popular among field linguists for analysing linguistic transcripts. The underlying model is an attribute value set for each

[1] The implementation is available from http://www.csse.unimelb.edu.au/research/lt/projects/paradigms/

[2] http://www.sil.org/computing/shoebox/

[3] http://childes.psy.cmu.edu/

Tag	Value	Description
\id	1612	identifier (used for hyperlinks)
\w	mbhū	orthographic form
\t	LDH	tone transcription
\p	n	part of speech
\pl	me-	plural prefix
\cl	9/6	noun class (singular/plural)
\en	dog	English gloss
\fr	chien	French gloss

Figure 2: Shoebox File Format, Adapted for Linguistic Paradigms (Bird, 1997)

entry, as shown in Figure 2. In the context of paradigms, each element corresponds to a cell in a table, and Shoebox can generate simple tabulated listings of forms which constitute a one-dimensional paradigm. The CHILDES CLAN tool supports transcription and analysis of conversation, and is widely used by psycholinguists in their study of child language acquisition. It has good search functionality that permits the generation of tabular reports. Despite their ability to generate simple paradigm-like reports, these systems do not provide an interface for generating arbitrary paradigms, nor do they permit paradigms to be saved in a format which permits reuse.

Outside purely linguistic description, work on computational morphology usually requires paradigms to be set up. For instance, Finnish and Romanian have such a large number of productive morphological processes it is impractical to list every form in the lexicon. Instead, regular derivational and inflectional processes are described using a formal system (such as a finite-state transducer). Groups of processes which apply to the same class of lexical items are sometimes referred to as a paradigm (e.g. (Tufis, 1989; Oflazer et al., 2001)). Unlike the descriptive viewpoint, in which a paradigm is a tabulation, here a paradigm is effectively treated as executable code which might be used to generate such tabulations. However, we are neutral on this issue since both viewpoints can be reconciled by treating a paradigm as a relation, as we do in §3.

3 Data Model

Linguistic paradigms associate linguistic forms with linguistic categories. For instance, the German definite article paradigm in Figure 1 categorises the form *den* as either masculine singular accusative or as dative plural. Systematic changes in layout, such as interchanging rows and columns, or flipping axes, have no affect on the associations between

D_1 gender	D_2 number	D_3 case	D_0 content
masc	sg	nom	der
masc	sg	acc	den
masc	sg	gen	des
masc	sg	dat	dem
masc	pl	nom	die
masc	pl	acc	die
masc	pl	gen	der
masc	pl	dat	den
fem	sg	nom	die
fem	sg	acc	die
fem	sg	gen	der
fem	sg	dat	der
fem	pl	nom	die
fem	pl	acc	die
fem	pl	gen	der
fem	pl	dat	den
neut	sg	nom	das
neut	sg	acc	das
neut	sg	gen	des
neut	sg	dat	dem
neut	pl	nom	die
neut	pl	acc	die
neut	pl	gen	der
neut	pl	dat	den

Figure 3: Function for the German Paradigm

forms and categories. Accordingly, a paradigm is a function that maps a vector of properties to content, as follows:

$$f : \langle \text{masc}, \text{sg}, \text{acc} \rangle \mapsto \text{den}$$

Generalising, let $D_0 \ldots D_n$ be a set of linguistic properties (or domains). Then a paradigm is a function:

$$f : D_1 \times \cdots \times D_n \rightarrow D_0$$

Let $D_1 = \{\text{masc}, \text{fem}, \text{neut}\}$, $D_2 = \{\text{sg}, \text{pl}\}$, and $D_3 = \{\text{nom}, \text{acc}, \text{gen}, \text{dat}\}$. Also, let $D_0 = \{\text{der}, \text{die}, \text{das}, \ldots\}$. The functional representation of the German paradigm is shown in Figure 3.

Observe that the original paradigm display in Figure 1 is a compact view of this table. It shows the domain values just once, and dispenses with the gender property for the plural forms.

Now, the above functional representation in Figure 1 is just a relational table with schema GermanParadigm (gender, number, case, content). We can use domain relational calculus to extract the columns of the original paradigm for display, e.g.:

$$\{s \mid t \in \text{GermanParadigm} \wedge t[\text{number}] = \text{`sg'}$$

$$\wedge\, t[\text{gender}] = \text{`masc'} \wedge t[\text{case}] = s[\text{case}]$$
$$\wedge\, t[\text{content}] = s[\text{content}]\}$$
$$= \{\langle \text{nom}, \text{der}\rangle, \langle \text{acc}, \text{den}\rangle, \langle \text{gen}, \text{des}\rangle, \langle \text{dat}, \text{dem}\rangle\}$$

The same query is expressed in SQL as follows:

```
SELECT case, content
FROM GermanParadigm
WHERE number = "sg"
AND gender = "masc".

nom, der
acc, den
gen, des
dat, dem
```

Standard XML technologies provide a more convenient way to map from this abstract representation to a range of visualisations. The relational table representation of a paradigm in XML is as follows:

```
<paradigm>
  <form>
    <attribute name="gender" value="masc"/>
    <attribute name="number" value="sg"/>
    <attribute name="case" value="nom"/>
    <attribute name="content" value="der"/>
  </form>
  ...
</paradigm>
```

XSL transforms provide a method to render this material into XHTML or into some other presentational markup language for delivery to users. Using this approach we will accomplish a round-trip, from existing visualisations to the abstract model discussed here, then back to visualisations. The next two sections describe this process in more detail.

4 Generating Paradigms

This section provides an overview of the steps required to produce a paradigm of possessive pronouns in Anejom̃, an Austronesian language of Vanuatu. The source data is from (Lynch, 1998). The process of generating paradigms is the same for paradigms that are more complicated, so the simplified Anejom̃ paradigm provides a helpful introduction. First there is a simple examination of the paradigm structure which motivates the choice of model for the paradigm. Then, by investigating an example query and looking at how it is processed and presented, the intricacies and obstacles to an effective model are evident. This leads onto the discussion in §6 and §7 which provides the finer details of the implementation and presents more elaborate examples in order to reveal the strengths and weaknesses of the model.

Figure 4 shows the architecture of the system. The processing pipeline has three components: an XML model, a query engine, and a presentation engine.

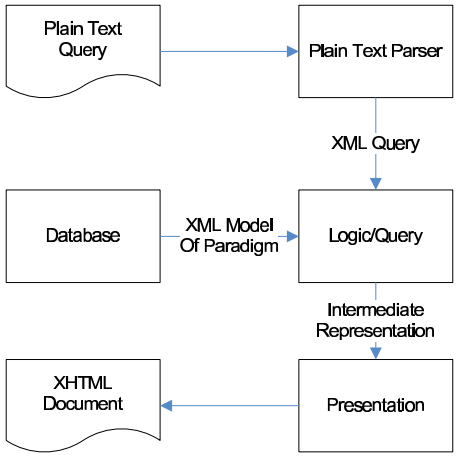


Figure 4: Architecture of System

Table 5. Anejom̃ Pronouns

	1. INC	1. EXC	2.	3.
Possessive				
Singular	—	*-k*	*-m̃*	*-n*
Dual	*-jau*	*-mrau*	*-mirau*	*-rau*
Trial	*-taj*	*-mtaj*	*-mitaj*	*-ttaj*
Plural	*-ja*	*-ma*	*-mia*	*-ra*

Figure 5: Anejom̃ Possessive Pronouns (Lynch 1998:106) - Scanned Version

Figure 5 shows a visualisation of a paradigm for Anejom̃. It displays suffix morphemes for possessive pronouns for different combinations of number and person. Each cell is characterised by its content and its attributes. For instance, the top right cell has content *-n*, a 'number' attribute whose value is 'singular', and a 'person' attribute whose value is '3'. Each attribute has a domain of possible values. For example, the domain of 'number' is 'singular', 'dual', 'trial' and 'plural'. The content is likewise a domain, having values such as '-jau' and '-mrau'. Figure 6 shows the XML model for Anejom̃. The attributes and their domains make up the first section, while the cells – the correspondences between content and attribute values, make up the second section.

The XML model provides a representation for the paradigm; the remainder of this section describes a plain text query language for generating different presentations from that model. The plain text query maps to an XML based representation. Then a XSL transform performs the underlying operations on the

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <attributes>
    <name name="person">
      <value value="1.INC"/>
      <value value="1.EXC"/>
      <value value="2"/>
      <value value="3"/>
    </name>
    <name name="number">
      <value value="singular"/>
..
      <value value="plural"/>
    </name>
    <name name="content">
      <value value="-"/>
      <value value="-jau"/>
      ..
      <value value="-ra"/>
    </name>
  </attributes>

  <paradigm>
    <form>
      <attribute name="person" value="1.INC"/>
      <attribute name="number" value="singular"/>
      <attribute name="content" value="-"/>
    </form>
    <form>
      <attribute name="person" value="1.INC"/>
      <attribute name="number" value="dual"/>
      <attribute name="content" value="-jau"/>
    </form>
    ..
    <form>
      <attribute name="person" value="3"/>
      <attribute name="number" value="plural"/>
      <attribute name="content" value="-ra"/>
    </form>
  </paradigm>
</document>
```

Figure 6: XML Model of Anejom̃ Possessive Pronouns

XML model of the paradigm. Here is an example. The Query 1 produces the visualisation of Figure 7 from the XML model of Figure 6. Translation 1 shows the full query. The table operator takes three arguments, the constraint applied to the vertical axis, the constraint applied to the horizontal axis, and the operation applied in each cell. The domain operation presents a list of all the values in a given domain. Note that the domain for the third argument of a table operation is different for each cell and determined by the values on the vertical and horizontal axes. Therefore Figure 7 shows a table with person and number as axes and content in the cells.

Query 1: table(person, number, content)

Translation 1: table(domain(person), domain(number), domain(content))

	1.INC	1.EXC	2	3
singular	-	-k	-m	-n
dual	-jau	-mrau	-mirau	-rau
trial	-taj	-mtaj	-mitaj	-ttaj
plural	-ja	-ma	-mia	-ra

Figure 7: Anejom̃ Possessive Pronouns: Table - Reproduced Visualisation

1.INC	singular	-
	dual	-jau
	trial	-taj
	plural	-ja
1.EXC	singular	-k
	dual	-mrau
	trial	-mtaj
	plural	-ma
2	singular	-m
	dual	-mirau
	trial	-mitaj
	plural	-mia
3	singular	-n
	dual	-rau
	trial	-ttaj
	plural	-ra

Figure 8: Anejom̃ Possessive Pronouns - Hierarchy induced by query

The model supports multiple visualisations of the data. For example, Query 2 produces a presentation of the XML model in a tree-like structure. In Query 2 the shorthand '/' symbol represents the hierarchy operation shown in Translation 2. The hierarchy operator takes two arguments, the constraint that forms a list and the operation applied to each element of the list. This produces the visualisation of Figure 8. Nesting table and hierarchy operations allows presentation of paradigms that are more complicated and n-dimensional.

Query 2: person/number/content

Translation 2: hierarchy(domain(person), hierarchy(domain(number), domain(content)))

5 Realising Paradigms

This section details the implementation responsible for presenting queries, using our running example. First, a PHP script maps the textual query representation to the equivalent XML representation shown in Figure 9. Then the 'logical' transform runs the query on the underlying XML model of the Anejom̃ paradigm (See Figure 6). Finally, the 'presentational' transform generates an XHTML presentation of that paradigm. Both transforms are written using XSLT.

```
<?xml version="1.0"?>
<document>
  <parse-tree>
    <operator opcode="table" instruction="1">
      <operand type="domain"
        arg="horizontal">person</operand>
      <operand type="domain"
        arg="vertical">number</operand>
      <operand type="domain"
        arg="cell">content</operand>
    </operator>
  </parse-tree>
</document>
```

Figure 9: XML Version of Query 1

The logical transform generates an intermediate representation from the XML query and the XML source model. The XSLT processor performs a depth first traversal of the query expression. For example, in Query 1 control starts at the table operation. The table operation requires calculating the domain of person and number before it can generate the cells. The domain operation generates the output tree of Figure 10 with a node for each value in its domain. The processor generates nodes 1INC, 1EXC, 2 and 3 for person. It then places the forms from the source tree that match the domain value under the corresponding output node.

When processing the table operation the XSLT processor searches the output trees of the vertical and horizontal branches for child nodes. The XSLT processor generates a cell for each combination of vertical horizontal child pairs. The combined set of nodes for each cell form the domain of the third argument. In the example, the first cell of the paradigm has the following mapping:

vertical: singular {-, -k, -m. -n}
horizontal: 1.INC{-, -jau, -taj, -ja}
cell(1,1): {-}

```
<form>
  <attribute name="person" value="1.INC"/>
  <attribute name="number" value="singular"/>
  <attribute name="content" value="-"/>
</form>
```

The query domain(content) in cell (1,1) produces a single node with a value '-'. The extended XML output of the sheet is shown in Figure 10. At each node the XSLT processor tags the number of leaves and maximum depth of the tree which simplifies the presentation logic.

The presentational transform renders the intermediate representation into XHTML for display on web browsers. It traverses the

```
<?xml version="1.0"?>
<document><operator optype="table">
  <vertical leaf-depth="1" leaves="4">
    <operator optype="domain" root-depth="1"
leaf-depth="1" leaves="4" direction="top-to-b.">
      <node ..>
        <att><html-att. element-name="th"/></att>
        <forms>
          <form>
            <att. name="person" value="1.INC"/>
            <att. name="number" value="singular"/>
            <att. name="content" value="-"/>
          </form>
          <form>
            <att. name="person" value="1.EXC"/>
            <att. name="number" value="singular"/>
            <att. name="content" value="-k"/>
          </form>
          <form>
            <att. name="person" value="2"/>
            <att. name="number" value="singular"/>
            <att. name="content" value="-m"/>
          </form>
          <form>
            <att. name="person" value="3"/>
            <att. name="number" value="singular"/>
            <att. name="content" value="-n"/>
          </form>
        </forms>
      </node> . <node heading="plural" ../>
    </operator>
  </vertical>
  <horizontal leaf-depth="1" leaves="4">
    <operator optype="domain" root-depth="1"
leaf-depth="1" leaves="4" direction="left-to-r.">
      <node heading="1.INC" ..>
        <forms>
          <form>
            <att. name="person" value="1.INC"/>
            <att. name="number" value="singular"/>
            <att. name="content" value="-"/>
          </form>
          <form>
            <att. name="person" value="1.INC"/>
            <att. name="number" value="dual"/>
            <att. name="content" value="-jau"/>
          </form>
          <form>
            <att. name="person" value="1.INC"/>
            <att. name="number" value="trial"/>
            <att. name="content" value="-taj"/>
          </form>
          <form>
            <att. name="person" value="1.INC"/>
            <att. name="number" value="plural"/>
            <att. name="content" value="-ja"/>
          </form>
        </forms>
      </node> . <node heading="3" ../>
    </operator>
  </horizontal>
  <cells>
    <row>
      <column>
        <operator optype="domain" root-depth="1"
leaf-depth="1" leaves="1" direction="left-to-r.">
          <node heading="-" ..>
            <forms>
              <form>
              <att. name="person" value="1.INC"/>
              <att. name="number" value="singular"/>
              <att. name="content" value="-"/>
            </form>
            </forms>
          </node>
        </operator>
      </column><column>..</column>
    </row><row>..</row>
  </cells>
</operator></document>
```

Figure 10: XML output from Query 1. The forms elements are only relevant in the horizontal and vertical nodes during processing. The common form element becomes the form element for the cell.

intermediate representation depth-first from the root. The XSLT document handles three classes of node; text leaves (node), text nodes with children (domains) and arrays of text nodes with children (tables). It must handle the following cases for each node; where the node is the contents of another node; and where the orientation of the node is vertical or horizontal. The next section provides further detail of how the query engine presents paradigms.

6 Operations

This section elaborates the domain, hierarchy and table operations, showing how they describe the presentation of a wide variety of linguistic paradigms. We also provide correspondences with relational queries:

Query 3: person

Translation 3: domain(person)

The SQL equivalent of Translation 3 is as follows:

```
SELECT person FROM paradigm
```

person
1.INC
1.EXC
2
3

Query 4: person/number

Translation 4: hierarchy (domain(person), domain (number))

```
SELECT * FROM
  (
    SELECT person FROM paradigm
      OUTER JOIN
    SELECT number FROM paradigm
  );
```

person	number
1.inc	singular
	dual
	trial
	plural
1.exc	singular
	dual
.	.
.	.

Query 5: table(person, number, content)

Translation 5: table(domain (person), domain (number), domain (content))

```
Qv = SELECT person FROM paradigm;
Qh = SELECT number FROM paradigm;
Qd = Qv OUTER JOIN Qh OUTER JOIN paradigm;
Qc = SELECT content FROM Qd;
```

The domain and hierarchy queries have straightforward mappings to SQL as shown for Translation 3 and Translation 4. The interactions of the table operation are complex, especially when looking at the query that produces the cells. To build the table the parser generates the axes using the query for the vertical and horizontal axes with a direct mapping of the queries from the first two arguments as shown in Translation 5. In this case, it is the queries for domain(person) and domain(number) that produce the desired SQL. A projection of the vertical and horizontal values (`Qv` and `Qc`) form the domain (`Qd`) for each of the cells (`Qc`). Any query on the cells applies only to this new domain (`Qd`). The result of the query `Qc` is a list of values which fill the table from the top left corner.

	Masculine Nouns	*Feminine Nouns*
Singular	*le bon livre* 'the good book'	*la bonne maison* 'the good house'
	ce livre vert 'this green book'	*cette maison verte* 'this green house'
	mon grand frère 'my big brother'	*ma grande soeur* 'my big sister'
Plural	*les bons livres* 'the good books'	*les bonnes maisons* 'the good houses'
	ces livres verts 'these green books'	*ces maisons vertes* 'these green houses'
	mes grands frères 'my big brothers'	*mes grandes soeurs* 'my big sisters'

Figure 11: French concord (Crowley, 1992, 322)

This covers the simple case where the paradigm has just three-dimensions. There are however, paradigms that have many more dimensions. The French concord of Figure 11 has five dimensions and provides a good source for more complicated queries. Consider the following two queries:

Query 6: table (gender, number, language/phrase)

Translation 6: table(domain(gender), domain(number), domain(language/phrase))

```
Qv = SELECT gender FROM paradigm
Qh = SELECT number FROM paradigm
Qd = Qv OUTER JOIN Qh OUTER JOIN paradigm
Qc =
  SELECT * FROM (
    SELECT language FROM Qd
    OUTER JOIN
    SELECT phrase FROM Qd
);
```

Query 7: table(gender, number, language)

Translation 7: table(domain(gender), domain(number), domain(language))

```
Qd = paradigm
Qc =
SELECT * FROM (
  SELECT language FROM Qd
  OUTER JOIN
  SELECT phrase FROM Qd
);
```

The difference in SQL for the cells between these two queries is the domain `Qd`. This represents the context for the query, its treatment is systematic throughout the XSLT logic allowing nesting of queries. Query 8 and Query 9 show queries that produce two different structures for the French language data. Using combinations of domain, hierarchy and table operation it is possible to generate almost all presentation layouts.

Query 8: table(gender/number, case/language, phrase)

Translation 8: table(hierarchy(domain(gender), domain(number)), hierarchy(domain(case), domain(language)), domain(phrase))

Query 9: table(gender, case, table (number, language, phrase))

Translation 9: table(domain(gender), domain(case), table(domain(number), domain(language), domain(phrase))

7 Presentation

This section describes the implementation of the presentation engine. This is the most complex component in the system; it produces XHTML from the underlying XML representation. The XSL transform does the processing with each operation acting as event. The domain operation is the simplest operation. It handles three distinct cases; one case for producing a list of values; one for producing a horizontal table; and one for producing a vertical table. The processing produces the following code for each:

```
Row:
<tr><td>Item 1</td><td>Item 2</td></tr>

Column:
<tr><td>Item 1</td></tr>
<tr><td>Item 2</td></tr>

Space separated list:
Item 1 Item 2
```

XSLT recursion solves the more difficult problem of constructing a hierarchy. The example of Table 1

Item 11		Item 12	
Item 21	Item 22	Item 23	Item 24

Table 1: Horizontal hierarchy of items.

Item 11	Item 21
	Item 22
Item 21	Item 23
	Item 24

Table 2: Vertical hierarchy of items.

is straightforward; each node has width equal to the number of its children (set with the colspan property). When the hierarchy is root, each level is a row with control grounded at the root. Control must be grounded at the root to avoid parts of the tree ending up in different rows. In Table 1 this equates to 'Item 11' and 'Item 12' forming one row and 'Item 21', 'Item 22', 'Item 23' and 'Item 24' forming the second row.

The same is true for the vertical realisation: the root node controls the generation of each row. However, in this case, there is a need for a policy for when to generate XHTML nodes. The problem is the XHTML language has one nesting of row and column yet two directions of spanning cells. Thus, in Table 2, 'Item 11' produces a node in row one but not row two and 'Item 21' produces a node in row three but not row four. This causes serious difficulties for any program written in a functional language. The solution is an intricate variable passing procedure where the generation of each label depends on whether it is the first node in the row. When it is first the label forms a cell with the rowspan property equal to the number of children in the hierarchy.

The generation of tables comes in three parts; the generation of the vertical axis; the horizontal axis and the cells. The XSL transform leaves the top-left square of the paradigm blank to avoid connection ambiguity problems. The generation of the horizontal axis is the same as when the table did not exist, albeit appropriately shifted by the depth of the vertical axis. The table operator iterates over each row generating first the vertical heading for that row then the cells for that row. This maps to the XHTML design of the table where the declaration of the rows comes before the columns.

The XSL transform treats operators as either controllers or fillers. As controller, the operator has responsibility for generating XHTML table and row tags. As filler, the operator just has responsibility for generating content. The nature of different orientations require different code for vertical and

horizontal orientations. When supported this allows integration and presentation of arbitrary commands. In fact this XSLT framework can display any query that used operations from §6.

8 Conclusion

This paper describes an XML model for linguistic paradigms, including a query language and implementation, along with a model for generating presentations and an implementation. This work provides a flexible and extensible representation for storing multidimensional linguistic paradigms; and a simple yet powerful method for accessing and analysing stored data. This model allows the easy manipulation of paradigm structure, and easy presentation of systematic patterns and variations. We believe that the XML representation will be useful for archiving linguistic paradigms and for the interchange of paradigms between programs. We also believe the presentation system supports multidimensional exploration of complex linguistic datasets, a linguistic version of what is known in the database world as online analytical processing (OLAP).

In the future, we plan to investigate the following: ordering paradigm content; generating paradigms from interlinear text; and investigating a multi-table model. Ordering the cells of a paradigm is an issue because it complicates the axes, which then require the repetition of headings. The second line of enquiry is the generation and integration of paradigm presentation into interlinear text systems, which requires a level of machine learning combined with an understanding of how to integrate different levels of linguistic description. The other issue is how to represent some of the relationships within paradigms such as the phonetic characters for a vowel and its height (eg. ϵ is a high vowel). We believe that the optimum solution for some of these problems is a multi-table model.

Acknowledgements

This paper extends earlier work by (Penton et al., 2004). This research has been supported by the National Science Foundation grant number 0317826 *Querying Linguistic Databases*.

References

Steven Bird. 1997. A lexical database tool for quantitative phonological research. In *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology*, pages 33–39.

Steven Bird. 1999. Multidimensional exploration of online linguistic field data. In Pius Tamanji, Masako Hirotani, and Nancy Hall, editors, *Proceedings of the 29th Annual Meeting of the Northeast Linguistics Society*, pages 33–47. GLSA, University of Massachusetts at Amherst.

Terry Crowley. 1992. *An Introduction to Historical Linguistics, second edition*. Auckland, NZ: Oxford University Press.

Edward Finegan. 1999. *Language: its structure and use*. Fort Worth: Harcourt Brace.

John Lynch. 1998. *Pacific Languages: an Introduction*. Honolulu: University of Hawai'i Press.

Kemal Oflazer, Sergei Nirenburg, and Marjorie McShane. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1):59–85.

David Penton, Catherine Bow, Steven Bird, and Baden Hughes. 2004. Towards a general model for linguistic paradigms.

Dan Tufis. 1989. It would be much easier if went were goed. In *Proceedings of the 4th European Conference of the Association for Computational Linguistics U.K. Manchester, 1989*.

Tauira: A tool for acquiring unknown words in a dialogue context

Maarten van Schagen
Human Media Interaction Group
University of Twente
7500AE Enschede
the Netherlands
schagen@cs.utwente.nl

Alistair Knott
Department of Computer Science
University of Otago
Dunedin 9015
New Zealand
alik@cs.otago.ac.nz

Abstract

This paper describes a tool for acquiring unknown words, which operates in a bilingual human-machine dialogue system. When the user's utterance includes a word which is not in the system's lexicon, the system initiates a subdialogue to find out about the new word, by querying the user about the syntactic validity of a number of example sentences generated automatically from the grammar's test suite. The tool can handle multiple unknown words, regular morphology and translation of new words within a very complex unification feature structure type hierarchy.

1 Introduction

A major problem for current wide-coverage symbolic parsing systems is the existence of **unknown words**: words which appear in the input strings to be parsed, but which are not in the system's lexicon. To give a representative example: when the English Resource Grammar (ERG: Copestake and Flickinger (2000)) was tested over a portion of the British National Corpus, unknown words accounted for 40% of all unparsed sentences (Baldwin et al., 2004). There are several ways of tackling this problem. Some of these are offline, and simply involve making the lexicon bigger, adding new entries by hand or by some semiautomatic process. Other methods are on-line, and are designed to cope with unknown words which are encountered during the parsing process itself.

This paper is concerned with on-line methods for dealing with unknown words. We focus on methods which are appropriate for implementation in a **human-machine dialogue system**. In this application, a human user enters a sentence; the system then parses the sentence, derives a semantic interpretation, and generates a suitable response. If the user's sentence contains an unknown word, then the system must somehow compute information about this word's syntactic characteristics (so that the sentence can be parsed), and about its semantics (so that the sentence can receive an interpretation). Some of this information can typically be inferred from the context in which the new word appears—but it is unusual if it can all be provided this way. Typically, some other source of knowledge needs to be actively consulted before the dialogue can resume.

A dialogue application opens up some new possibilities for unknown word processing, because when the system encounters an unknown word, it can initiate a **subdialogue** to find out more about this word. We will begin in Section 2 by outlining current approaches to unknown words in parsing, both in dialogue and non-dialogue applications. In Section 3 we will propose a new approach, which combines and extends existing approaches. In Section 4 we will present a new tool for unknown word processing which implements the proposed approach. Section 5 gives some examples of word-authoring dialogues produced by the tool we built, and Section 6 describes some avenues for further work.

2 Current approaches to handling unknown words in parsing

2.1 Knight: word-authoring dialogues

The suggestion that lexical items can be authored in a human-computer dialogue system is not new. The idea is first proposed by Knight (1996) as the **learning by instruction** paradigm. Knight proposes two dialogue based methods whereby a naive user could add new words to a system's lexicon. The first approach is to ask a set of multiple choice questions in which the new word is placed in different grammatical contexts and the user is asked about their syntactic correctness or where the user is asked various conceptual questions about the nature of the new word, as illustrated in the dialogue in Figure 1. The second approach is

H: John is hungry.
C: I don't know the meaning of "hungry". Is "very hungry" a reasonable phrase?
H: Yes.
C: Is "hungry" a visually detectable property ?
H: No. (...)

Figure 1: A multi-choice word-authoring dialogue

to let the user **paraphrase** the sentence containing the unknown word using concepts the system is already familiar with, as illustrated in Figure 2. Knight observes that although the

H: "John is hungry."
C: I don't know the meaning of "hungry"
H: I mean: "John wants to eat". (...)

Figure 2: An authoring dialogue using paraphrases

multiple choice method might work well for syntax, for semantics, the appropriate set of questions is rather ill-defined, and giving good answers requires a fair amount of semantic sophistication on the part of the user. He therefore did not try implementing this method. Even at the syntactic level, it is quite a challenge to decide what syntactic contexts to present the unknown word in when querying the user. We somehow need to choose the most informative contexts, so that the user is not queried more often than necessary, and so that we eventually end up identifying the actual syntax of the new word. If we assume a wide-coverage grammar such as the ERG, with over five hundred lexical types, this is a complex task.

2.2 Erbach, Barg & Walther, Fouvry: making use of sentence context

Even without asking the user about how a new word can be used in other contexts, it is still possible to extract a lot of information about the word's syntactic characteristics from the original sentence in which it appears. If we assume (a) that the word is of a syntactic type which the grammar already knows about, and (b) that the sentence would be parseable if the word were correctly identified as this type, there are only certain possibilities as to what type it can be. For example, a human reader ignorant of the word *zapf* could deduce from the sentence *the zapf chased his mother* that *zapf* is a noun, that *zapf* is singular (otherwise it would be *the zapf chase their mother*) and *this particular instance* of *zapf* is male.

A number of researchers have considered how best to make use of syntactic context to produce a set of **hypotheses** about an unknown word. Most such proposals are tied to a specific grammar formalism—typically, some form of typed unification grammar. An influential approach was that of Erbach (1990). In his system (as in many others), a set of **open-class word types** is defined (a subset of the full set of lexical types), and it is assumed that the unknown word is one of these types. Erbach defines a formalism for representing *disjunctions* of word types, and assigns the unknown word a lexical type using this formalism.

Much of the complexity of this formalism derives from the fact that lexical types can be defined for various **features** which can take different values; for instance, the type `noun` takes a feature `number` which can be `singular` or `plural`. When an unknown word is encountered it must be filtered before it can actually be added as a new word. This filtering process consists of selecting which features may be included in the final lexical entry. For the type `noun` the feature `number` of an unknown word derived from the sentence context will be included in the new lexical entry. The feature `tense`, however, which can also be derived from the sentence context for the unknown word, will not be included in new lexical entries for the type `noun`.

Barg and Walther (1998) refine Erbach's treatment of unknown words. A word according to Barg and Walther is not just known or unknown, but is an entry open to constant revision. Every time a word appears within the sentence context a revision takes place. A word not available in the lexicon is represented by the conjunction of all open class word types. For the revision process, features are marked as either generalizable or specializable. Generalizable features are features such as `gender` for nouns, which can have different values in different contexts (e.g. the word *zapf* in the sentences *the zapf chased his mother* and *the zapf chased her mother*). Specializable features are features such as `number`, which once they occur once with a value in a context cannot occur with another value in a different context. (Note that when `number` for a noun is specializable, morphologically irregular words such as *sheep*

should be defined in two separate lexical entries, with `number` set to `singular` and `plural` respectively.)

Fouvry (2003) adapts the Erbach/Barg&Walther technique for the LKB parsing system (Copestake and Flickinger, 2000). He notes an important method for simplifying the technique, which stems from the fact that in many modern unification-based grammars (including many developed for the LKB system) lexical types are not associated with features, for efficiency reasons. Rather, for instance, there are individual entries in the **word type** hierarchy for `singular-np`, `plural-np`, and so on. Many of the complex filtering techniques proposed by Erbach, or grammar annotations proposed by Barg & Walther, are simply no longer necessary in such grammars. In our unknown word mechanism, which is also an adaptation of the LKB parser, we will make use of this simplification.

2.3 Summary, and some remaining problems

To some extent, the two basic approaches to on-line unknown word processing just discussed have complementary benefits and drawbacks. There are good syntax-based methods for deriving hypotheses about an unknown word's syntactic type from the context it appears in. However, there are often several alternative hypotheses for a given word, and processing a corpus of sentences in 'batch' mode to decide between these is a rather undirected process. Dialogue-based approaches offer the possibility of a tightly focussed set of questions to focus in on the correct hypothesis, with the user providing the answer at each stage. The problem is in deciding how to generate suitable questions.

In addition, there are some extensions to the syntactic paradigm which have not yet been considered. No-one has yet developed a way to process *multiple* unknown words, whether these are interpreted as a single multi-word lexeme, or two separate lexemes. The problem, as Fouvry notes, is that when there is more than one unknown word, the space of possible parses becomes extremely big. Also, no-one has a way to consider different *morphological analyses* of the unknown word when generating hypotheses about its lexical type. Finally, as Knight noted, it is difficult to use a multiple choice method to define the *semantics* of an unknown word. For this task, a sentence paraphrasing process seems more appropriate.

3 A new proposal for unknown words

We propose a new algorithm for unknown word processing, which draws on the strengths of Knight's dialogue-based methods and of Fouvry *et al's* syntactic hypothesis-formation methods. The system begins by deriving a set of hypotheses about the type of the unknown word, just as Fouvry does. It then generates a set of **test sentences** to present to the user, in a Knight-style multi-choice question. The answer to this question reduces the set of hypotheses, and we iterate by asking further questions. When the syntactic type of the word is established, we finish by asking a paraphrase question, to establish its semantics.

A key feature of the proposed system is its method for generating test sentences. We propose to use the **test suite** of sentences which comes with the grammar for this purpose. All large-scale grammars developed nowadays come with special-purpose test suites, and often with special tools for running the parser on these suites (such as the `tsdb++` system for LKB: Oepen (2001)). The important point is that the sentences in a given test suite collectively provide a very good description of the complete set of constructions covered by the grammar it is developed for. (At least they should do, if the test suite is well designed.) An important component of any test suite is a collection of **minimal pairs** of sentences, which differ only in a single grammatical aspect. Figure 3 gives an example of some minimal pairs for verbs which take a subject but no object. Such minimal pairs are exactly the kinds of sentences we need in order to decide how to classify a new word.

It rained.
Abrams barked.
The window opened.
It barked.
Abrams opened.
The window rained. (. . .)

Figure 3: Minimal pairs in the MRS test suite

There are three additional features of the new algorithm, which we will discuss in turn.

3.1 Preprocessing, for multi-word lexemes

As noted in Section 2.3, traditional unknown word processing methods using sentence context are unable to handle multiple unknown words within one sentence, due to the compounded ambiguity. Processing unknown words within the context of a dialogue, however, opens up the possibility of letting the user work around this problem.

Consecutive unknown words can make up just one lexical entry such as *yellow-eyed penguin*. Therefore for each sequence of consecutive unknown words the user is first consulted as to whether or not these make up one lexical entry.

After it is resolved whether multiple unknown words make up one lexical entry there might still be multiple new lexical entries in one sentence. To work around the problem of compounded ambiguity in cases like this, the user is asked to provide new sentences, each containing just one of the unknown words.

The two previous two user-consultation steps result in sentences with only one unknown lexical entry. From these sentences information can be gained to create hypotheses for the unknown words.

3.2 Extensions to deal with morphological ambiguity

In a modern grammar like the ERG, with syntactic features compiled into the hierarchy of lexical types, the syntactic character of an unknown word can be expressed very simply as a set of alternative **hypotheses**—basically, a set of all the lexical types to which the word can still be assigned. However, there is one further uncertainty, which relates to the morphological analysis of the unknown word. If the unknown word is *bobsled*, for instance, the system can hypothesise that it is an uninflected noun, but also that it is the regular past tense of a new verb *to bobsle*.

We extend the algorithm to deal with regular morphology as follows. Each word has a **stem** (e.g. *walk* or *dog*) on which so-called **morphological inflection rules** can be applied which will add a prefix and/or suffix to the stem. Note that these morphological rules will never decrease the size of the stem. The rule for plurality will inflect *dog* to *dogs* and *albatross* to *albatrosses*, but not *analysis* to *analyses*.

To deal with regular morphology in unknown words these morphological rules can be applied backwards. Applying the plurality rule backwards on *zapfes* will give *zapfe* as a possible stem. Applying the third person present rule backwards on the same word will give *zapfe* and *zapf* as possible stems. A hypothesis about a word's syntactic character now becomes a tuple, whose first element is a lexical type, and whose second element is a stem. Of course certain morphological rules can only be applied to certain types. When entering *The winner is zapfing* both the pair (`proper-name`,*zapfing*) and (`verb`,*zapfe*) occur as hypotheses, but not (`proper-name`, *zapfe*) since proper names cannot be inflected using the present participle rule.

3.3 Multilingual paraphrases, for word semantics

In grammars which have a treatment of compositional semantics, such as those supported by the LKB system, it is necessary to produce a representation of the semantics of a new word, as well as of its syntax. There are simple ways of doing this—for instance, we can just create a 'dummy' semantic value, derived from the orthography of the word. However, the grammar we use is bilingual, and is intended for use in sentence translation or bilingual dialogue applications: our grammar can parse and generate both English sentences and Māori sentences. It is useful in this context to be able to specify semantic correspondences between sentences containing the new word and translations of these sentences in the other language. We therefore propose a simple method for specifying the semantics of an unknown word, by allowing the user to enter paraphrasing sentences in the other language.[1]

The semantics of a sentence is given as a formula in Minimal Recursion Semantics (MRS: Copestake *et al.* (1999))—basically, a 'flat' collection of predicates with associated arguments. For the purposes of defining the semantics of new words, we can simply treat the sentence as a set of predicates: for instance *The kitten eats* is represented by {`determiner-pred`, `little-pred`, `cat-pred`, `eat-pred`}; and its Māori translation *ka kai te punua poti* (literally *Present eat the cat little* is represented by the same set of predicates.

[1] Needless to say, our system can also be applied to a monolingual grammar. In this case, no paraphrase needs to be specified. However, the paraphrasing system we describe here might still be of use, in specifying words which are synonyms of one another.

Using these set semantics the user can be requested to give a paraphrase in Māori for the original English phrase the unknown word was used in. The semantics of the unknown word in English are then equal to the set of semantics of the paraphrase excluding the semantics of known words in the original phrase. Of course this also works the other way around.

The paraphrase can again contain unknown words. But since there will be only one new lexical entry (that of the translation of the original unknown words, possibly consisting of multiple new words) the same methodology as for the original sentence applies for the translation.

4 The Tauira system

In this section, we describe the system we have built to implement the unknown-word-processing algorithm just outlined. The system is called Tauira, which is Māori for 'student', and also for 'example'. Preprocessing to identify multiword lexemes and multiple lexemes is straightforward, so there are two main components of the system to describe. Section 4.1 describes the way in which test sentences featuring the new word are created by selecting and transforming sentences from the test suite, and Section 4.2 describes the way these test sentences are used to form a series of questions to ask the user.

4.1 Creating test sentences from the test suite

To begin with, we preprocess the test suite offline to produce a set of **test items**. A test item is a sentence from the suite in which one 'target' open-class word has been extracted, together with its original lexical type and morphological rule. In addition all possible lexical types that can appear in this word's position (termed the **associated types**) are added. The idea is to create test sentences for the user by plugging the unknown word into the place from which the target word was removed. For example, from the test suite sentence *How happy was Abrams*, we would derive the following test items:

- *How _ was Abrams*, <adjective,nil>, {*adjective*,*adverb*}
- *How happy was _*, <propernoun,nil>, {*propernoun*,*dayoftheweek*...}

The associated types for the first of these items includes 'adverb', because of sentences like *How early was Abrams*. Those for the second item include 'dayoftheweek' because of sentences like *How happy was Tuesday*.

After this preprocessing has taken place, the effectiveness of each test item in reducing the set of hypothesis types can be evaluated. For each open-class word type, a number of benchmarks can be defined. Firstly, we define the number of test-items whose original type is this word type. (If this is zero, then there are no sentences the system can use to verify that an unknown word is of this type.) Secondly, we define the number of **positively indistinguishable** open-class word types for this type. This is calculated as the number of open-class word types which are found in *all* the test items which include this type in their associated types. This number represents the number of hypotheses which would remain if the user affirmed that an unknown word could be used in all these test items. Finally, we define the number of **negatively indistinguishable** open-class word types for the given type. Type $wt2$ is negatively indistinguishable from type $wt1$ if there are no test items where $wt1$ can be used and $wt2$ can*not* be used. We calculate the number of negatively indistinguishable types for word type $wt1$ by searching through its set of positively indistinguishable word types, and removing all types $wt2$ that do not have $wt1$ in *their* set of positively indistinguishable word types.

These benchmarks are in fact very useful as a formal way of evaluating the adequacy of a test suite accompanying a grammar. What we want, for all word types, is for there to be *no* positively indistinguishable types, or *no* negatively indistinguishable types. If there is a genuine reason for distinguishing between two lexical types, we expect there to be test items which allow us to distinguish them.

We ran these benchmarks on several test suites associated with the ERG grammar. We found some gaps in individual test suites; for instance, in the MRS test suite, there are no sentences for distinguishing between unergative and unaccusative verbs. Joining several test suites together might solve the problem, but we were not able to do this, as the algorithm for creating test items uses a lot of memory. So there are still some gaps in the set of test items we created for the ERG.

4.2 Dialogue strategies

After one unknown lexical entry is identified per sentence, a set of the current possible **hypotheses** (pairs of word types and stems) is generated. Different types of questions are then posed in order to reduce the number of hypotheses.

For unknown words without morphological ambiguity, **multiple choice questions** are generated for each hypothesised word type. These questions present a set of alternative sentences formed from test items and featuring the unknown word, and ask the user to choose which sentence (if any) provides an appropriate context for it. The algorithm for creating sentences is as follows. For the hypothesis whose word type occurs most frequently in the lexicon, we select the test item whose set of associated types has the smallest intersection with the set of current hypotheses, and create a sentence using this test item and the unknown word. We then iterate, looking for additional test items whose set of associated types do not overlap with the associated types of the items chosen so far. Finally, we include the additional option 'none'.

For unknown words with morphological ambiguity, **yes-no questions** are generated about the validity of a single sentence. Since morphological rules within our setup never decrease the size of the stem, the shortest stem will always be queried first. When a sentence is accepted, only the hypotheses containing a corresponding stem and original or associated types remain. When a sentence is not accepted the hypotheses containing a corresponding stem and the original sentence type are removed.

Finally, if the test suite simply fails to provide sentences which distinguish between two types, we fall back on a simple **similar words** question, which presents examples of words from the lexicon of the remaining possible word types, and asks the user to choose which sets of words are most like the unknown word.

5 Results: some dialogues with Tauira

In this section some example dialogues with Tauria are given, to demonstrate what it is capable of. The first of these examples are based on a set of test items derived from test suites associated with the ERG grammar.

The example in Figure 4 shows how a proper name in English is authored in three simple questions, even if the original phrase the word is very ambiguous. To show how ambiguous this sentence is, a syntactically equal sentence sentence is given in Figure 5 with the unknown adjective *Dutch*. The last question exemplifies the back-up mechanism when there are no suitable test suite sentences to distinguish between word types. In this case there is no test suite sentence to distinguish between transitive and intransitive adjectives.

H: my name is maarten
C: I do not understand the word MAARTEN.
Which of these phrases illustrates the correct use of MAARTEN ?
1. how maarten was abrams
2. the maartens of the projects are trustworthy
3. mr maarten browne is the manager
4. there are five maarten in the room aren't there
5. None
H: 3
C: Is 'five of the seven maartens work for abrams' a correct sentence ?
H: no
C: Is 'maarten knows that sara will sleep doesn't he' a correct sentence ?
H: yes (...)

Figure 4: Example adding a proper name in a very ambiguous sentence

H: my name is dutch
C: I do not understand the word DUTCH.
Which of these phrases illustrates the correct use of DUTCH ?
1. how dutch was abrams
2. the dutches of the projects are trustworthy
3. mr dutch browne is the manager
4. there are five dutch in the room aren't there
5. None
H: 1
C: Which of these phrases illustrates the correct use of DUTCH ?
1. abrams works for a dutch manager
2. how dutch does abrams interview a programmer
3. dutch hundred twenty dogs bark
4. None
H: 1
C: Which of these words is most like your word ?
1. 'european', 'stupid', 'gusty'
2. 'uncertain', 'invisible', 'aware'
H: 1 (...)

Figure 5: Example adding an intransitive adjective in a very ambiguous sentence

The example in Figure 6 exemplifies morphological disambiguation. The sentence *I danced*

H: I danced
C: I do not understand the word DANCED.
Is 'the dog arrived and danced' a correct sentence ?
H: yes
C: Is 'abrams dancs for browne' a correct sentence ?
H: no
C: Which of these words is most like your word ?
1. 'stand', 'orientate', 'bay'
2. 'bust', 'grow', 'blow'
H: 1 (...)

Figure 6: Example illustrating morphological stem reduction

for unknown word *danced* has three possible stems: *danc* (which takes *-ed*, c.f. *walk*); *dance* (which takes *-d*, c.f. *chase*); and *danced* (as an uninflected present-tense verb, c.f. *bobsled*). The first question checks pairs with intransitive verbs and the stems *danc* and *dance*. The second question checks the stem *danc* for an unergative intransitive verb and is refused, so this pair is removed from the complete set of hypotheses.

The example in Figure 7 exemplifies how the system deals with multiple words and translations when describing an encounter between two inhabitants of the Otago Peninsula: the *albatross* (*toroa*) and the *yellow-eyed penguin* (*hoiho*). Since the Māori grammar is still under development and consists of no more then 15 different types, questions posed (if any) will not have many alternatives. In the final utterance, the system, having created the necessary new lexical items, reprocesses the original sentence containing the unknown words, and produces a set of translations.

6 Summary and further work

Tauira extends the theory of Erbach (1990), Barg and Walther (1998) and Fouvry (2003) in two straightforward ways. Firstly it simplifies the creation of a new lexical entry by explicitly formulating a set of simple hypotheses, which can be eliminated one by one, instead of creating a disjunct feature structure which needs to be filtered. Secondly, it takes morphological information into account in unknown word processing. This is a first step to truly robust processing of unknown words not offered by any of the previous works. Thirdly, using a sentence paraphrasing task, it is able to provide simple semantics for unknown words, to allow sentence translation using the newly authored words.

Tauira also has many advantages over other lexical acquisition tools which do not operate within a dialogue context. First of all, more then one unknown word per sentence can be resolved; this was a serious problem for Erbach, Barg and Walther and Fouvry, but one for which there are easy work-arounds in a dialogue context. Moreover, Tauira provides a simple natural language dialogue through which a non-linguist user can author new words. Our human-machine dialogue system asks many kinds of clarification question in different circumstances; the questions asked by Tauira are very easy to integrate into this general framework.

Finally, Tauira can be used on any grammar and test suite developed for the LKB system. The questions it asks the user are generated fully automatically from the grammar's test suite, and therefore evolve together with the development of the grammar and test suite. In addition, as a side-effect, Tauira's routines for preprocessing the test suite define benchmarks which can be used to formally evaluate a test suite's coverage in relation to a grammar.

There are several things we would like to do in future work. These include: running a user evaluation, checking for incorrect spelling in unknown words, using a statistical part of speech tagger to decrease the initial set of hypotheses, taking syntactic, semantic and multi-word homonyms into account, and dealing with irregular morphology. Ideas to realize these future works are dealt with in detail in the Tauira technical report (van Schagen, 2004).

We are also interested in the prospect of fully automating the word-authoring dialogues. Instead of querying a user about the validity of a given sentence featuring the new word, it may be possible to search for sentences on the web which have the syntactic structure of the test sentence and which contain the new word in the appropriate position. (Naturally, it does not matter what the other words in the sentence are, provided they have the right parts of speech.) Finding such a sentence is akin to receiving the answer 'yes' in an authoring dialogue. Clearly, identifying the sentences retrieved by the web search which have the right syntactic structure would be a major task; however, the directed nature of the question-answering process does at

least strongly minimize the number of searches which would be needed. This is a topic we plan to address in future work.

References

T Baldwin, E Bender, D Flickinger, A Kim, and S Oepen. 2004. Road-testing the english resource grammar over the british national corpus. In *Proceedings of LREC 2004*, Lisbon, Portugal.

P Barg and M Walther. 1998. Processing unknown words in HPSG. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montréal, Canada.

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

C Erbach. 1990. Syntactic processing of unknown words. In P Jorand and V Sgurev, editors, *Artificial Intelligence IV: methodology, systems, applications*, pages 371–382. North-Holland, Amsterdam.

F Fouvry. 2003. Lexicon acquisition with a large-coverage unification-based grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics, Research notes and demos*, Budapest, Hungary.

K Knight. 1996. Learning word meanings by instruction. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, Oregon.

Stephan Oepen. 2001. [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Maarten van Schagen. 2004. Tauira: A bilingual dialogue-based lexical acquisition system. Technical Report OUCS-2004-06, Department of Computer Science, University of Otago, Dunedin, New Zealand.

H: the albatross chases the yelloweyed penguin
C: I do not understand the words ALBATROSS, YELLOWEYED, PENGUIN.
Is YELLOWEYED PENGUIN one word ?
H: yes
C: Could you provide me an example sentence with the unknown word 'ALBATROSS' using words I know ?
H: the albatross eats
C: Which of these phrases illustrates the correct use of ALBATROSS ?
1. the albatrosses of the projects are trustworthy
2. mr albatross browne is the manager
3. browne was hired on january albatross 1984
4. None
H: 4
C: Is 'he showed her an albatross' a correct sentence ?
H: yes
C: Which of these words is most like your word ?
1. 'sec', 'counter narcotics', 'paycheck'
2. 'school', 'black', 'distance'
3. 'delivery', 'record', 'treatment'
4. 'specification', 'subscriber', 'deviation'
H: 1
C: What is the translation of 'the albatross eats' in Māori.
H: ka kai te toroa
C: Is 'kua pai te toroa' a correct sentence ?
H: yes
C: Which of these phrases illustrates the correct use of YELLOWEYED PENGUIN ?
1. the yelloweyed penguins of the projects are trustworthy
2. mr yelloweyed penguin browne is the manager
3. browne was hired on january yelloweyed penguin 1984
4. there are five yelloweyed penguin in the room aren't there
5. None
H: 5
C: Is 'list yelloweyed penguins who have bookcases' a correct sentence ?
H: yes
C: Which of these words is most like your word ?
1. 'sec', 'counter narcotics', 'paycheck'
2. 'school', 'black', 'distance'
3. 'delivery', 'record', 'treatment'
4. 'specification', 'subscriber', 'deviation'
H: 1
C: What is the translation of 'the albatross chases the yelloweyed penguin' in Māori.
H: ka whai te toroa i te hoiho
C: 'the albatross chases the yelloweyed penguin' translates to:
ka whāia te hoiho e te toroa
ka whai te toroa i te hoiho

Figure 7: Example illustrating multiple words and translation

Querying and Updating Treebanks: A Critical Survey and Requirements Analysis

Catherine Lai and Steven Bird
Department of Computer Science and Software Engineering
University of Melbourne, Victoria 3010, Australia
{clai,sb}@csse.unimelb.edu.au

Abstract

Language technology makes extensive use of hierarchically annotated text and speech data. These databases are stored in flat files and manipulated using corpus-specific query tools or special-purpose scripts. While the size of these databases and the range of applications has grown rapidly in recent years, neither method for managing the data has led to reusable, scalable software. The formal properties of the query languages are not well understood. Hence established methods for indexing tree data and optimizing tree queries cannot be employed. We analyze a range of existing linguistic query languages, and adduce a set of requirements for a reusable, scalable linguistic query language.

1 Introduction

Corpora form the backbone of language technology. However corpora usually need to be annotated with structural information describing, for example, syntax or phonology. Query languages are necessary to extract useful information from these massive data sets. Moreover, annotated corpora require thousands of hours of manual annotation to create, revise and maintain. Query languages are also useful during this process. For example, queries can be used to find parse errors or to transform annotations into different schemes.

However, the query languages currently available for annotated corpora suffer from several problems. Firstly, updates are not supported as query languages focus on the needs of linguists searching for syntactic constructions. Secondly, their relationship to existing database query languages is poorly understood, making it difficult to apply standard database indexing and query optimization techniques. As a consequence they do not scale well. Finally, linguistic annotations have both a sequential and a hierarchical organization. Query languages must support queries that refer to both of these types of structure simultaneously. Such hybrid queries should have a concise syntax. The interplay between these factors has resulted in a variety of mutually-inconsistent approaches.

This paper aims to describe the query language requirements for navigating and modifying structurally annotated corpora. We focus on tree structured annotation. This entails sequential structure.

This paper is organized as follows. Section 2 surveys six linguistic tree querying languages and where appropriate the data models they are based on. The survey is the basis of the linguistic tree query requirements presented in Section 3. We conclude in Section 4 with suggested areas for further work.

2 Tree Models and Query Languages

The prototypical hierarchical linguistic annotation is the syntax tree, an ordered tree where terminals (leaves) contain the text of a sentence being analyzed. Non-terminals represent syntactic categories, and the hierarchical organization represents constituency. The leaf level is usually considered immutable. The queries in Figure 1 have been chosen to highlight the expressive capabilities of current tree query languages.

A query language must be able to accurately specify which subtrees to match in a corpus. This means quantifying the existence of nodes and succintly stating the relationships between them. Q1 is a simple query based on the dominance relation inherent in trees. As mentioned earlier, however, the sequential ordering of nodes is also an important factor. Q3 and Q4 demonstrate the precision that is available for describing subtrees constrained by both dominance and precedence relations.

It is also desirable to specify subtrees by what they do not contain. This requires some form of negation. Q2 is a simple example of this type of

Q1. Find sentences that include the word 'saw'.
Q2. Find sentences that do not include the word 'saw'.
Q3. Find noun phrases whose rightmost child is a noun.
Q4. Find verb phrases that contain a verb immediately followed by a noun phrase that is immediately followed by a prepositional phrase.
Q5. Find the first common ancestor of sequences of a noun phrase followed by a verb phrase.
Q6. Find a noun phrase which dominates a word *dark* that is dominated by an intermediate phrase that bears an L-tone.
Q7. Find an noun phrase dominated by a verb phrase. Return the subtree dominated by that noun phrase only.

Figure 1: Syntactic Queries for Comparing Tree Query Languages

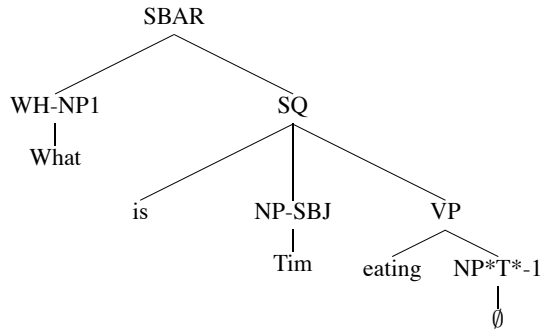


Figure 2: Trace elements in Penn Treebank

query. Q5 contains an implicit negation: we need to select the common ancestor that has no descendant that is also a common ancestor of the nodes in question. These queries also explore the interaction between quantification and negation of nodes and subtrees.

Linguistic query languages, in general, need to be able to deal with heterogeneous data. Many interesting queries fall on the interface of linguistic fields. For example, Q6 requires both syntactic and phonological data. This can be represented as two trees that intersect at the word level (Cassidy and Harrington, 2001).

Finally, trees can be large, and it is often undesirable to return whole trees for which the query tree matches a tiny fragment. Thus, we need a way to specify what part of a query simply constrains context, and what part should be returned. Q7 is a simple test of a query language's ability to control output.

2.1 Penn Treebank and Tgrep2

Penn Treebank contains approximately 50,000 parse trees of Wall Street Journal text (Marcus et al., 1994). Each parse is represented as an ordered tree, and syntactic dependencies are indicated using

```
Q1.  S << saw
Q2.  S !<< saw
Q3.  NP <- N
Q4   VP=vp << (V . (N >> =vp . PP >> =vp))
Q5.  *=p << (NP=n .. (VP=v >> =p
             !>> (* << =n >> =p)))
Q6.* Not expressible
Q7.  VP << 'NP
```

Figure 3: Tgrep2 Queries; Q5 is taken from (Rohde, 2001)

zero-width *trace* elements co-indexed with a full noun phrase (cf. Figure 2).

Tgrep2 is a *grep*-like tool for this database (Rohde, 2001), and all example queries can be specified correctly except Q6 (cf. Figure 3). Queries are nested expressions involving nodes and relationships between nodes. Nodes are specified using strings or regular expressions. Tgrep2 supports a large number of node relationships, including immediate precedence: `A` immediately precedes `B` (ie `A . B`) if the right corner of `A` immediately precedes the left corner of `B` in the sentence on which the tree is built. A wildcard (`*`) can be used for a node when there is no constraint on its name. Node identifiers allow multiple sub-expressions to refer to the same node (e.g. Q4). Tgrep2 can specify non-inclusion or non-existence (Q2, Q5).

Query execution uses a binary file representation of the data, including an index on the words in the trees. The top node in a query is first matched and the rest of the tree is matched recursively, constrained by node relations. The output is a set of human-readable subtrees. The subtree matched by the first node mentioned in the query is returned by default, but another return subtree can be specified by the user as shown in Q7. This output cannot be queried, since it must first be converted into the binary format; i.e. Tgrep2 is not compositional.

Not all node relationships are primitive (e.g. sibling relation (`$`)). However, dominates type relations such as leftmost descendant (`<<,`) cannot be derived. The depth of such a descendant is unknown. Thus, the language could be greatly simplified with a closure operator.

2.2 The TIGER Corpus and TIGERSearch

The TIGER corpus contains syntactically annotated German newspaper text (Brants et al., 2002). The syntax of German permits *discontinuous constituents*, which are represented in the corpus as trees with crossing edges. TIGERSearch is a logic and constraint programming approach to querying

```
Q1.   #s:[cat="S"] & #l:[lex="saw"]
      & #s >* #l
Q2.*  #s:[cat="S"] & #l:[lex="saw"]
      & #s !>* #l
Q3.   #n1:[cat="NP"] & #n2:[pos="N"]
      & (#n1 >* #n2) & (#n1 >@r #n3)
      & (#n2 >* #n4)
Q4    #vp:[cat="VP"] & #v: [pos="V"]
      & #np:[cat="NP"] & #pp:[cat="PP"]
      & #vp >* #v & #vp >* #np
      & #vp >* #pp & #v >@r #vr
      & #np >@l #npl & #vr .1 #npl
      & pp >@l ppl & npl .1 ppl
Q5.*  #vp:[cat="VP"] & #np:[cat="NP"]
      & (#x >* #v) & (#x >* #np)
      & (#v .* #np)
Q6.   Not expressible
Q7.*  #vp:[cat="VP"] & #np:[cat="NP"]
      $ (#vp >* #np)
```

Figure 4: TigerSearch Queries; (*) Queries are approximations and may not produce the correct output

these so-called "syntax graphs" (König and Lezius, 2001). Query graphs are built using two main relations: immediate dominance (>) and immediate precedence (.). Closures of these relations (>* and .* respectively) and other node relations such sibling ($) and left and right corners (>@l,>@r) increase expressiveness. However, intersecting hierarchies are not supported (Q6).

A precedes B (i.e. A .. B) if the left corner of A precedes the left corner of B. Immediate precedence means the distance between left corners is 1. Queries requiring immediate precedence (e.g. Q4) will not be correctly described using this relation. Left and right corners can be used to define the query we want (cf. fig 4). However, this mimics the Tgrep2 precedence definition and may fail if the syntax graph has crossing branches.

Nodes are implicitly existentially quantified before the graph description. This means non-inclusion in Q2 will fail unless the negated node exists in the graph. The expression of Q5 is incorrect as the non-existence of a lower common ancestor cannot be established. The closest approximation will include all common ancestors. However, we can use the (somewhat unintuitive) fact that a rightmost child of a node must dominate the right corner of that node to formulate Q3.

The corpus of syntax graphs is indexed before querying. This index contains inferred facts about the corpus graphs with respect to TIGERSearch relations and predicates. In effect, the corpus data becomes a prolog fact database. The query processor attempts to unify elements of corpus graphs with

```
Q1.  [Syntax=S ^ Word=saw]
Q2.* Not expressible
Q3.  end(Syntax=NP, Syntax=N)=1
Q4.* [Syntax=VP ^ [Syntax=V ->
          [Syntax=NP -> Syntax PP] ]]
Q5.* [Syntax!=x ^ [Syntax=NP -> Syntax=VP]]
Q6.  [Syntax=NP ^
           [Word=dark ^ intermediate=L-]]
Q7.? [Syntax=VP ^ #Syntax=NP]
```

Figure 5: Emu Queries

the query graph. Exhaustive search is avoided using label based filtering and by re-ordering the query graph traversal.

Matching syntax graphs are returned in their entirety. This means output reduction of Q7 cannot be done.

2.3 Emu Query Language

The Emu speech database system (Cassidy and Harrington, 2001) defines an annotation scheme involving temporal constraints of precedence and overlap. Emu annotations are stratified into levels; each level is an interval structure, and elements on each level overlap those on the same and on other levels. The overlap relation is called dominance in the Emu documentation, but it is a reflexive, symmetric and non-transitive relation best understood as temporal overlap. Given this approach to dominance, nodes in an Emu structure can be "dominated" by multiple parent nodes. Hence Emu claims to support multiple intersecting hierarchies. Example Emu queries are given in Figure 5.

Built in functions start(), end(), mid() and position() allow expression of the positional constraint in Q3. However lack of precision in dominance and precedence relations is a problem when dealing with syntax. Immediate dominance is only expressible between the appropriate levels. However, syntax trees do not easily split into such identifiable levels. Precedence is only defined for nodes on the same level. There is no way to describe immediate precedence within a level. Negated relations are not possible hence Q2 type non-inclusion cannot be expressed either.

Each query only returns one node per match. The target node can be specified by the user (Q7). However, this is unhelpful if structure is of interest. It also prevents query composition. Query constraints are tested in all possible match positions in the structure. This is satisfactory for small data sets but does not scale up well.

```
Q1.  node: S
     query: saw exists
Q2.  node: S
     query: !saw exists
Q3.  node: NP
     query: NP iDomsLast1 N
Q4.? node: VP
     query: V iprecedes NP
        and NP iprecedes PP
Q5.* Not expressible
Q6.* Not expressible
Q7.  node: VP
     query: exists NP
     node: NP
     query: exists *
```

Figure 6: CorpusSearch Queries

2.4 CorpusSearch

CorpusSearch was developed for the Penn-Helsinki Parsed Corpus of Middle English, though it can be used with any corpus annotated in the Penn Treebank style. An interesting feature of this language is that queries are limited in scope to subtree rooted by a specific type of node. In Q1 `exists` asks that the word 'saw' exists in a subtree rooted with an 'S'. Surprisingly, the search function (relation) `dominates` has been discontinued. This means nested dominance queries cannot be expressed. Precedence is only defined among siblings which greatly restricts the number of queries possible. Q4, for example, will miss many hits.

Negation of dominance has the semantics required for Q2. However, `VP precedes !NP` will not match verb phrases that are rightmost amongst their siblings. Search functions such as `iDomsLast` exist to express this sort of positional constraint instead. Wildcards can be specified, however the lack of a dominance search function means Q5 cannot be expressed. Multiple domination, as in Q6, is not supported.

A striking feature of this language is that it treats regular expressions over node names as variable names. Thus `A iprecedes B*|C` and `B*|C iprecedes D` describes a sequence of three nodes, while `A iprecedes B*|C and C|B* iprecedes D` describes two unrelated sequences, each of two nodes.

CorpusSearch is compositional. This allows Q7 to be specified in two stages.

2.5 NiteQL

NiteQL extends the MATE workbench query language Q4M (McKelvie et al., 2001) and has been released as part of the NITE XML Toolkit (Heid et al., 2004). Queries consist of weakly typed variable declarations followed by match conditions. Matches are evaluated over attribute, structure and time constraints. Precedence can be defined by the application designer depending on the data model.

```
Q1. ($s syntax) ($w word):
       ($w@orth=="saw") && ($s@cat=="S")
        && ($s ^ $w)
Q2.*($s syntax) ($w word):
       ($w@orth=="saw") && ($s@cat=="S")
        && !($s ^ $w)
Q3. ($np cat) ($w word) :
       ($np@cat=="NP") && ($w@pos=="N")
        && ($np ^1[-1] $w)
Q4. ($vp syntax) ($v word)
    ($np syntax) ($pp syntax):
        ($v@pos=="V") && ($np@cat=="NP")
        && ($pp@cat=="PP")
        && ($vp@cat=="VP")
        && ($v <>1 $np) && ($np <>1 $pp)
        && ($vp ^ $v) && ($vp ^ $np)
        && ($vp ^ $pp)
Q5.*($vp syntax) ($np syntax) ($x syntax):
        ($vp@cat=="VP") && ($np@cat=="NP")
        && ($x ^ $vp) && ($x ^ $np)
        && ($np <> $vp)
Q6. ($s syntax) ($i intermediate)
    ($w word):
       ($s@cat=="NP") && ($i@tone=="L-")
        && ($w@orth=="dark")
        && ($s ^ $w) && ($i ^ $w)
Q7. (exists $vp syntax) ($np syntax):
       ($vp@cat=="VP") && ($np@cat=="NP")
        && ($vp ^ $np)
```

Figure 7: NiteQL Queries

The queries in Figure 7 assume the model used by Tgrep2. If crossing branches are permitted, and the left corners precedence definition is used, then NiteQL will behave like TIGERSearch instead. Dominance and precedence relations can take modifiers that provide more positional constraints. In Q3, `^1` indicates immediate dominance and `[-1]` indicates rightmost descendant. Any sibling position at any level can be specified.

Like TIGERSearch, variables are existentially quantified when declared so Q2 and Q5 cannot be correctly expressed. Quantification (`exists`, `forall`) can be used in variable declarations. However their main purpose is to suppress marked in query output as used in Q7.

The NITE project encourages storage in standoff XML. An XPath-like pointer relation enables the formation of secondary (non-tree) edges between

```
Q1.  /S[//_[@lex = 'saw']]
Q2.  /S[not //_[@lex = 'saw']]
Q3.  //NP{/N$}
Q4   //VP{V -> NP -> PP}
Q5.? //_{//NP --> VP}/
        ancestor-or-self::*[1]
Q6.* Not expressible
Q7.  //VP/NP
```

Figure 8: LPath Queries

nodes. These can also occur between separate hierarchies. The final output is an XML document listing pointers to matches in the corpus. This is useful for searches during the annotation process. NiteQL's type system allows queries on intersecting hierarchies as seen in Q6. Complex queries provide compositionality and can be used to structure results to some extent.

2.6 LPath

LPath is a path language for linguistic trees extending XPath, with with an immediate precedence and a scoping operator (Bird et al., 2004). LPath queries can be translated into SQL for efficient evaluation. The LPath versions of the example queries are shown in Figure 8. The LPath representation of Q5 follows from node set selection in XPath. The positional predicate `[1]` is applied as the ancestor axis is traversed. In this case, only the first node in *reverse document order* is selected. This is the first common ancestor required by the query.

3 Requirements for Tree Query

Tree query languages need to be able to express node relationships succinctly. The languages we have surveyed needed more than dominance and precedence relations to express the specialized relations invoked in the example queries. A reasonable definition of immediate precedence is clearly necessary. However, more is required than positive descriptions. Non-inclusion queries such as Q2 and Q5 could not be expressed in all languages.

3.1 Simple Navigation

Subtree Matching. A vital part of subtree matching is accurate specification of the query tree. An inventory of subtree description types is given in Figure 9. Subtree description may also be facilitated with a graphical interface that maps tree diagrams to expressions in a query language. This is an attractive option for non-computer scientists and already exists in tools such as TIGERSearch.

Immediate dominance:
A dominates B, A may dominate other nodes

Positional constraint:
A dominates B, and B is the first (last) child of A

Positional constraint with respect to a label:
A dominates B, and B is the last B child of A

Multiple dominance:
A dominates both B and C, but the order of B and C is unspecified.

Sibling precedence:
A dominates both B and C, B precedes C; A dominates both B and C, B immediately precedes C

Complete description:
A dominates B and C, in that order, and nothing else

Multiple copies:
A dominates B and B, and the two Bs are different instances

Negation:
A does not dominate node with label B

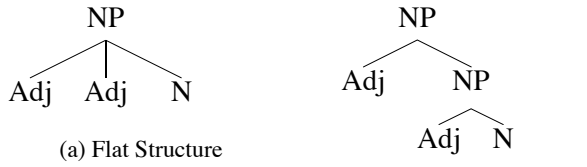
Figure 9: Subtree Matching Queries

Returning subtrees. The query languages had some capacity to constrain the output of a query as was shown in example query Q7. However, only NiteQL could choose specific nodes (as opposed to subtree roots) to output. This sort of precise reduction needs to be further supported.

Reverse navigation. Query specifications tend to reflect top down, left to right tree navigation. On the other hand, context can occur in any direction from a node. This is a problem for languages such as Tgrep2 where graph description focuses on one particular node at a time. Reverse relations, such as 'follows', are implemented in Tgrep2 and are necessary for queries such as Q5. However, reverse relations have less value in a language like TIGERSearch where the ordering of graph description is not important. In terms of expressiveness, necessity of these relations depends on the query structure.

However, reverse navigation is relevant for developing a matching strategy. For example, matching an ancestor only requires nodes on the path from the current node to the root. Strategies that allow reduction of the search space need to be employed.

Non-tree navigation. Queries are not always described in terms of edge traversal structure. Queries are often specified over the sequence of terminals (i.e. the text), regardless of hierarchical organization. Example queries have shown that this notion of sequential navigation needs to be extended to non-terminal nodes, e.g. to permit searching for sequences of one or more adjective



(a) Flat Structure

(b) Chomsky Adjunction

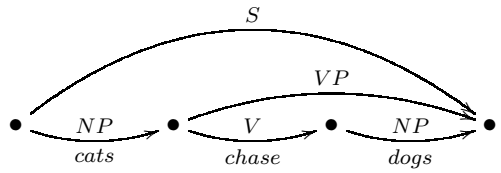
Figure 10: Two Representations for Optional, Repeatable Constituents

S → NP VP
VP → V NP
NP → cats
NP → mice
V → chase

(a) Phrase Structure Grammar

cats chase mice
NP chase mice
cats V mice
cats chase NP
NP V mice
NP chase NP
cats V NP
cats VP
NP VP
NP V NP
S

(b) Proper Analyses



(c) Syntactic Chart

Figure 11: Immediate Precedence in Linguistic Trees

followed by a noun, regardless of the internal organization of the noun phrase (cf. Figure 10). All surveyed languages support some notion of precedence though not all allow for immediate precedence. Immediate precedence in a tree can be conceived in terms of the "proper analyses" of early generative grammar (Chomsky, 1963) or syntactic charts, as shown in Figure 11.

3.2 Closures

The surveyed languages include closures of basic relations such as dominance, precedence and sibling precedence. These are necessary as distance between nodes of interest can be arbitrarily large. These are generally represented as separate relations in tree querying languages. However, closures are required of more complicated structures that are not handled currently.

We may wish to find parts of a corpus that fit a particular grammatical theory. For example, Chomsky adjunction in Figure 10 can be described by the productions: `NP` → `Adj NP`, `NP` → `N`. This translates to an LPath-like expression `(/NP[/Adj])*/N`. Here closure is atomic and each repetition involves a single step along some axis. In general we would like to be able to express any self-recursive rule $\texttt{A} \rightarrow \texttt{L}_1 \cdots \texttt{L}_n\texttt{A}\texttt{R}_1 \cdots \texttt{R}_n$ using a closure. In LPath this might possibly be specified by `(/A[<=Ln...<=L1, =>R1...=>Rn])+`.

Closures involving more than one step are also required. For example, a path consisting of alternating `VP` and `S`'s: `(/VP/S)+`. Outside of tree navigation we wish to find regular sequences such as consonants and vowels: `(->C->V)+`. Moreover some structure may best be described using nested closures, e.g. `((->C)+->V)+`.

Q5 represents a class of queries that ask for the first common ancestor of tree fragment. However, negation semantics means this can only be specified in Tgrep2 and LPath. This problem may be addressed more generally with the greedy matching approach of regular expression processing. That is, signal that we want the first match only. However, this is not compositional.

3.3 Beyond ordered trees

Queries may need to extend beyond sentence boundaries. For example, anaphoric arguments may occur in previous sentences (Prasad et al., 2004). If trees represent sentences and querying is restricted to subtree matching this is a problem. One solution is to include multiple sentences in trees. However, this drastically increases the size of trees. Query trees are generally very small (if spread widely) so massive trees decrease filter effectiveness during query processing and have a bad effect on matching algorithms.

This presents a good case for querying over ordered forests. In fact this is necessary when querying the Verbmobil treebanks of spontaneous speech (Hinrichs et al., 2000). Here discourse turns are modelled to include repetitions, interjections, disfluencies and sentence fragments. These are represented as trees disconnected from surrounding well-formed sentences. Trees can occur wrapped in other trees as seen in Figure 12. VIQTORYA (Steiner and Kallmeyer, 2002) is a query language developed for these treebanks. However, this can be considered a subset of the TIGERSearch language so was not discussed in the survey.

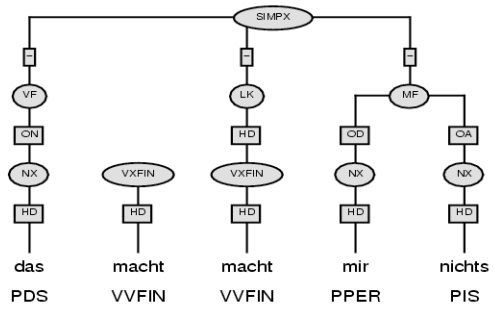


Figure 12: Forest representation of the Verbmobil corpus (Steiner and Kallmeyer, 2002)

There is a general need to move beyond single tree searches and integrate different types of linguistic data. Querying intersecting hierarchies has been well motivated by the workbenches such as Emu and the NITE project. There is also a need to query over relational and structural data. (e.g. Switchboard Treebank). We may want to match subtrees depending on the attributes of a word stored elsewhere (e.g. verb class in dictionary). Scope for these types of queries needs to be included in query language development.

Beyond this, there is a need to query non-tree structure. For example, Penn Treebank and the TIGER corpus includes secondary edges. It would be useful to navigate these links to extract information about the long range phenomena stored there. This means a definite move from tree based models which needs to be explored further.

3.4 Update

Curating a corpus of trees requires frequent updates. Tree edits often describe restructuring of constituents. For example, transforming structure to (resp. from) a small clause representation involves insertion (resp. deletion) of a common parent. Changing annotation style to reflect X-bar theory involves relabelling certain `NP` nodes to `N'`. Another useful transform is to reattach a phrasal adjunct to a higher level node, which calls for a notion of subtree movement.

Insertion, deletion and relabelling nodes are standard tree editing operations. However, linguistic trees are more constrained than general trees. Freedom of movement of constituents almost always depends on preserving the base text. Subtree deletion is not allowed (except zero-width elements) nor is re-ordering of leaves. Any subtree can only legally move to a limited number of locations without perturbing the text.

Subtree movement can be described in terms of node insertion and deletion. However, this will be extremely tedious for the user to specify as subtrees may be extremely large. Thus subtree movement should appear as a basic operation. (Cotton and Bird, 2002) present a tree edit operations all in terms of node movement of a distinguished node. The direction and surrounding structure determines where the node is reattached. Further operations are required to deal correctly with empty constituents. All update operations should have inverses so edits can be reversed.

Syntactically annotated corpora are often annotated with respect to a particular grammar. These grammars may be updated and annotations need to be changed to reflect this. However, it is inefficient to reannotate the entire corpus every time this happens. A useful update mechanism should be able to compare grammars and then implement changes only where necessary. The closures described previously will be useful here.

4 Conclusion

Several linguistic tree languages have been proposed in earlier work, and we have investigated their expressiveness and conciseness for a range of practical queries. Our survey has led us to propose a number of requirements for any general-purpose linguistic tree query language. They should permit hierarchical and sequential navigation, including an immediate precedence relation which cuts across the hierarchy. They should go beyond simple subtree matching to support a range of closures which correspond to grammar fragments, and positive and negative constraints on context. Whether as intersecting hierarchies or ordered forests, multiple tree querying must also be developed further. Requirements for a tree update language derive from a need to maintain the underlying text and present natural edit operations to the user.

We have broached several topics which require further investigation. Query languages incorporating variables, quantification, negation, and closures need to be better understood. This can be done by manually translating such queries to a language of first order logic or modal logic, exploring the kinds of nested quantification required, and consequences for implementation. The existing, well-understood relational and semi-structured query languages and finite automata could play a similar role.

Query expressions which can be mapped to the forwards and downwards subset of tree navigations are amenable to implementation in a streaming processor, opening the way for a true tree-grep tool which is able to function in a pipeline mode on

unpreprocessed treebank files. Other areas of further work include an exploration of an appropriate typing system as used to navigate intersecting hierarchies; investigation of boundaries for contextual search; and the interaction of indexing and updates.

Acknowledgements

This research has been supported by an Australian Postgraduate Award (Lai), and a US National Science Foundation grant number 0317826 *Querying Linguistic Databases*, to the University of Pennsylvania (Bird).

References

S. Bird, Y. Chen, S. Davidson, H. Lee, and Y. Zheng. 2004. LPath: A path language for linguistic trees. Unpublished manuscript.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories Sozopol*.

S. Cassidy and J. Harrington. 2001. Multi-level annotation in the Emu speech database management system. *Speech Communication*, 33(1-2):61–77.

N. Chomsky. 1963. Formal properties of grammars. In D. Luce, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume 2, pages 323–418. New York: Wiley and Sons.

S. Cotton and S. Bird. 2002. An Integrated Framework for Treebanks and Multilayer Annotations. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1670–1677. ELRA.

U. Heid, H. Voormann, J-T Milde, U. Gut, K. Erk, and S. Pado. 2004. Querying both time-aligned and hierarchical corpora with nxt search. In *Fourth Language Resources and Evaluation Conference, Lisbon, Portugal*.

E. W. Hinrichs, J. Bartels, Y. Kawata, and V. Kordoni. 2000. The VERBMOBIL Treebanks. In *KONVENS 2000 Sprachkommunikation, ITG-Fachbericht 161*, pages 107–112. VDE Verlag.

E. König and W. Lezius. 2001. The TIGER language - a description language for syntax graphs. Part 1: User's guidelines. Technical report, University of Stuttgart, Stuttgart, Germany.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

D. McKelvie, A. Isard, A. Mengel, M. B. Moller, M. Gross, and M. Klein. 2001. The MATE workbench — an annotation tool for XML coded speech corpora. *Speech Communication*, 33(1-2):97–112.

R. Prasad, E. Miltsakaki, A. Joshi, and B. Webber. 2004. Annotation and Data Mining of the Penn Discourse TreeBank. In *Proceedings of the ACL Workshop on Discourse Annotation Barcelona, Spain*.

D. Rohde. 2001. Tgrep2 user manual.

I. Steiner and L. Kallmeyer. 2002. VIQTORYA – A Visual Query Tool for Syntactically Annotated Corpora. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1704–1711. ELRA.

Converting the Penn Treebank to Systemic Functional Grammar

Matthew Honnibal
Department of Linguistics, Macquarie University
Macquarie University
2109 Sydney
Australia
mhonn@it.usyd.edu.au

Abstract

Systemic functional linguistics offers a grammar that is semantically organised, so that salient grammatical choices are made explicit. This paper describes the explication of these choices through the conversion of the Penn Treebank into a systemic functional grammar corpus. Developing such a resource can help connect work in natural language processing to a significant body of research dealing explicitly with the issue of how lexical and grammatical selections create meaning.

1 Introduction

The Penn Treebank was designed to maximise consistency and annotator efficiency, rather than conformity with any particular linguistic theory (Marcus et al., 1994). This results in trees that strongly suggest the use of synthetic features to explicate semantically significant grammatical choices like *mood*, *tense*, *voice* or *negation*. These distinctions lie latent in the configuration of the tree in the Treebank II annotation scheme, making it difficult for a machine learner to make use of them.

Rather than the ad hoc addition of this information at the feature extraction stage, the corpus can be re-presented in a way that makes feature extraction more principled. This involves increasing the size and complexity of the representation of a sentence by organising the tree semantically. Organising a grammar semantically is by no means a trivial task, and has been an active area of linguistic research for the last forty years. This paper describes the conversion of the Penn Treebank into a prominent output of such research, systemic functional grammar (SFG).

Systemic functional grammar does not confine its description to syntactic structure, but includes a representation of the choices grammatical configurations represent — or 'realise', to use the term preferred in the linguistics literature (Halliday, 1976).

There is growing evidence that systemic functional grammar can be usefully applied to natural language processing (Munro, 2003; Couchman and Whitelaw, 2003), and there is a strong history of interaction between systemic functional linguistics and natural language generation (Matthiessen and Bateman, 1991). However, there is currently a lack of computational SFG resources. There is no standard format for machine readable annotation, no annotated corpora, and no useable parsers. Converting the Penn Treebank will make a large body of SFG annotated data available to computational linguists for the first time, an important step towards addressing this situation.

We first discuss some preliminaries relating to the nature of systemic functional grammar, and the scope of the converted corpus's annotation. We then discuss the conversion of the treebank's phrase-structure representation to SFG constituency structure, and finally we discuss the addition of interpersonal and textual function structures.

2 Some preliminaries

2.1 Structure of the SFG analysis

Systemic functional grammar divides the task of grammatical analysis — the process of stating the grammatical properties of a text — into two parts: analysis of syntactic structures, and analysis of function structures.

SFG syntactic analysis is constituency based, and is predicated on Halliday's notion of the rank scale (Halliday, 1966): clauses are composed of groups/phrases, which are composed of words, which are composed of morphemes. The main concerns of SFG syntactic analysis are the chunking of words into groups/phrases, and the chunking of groups/phrases into clauses. Levels of constituency between groups/phrases and their words are recognised in the literature (Matthiessen, 1995), but rarely brought into focus in research unless the group/phrase contains, or is, an embedded constituent from another rank (e.g., a nominal group like 'the man' with an embedded relative clause like 'who knew too much').

Function structures can refer to any rank of the constituency, but clause rank functional analysis is generally regarded as the most important. The grammar defines a set of systems, which can be defined recursively using conjunction and disjunction. They are usually represented graphically in system networks (Matthiessen, 1995), as in Figure 1.

In this figure, the nested disjunction 'indicative or interrogative' represents a more delicate, or finer grained, distinction than that between indicative and imperative. After selecting from the initial choice, one proceeds from left to right into increasingly delicate distinctions. These systems are categorised into three metafunctions, which represent different types of meaning language enacts simultaneously (ideational, interpersonal and textual) (Halliday, 1969).

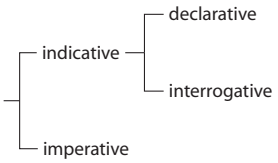


Figure 1: A simple *mood* system, '(indicative or interrogative) or imperative'

2.2 Scope of target annotation

There is no clearly defined limit to systemic functional grammar, in the sense that one could say that a text has been 'fully' analysed. The grammar is constantly being extended, with new kinds of analysis and levels of delicacy suggested. The ultimate aim of the approach is to distinguish every semantically distinct different wording choice (Hasan, 1987).

When working with systemic functional grammar, then, practitioners generally define the scope of their analysis. We must do the same, although the reasons are different. Analysis, so far, has always been performed manually, with only finite time available. Projects have therefore had to decide between the size of a sample and the detail of its analysis. In our case, we are limited to the kinds of analysis which can be directly inferred from the Penn Treebank. Future research will doubtless leverage other resources to extend the analysis of the corpus we present, but attempts to do so are beyond the scope of this paper.

The Penn Treebank presents accurate constituency and part-of-speech information. This is enough information to annotate the corpus automatically with roughly two thirds of the most important clause rank systems: *mood* and *theme*, but not *transitivity*.

The distinction between systems which can be automatically annotated and systems which cannot lies in the way the systems are realised. *Mood* and *theme* are realised primarily through the order of constituents (the order of Subject and Finite in the case of *mood*, and the first Adjunct, Subject, Complement or Predicator in the case of *theme*). They are realised structurally, as opposed to lexically. Other systems are realised through the selection of grammatical items (also called 'function words' — a term we prefer not to use because of the special sense of 'function' in the context of SFG).

Systems that are realised with grammatical items, such as *voice*, *polarity* and *tense*, can also be automatically annotated. Lexically realised systems, on the other hand, require a lexicon or equivalent resource, since the choice of words within identical syntactic structures changes the selection from the system. Trees which are identical at every level except their leaves have different *process type* selections. The central system of transitivity, process type, cannot be analysed for this reason.

The annotation of the corpus we present therefore attempts to include selections from the following systems at clause rank:

- interpersonal
  - mood (i.e. mood type and role tags for Subject, Finite, Predicator, Adjunct, Complement, Vocative)
  - clause class
  - status
  - tense
  - polarity
- textual
  - theme (i.e. role tags for Textual Theme, Interpersonal Theme, Topical Theme, Rheme)
  - voice

Ideational analysis is omitted entirely, because transitivity analysis requires a more complicated approach, as discussed above. Although arguably some aspects of taxis and expansion type could be annotated automatically, because the central information cannot be annotated, we have left it out entirely.

3 Constituency Conversion

We have not found it necessary to use a method of automatic rule induction to generate a CFG. The

lack of a suitable training set made that approach impractical for the time and resources we have had available; and good results have been obtained by simply using a set of hard-coded transformation functions, implemented as a Python script. This approach does have a significant drawback, however: because the script does not output a conversion grammar, correcting systematic errors and other maintenance or extension tasks are much more difficult.

The first process in the conversion of a sentence is to parse the Lisp-style string representation into a tree of generic node objects. Each node contains a function tag (which may be null), a node label and a set of children (which may be empty). The root node is then used to initialise a sentence object, which sorts its immediate children into clause, group and verbal group objects. As each class is initialised, it initialises a clause, verbal group, other group or lexis object with each of its children. The tree is thus recursively re-represented by more specific constituent objects, rather than generic node objects. Subtyping the nodes facilitates the changes to the structure that must be performed, since the structural changes are mostly specific to either verbal groups or clauses.

These changes are divided into a series of steps, each coded as a function. Each function contains a series of conditionals which identify the structure being targeted and how it should be altered. The most significant functions are described in more detail below. This is not an exhaustive list, however, as several trivial changes have been omitted. These include things like node relabelling and the addition of group nodes for conjunctions. There are many changes of this sort, some introduced by the specific mechanics of altering the tree. They are not generally interesting differences between the constituency representations of the Treebank's phrase-structure representation and systemic functional grammar.

3.1 Raising verb phrase predicates

The most obvious difference between SFG constituency and the Treebank II annotation scheme is the flatter, 'minimal bracketing' style SFG uses. To convert a tree to SFG clause constituency, all complements and adjuncts must be raised by attaching them to the clause node; in the Treebank annotation they attach to the verb. Figure 2 illustrates the raising of clause constituents from the verb phrase.

3.2 Raising hypotactic clauses

SFG represents the distinction between hypotaxis and parataxis with features, rather than tree structure. All non-nominalised, non-embedded clauses are therefore siblings dominated by the root clause complex.

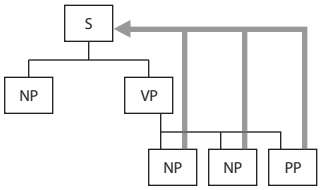


Figure 2: Raising of NP and PP nodes dominated by a VP

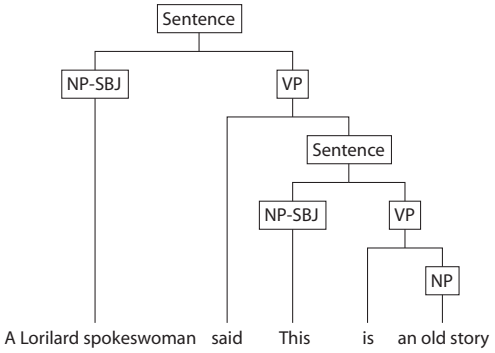


Figure 3: A clause dominating another

Figure 3 shows the Treebank representation, with a hypotactic clause as a child of a VP. Hypotactic clauses are raised to be siblings of the nearest clause node above them. Figure 4 shows the tree after this has been performed.

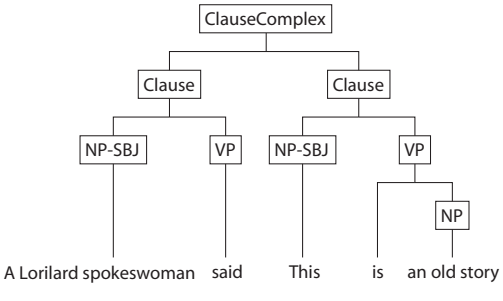


Figure 4: Equally ranked clauses

3.3 Flattening auxiliaries

In the Treebank II annotation scheme, each auxiliary — and the main verb — is given its own node, dominated by the auxiliary before it. This structure needs to be flattened to match the SFG representation. If all of a verb phrase's lexical items have POS tags in the following list: VB, VBD, VBG, VBN, VBP, VBZ; and it only has one verb phrase child, then its lexis attaches to the verb phrase below it. The empty internal node will later be removed in

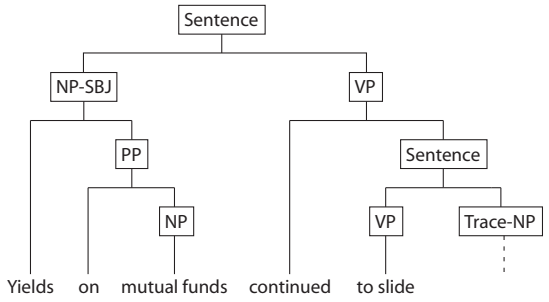


Figure 5: Treebank representation of a sentence that contains a verbal group complex

the generic 'flattening' stage.

3.4 Verbal group complexing

SFG distinguishes between clause complexes and verbal group complexes. The rules for parsing a tree as one or the other type of construction are quite simple.

If a verb phrase has one verb phrase child, and dominates a lexis node that is not a finite, then it is treated as a verbal group complex. Additionally, if a verb phrase has a sentence child that is not a direct quotation, does not have the function tag PRN (parenthetical), and is not labelled SBAR (used for relative and subordinate clauses), it is treated as a verbal group complex. For example, SFG renders the tree in Figure 5 as a single clause, with the verbal group "(continued) (to slide)".

Group and phrase complexing is actually represented a little inaccurately in the script. Ideally, a structural Complex node should be created, and all groups attached to it. This representation would mirror the way clause complexing is handled. Instead, group or phrase complexing is treated like rank-shifting, with the first group dominating the others. This concern is not crucial, however, since it does not affect the clause division or the annotation of function structures.

3.5 Ellipsis

Ellipsis was the most difficult case to deal with, since it involves more than just relocating nodes in the tree. A new clause is created when a verb phrase is identified as part of a clause with an ellipsed subject. The verb phrase is moved to the new clause, along with all of its children, and any items identified as ellipsed are copied and attached. Lexis that is copied in this way must be renumbered, so that the clause sorts properly.

When a verb phrase has two or more verb phrase children, each verb phrase child after the first is moved to a new clause. Figure 6 shows the structure of a sentence containing an ellipsed clause. The siblings of the dominant verb phrase (such as the subject), all lexis of the dominant verb phrase (such as the finite), and all children of the ellipsed verb phrase (such as the complement) are copied to the new clauses. In effect, the only items in the 'original' clause that are not in the 'ellipsis' clauses are children of the first verb phrase (such as the adverbial phrase).

It is not entirely clear that copying the words is the best solution. A trace — an empty group that simply references the original version — is possibly more convenient. The trace solution is more convenient when using the corpus as training data for a computational linguistics task, while copying the elements makes the corpus easier to use for linguistic research. The SFG literature is unhelpful for these kinds of decisions: it is concerned with content descriptions, not representation descriptions.

3.6 Pruning and truncating

Lexical nodes that contain only punctuation or traces are pruned from the tree. Group nodes that contain no lexis are also pruned. This operation is performed recursively, from the bottom up, clearing away any branches that have no lexical leaves. Internal nodes that contain only one child are replaced by that child, truncating non-branching arcs of the tree.

The clearance of punctuation is a problem with the script as it currently stands, since clearly this information should not be lost.

4 Adding Metafunctional Analysis

Function structures must be added after the constituency conversion. The structures attach to clauses in the constituency tree, making separation into clauses essential before systems can be annotated.

Function structures fall into two categories: metafunctional roles, and systems. Metafunctional roles describe the interpersonal, textual or ideational function of a particular constituent, which is considered the role's *realisation*. Systems are instead disjunctions from which a term is selected if the entry condition is met. The names of metafunctional roles are generally capitalised in the literature, while system names are given in italics. We follow this convention to help make the distinction clearer.

As with the constituency conversion, function structures were added by hard-coded functions, implemented as a Python script. Four kinds of information are used for metafunctional analysis:

1. The Penn Treebank's function tags
2. The Penn Treebank's POS tags

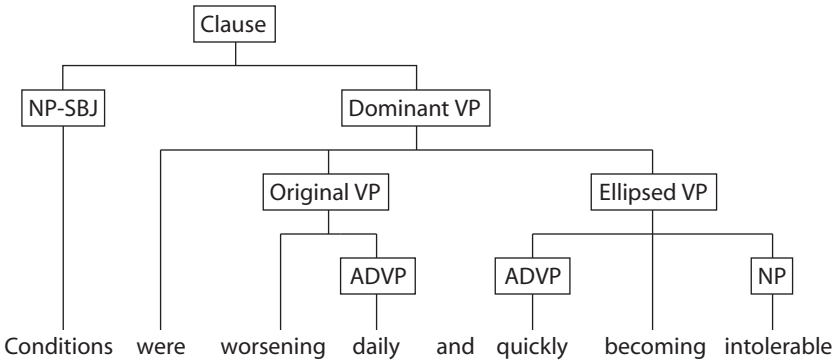


Figure 6: Treebank representation of an ellipsed clause, with verb phrases named

3. The value of other systems
4. The order of constituents in the SFG representation

The use of values from other systems makes the annotation procedure order dependent. They are usually used to determine whether a system's entry condition has been met. For instance, *tense* is not selected by non-finite clauses — so the function that discerns *tense* first checks the that requirement, and assigns null *tense* if the clause has no Finite.

The subsections below give a brief linguistic description of the system being annotated, and then describe the way its selection is calculated. If the entry condition is not met, the selection is considered 'none'.

4.1 Class

Class is an interpersonal system with the possible values 'major' and 'minor'. Major clauses are those with a verbal group. Minor clauses are equivalent to sentence fragments in other grammatical theories. An example from the Penn Treebank is the fragment "Not this year."

If a clause contains a verbal group, it is marked 'major clause'. If it has no verbal group, it is marked 'minor clause'.

4.2 Finite

Finite is an interpersonal role. The Finite is the *tense* marker of a verbal group. It is either the first auxiliary, or it is included with the lexical verb as a morphological suffix. The Finite is a significant unit of the grammar, because the placement of it in relation to the Subject realises *mood type*, and its morphology realises *tense selection* and number agreement with the Subject.

If a clause is minor *class*, or the first word of its verbal group has one of the following POS tags: TO, VBG, VBN; then it does not contain a Finite. Otherwise the first word of the verbal group receives the interpersonal role Finite.

4.3 Predicator

Predicator is an interpersonal role. The Predicator is the lexical verb of a verbal group.

If a clause is minor *class*, it does not contain a Predicator. Otherwise, the last word of the verbal group receives the interpersonal role Predicator. If a verbal group has only one word, that word will therefore receive two interpersonal roles (Finite and Predicator). This is the analysis recommended in the literature (Halliday, 1994).

4.4 Status

Status is an interpersonal system with the possible values 'free' and 'bound'. *Status* refers to whether a clause is 'independent' or 'dependant', to use the terms from traditional grammar.

Minor clauses do not select from the *status* system, so receive the value 'none'. Major clauses that have no Finite, or were originally attached to another clause and were tagged SBAR, or are rank-shifted, are considered bound. All other clauses are considered free.

4.5 Subject

Subject is an interpersonal role. The Subject of a verbal group is the nominal group whose number the verbal group must agree with.

Nominal groups realising Subject are generally tagged explicitly in Treebank II annotation. The exception to this is wh- subjects like 'who', 'what' or 'which'. If no nominal group has the function tag SBJ, and there is a wh- nominal group that was not attached to the verbal group, that nominal group is considered the Subject.

In clauses with an Initiator ('I made him paint the fence'), two nominal groups will usually have been marked subject ('I', 'him'). In these cases, the first

occurring nominal group is considered the subject ('I').

4.6 Mood type

Mood type is an interpersonal system with the possible values 'declarative', 'interrogative' and 'imperative'. *Mood type* refers to whether a clause is congruently a question (interrogative), command (imperative) or statement (declarative).

Minor and bound clauses do not select from this system, and therefore receive the value 'none'. Free clauses with no subject are marked 'imperative'. Clauses with the node labels SQ or SBARQ are marked 'interrogative'. Other free clauses are marked 'declarative'.

4.7 Tense

Tense is an interpersonal system whose value is some sequence of 'present', 'past', 'future', 'modal'. *Tense* refers to the temporal positioning of the process of a clause, with respect to the time of speaking. In English, it is a serial value, because sequences of tenses can be built ('have (present) been (past) going (present)').

Finite declarative and interrogative clauses receive one or more *tense* values. The function iterates through the words of the verbal group (or the first verbal group in a verbal group complex), and assigns these values based on the words' POS tags, and in special cases their text.

If a tag is either VBD or VBN, the value 'past' is appended to the *tense* list. If the tag is either VB, VBG, VBZ or VBP, the value 'present' is appended to the *tense* list. If the tag is MD, then the text is checked. If the word is "'ll", 'will' or 'shall', the value 'future' is appended to the *tense* list. The value 'modal' is appended to the *tense* list for lexical items tagged MD. When an MD tag is seen, the next word in the list is skipped, since it will be a bare infinitive that does not represent a *tense* selection. If the lexical items 'going' or 'about' are seen, the value 'future' is appended to the *tense* list, and the next two words are skipped, as they will be 'to' and an infinitive verb. This does not occur if 'going' is the last word of the verbal group, since in that case it is the process, not a *tense* marker.

Passive clauses will have received an extra 'past' *tense* value, so when a clause is labelled passive, its last *tense* selection is removed.

4.8 Polarity

Polarity is an interpersonal system with the possible values 'positive' and 'negative'. Polarity refers to whether the verbal group is directly negated.

Polarity is the simplest system to determine, since it only involves checking the verbal group for the word "not" (or "n't"). Looking at negation more generally would be far more difficult, since it is more of a semantic motif than specific grammatical system.

4.9 Adjuncts, Complements, Vocatives

Adjunct, Complement and Vocative are interpersonal roles. Nominal groups can be either Vocatives, Adjuncts or Complements. Adjuncts represent circumstances of a clause — the where, why and when of its happening. Complements represent its non-Subject participants — the whom, to whom and for whom of its happening. Vocatives are nominal groups that name the person the clause is addressed to.

Adverbial groups, prepositional phrases and particles are always given the interpersonal function 'Adjunct'. Vocatives are explicitly marked in the Treebank, with the VOC tag. Nominal groups that realise an adverbial function are also explicitly tagged, with either TMP, DIR, LOC, MNR or PNR. Nominal groups with one of these tags receive the interpersonal role 'Adjunct'. All other non-Subject nominal groups receive the interpersonal role 'Complement'.

4.10 Voice

Voice is a textual system with the possible values 'active', 'passive' and 'middle'. *Voice* refers to whether the Subject is also the 'doer' of the clause, or whether the participants have been switched so that the Subject is the 'done to'. Compare the active clause "the dog bit the boy" with the passive version "the boy was bitten by the dog". If clauses do not have a 'done to' constituent which might have been made Subject (i.e. a Complement), they are considered 'middle' ('the boy slept').

Minor clauses do not select for *voice*, and therefore receive the value 'none'. Non-finite clauses are typed according to the POS tag of their Predicator. If the tag is VBG, *voice* is determined to be active; if the tag is VBN, *voice* is determined to be passive. Infinitive non-finite clauses receive the value 'none'.

Finite clauses with a final *tense* other than 'past' are labelled active. If the final *tense* is 'past', and the penultimate word of the verbal group is a form of the verb 'be', the clause is labelled passive, and the *tense* sequence is corrected accordingly.

Active clauses are then subtyped into true active and middle voices. Middle clauses are active clauses which have at least one complement.

4.11 Theme/Rheme

Theme and Rheme are textual roles. Theme refers to the order of information in a clause. The Theme/Rheme structure of a clause is often called Topic/Comment in other theories of grammar. The Theme is the departure point of information in a clause. The Rheme is the information not encompassed by the Theme.

The first Adjunct, Complement, Subject or Predicator that occurs is marked 'Topical Theme'. Any conjunctions that occur before it are marked 'Textual Theme', while any vocatives or finites that occur before it are marked 'Interpersonal Theme'. All other clause constituents are marked 'Rheme'.

5 Accuracy

Accuracy was checked using 100 clauses that had not been sampled while the script was being developed or debugged. Each clause was checked for constituency accuracy to the group and phrase rank — i.e., clause division and clause constituency were checked. Each of the eleven function structures were also checked: *clause class*, *status*, *mood*, *tense*, *polarity*, Subject, Finite, *voice*, Topical Theme, Textual Themes, Interpersonal Themes.

Two errors were found, both on the same clause. The *status* selection of an indirect projected speech clause was marked 'free' instead of 'bound'. This occurred because the projected clause was topicalised (i.e., it occurred before the projecting clause), which is rare for indirect speech. To correct this, the script must consider the presence or absence of quotation marks, which may be complicated by the slightly inconsistent attachment of punctuation in the Penn Treebank (Bies, 1995). Because the *status* of this clause was given as free, the clause incorrectly met the entry condition for the *mood type* system, causing the second error — a *mood type* selection of 'declarative' instead of 'none'.

In this somewhat small sample, 1198/1200 (99.83%) properties were correct, and 99% of clauses were annotated without any errors. The lack of plausible Adjunct subtyping may present problems for the accurate determination of Topical Theme in a more register varied sample, such as the Brown corpus.

Adjuncts should be subtyped into Modal Adjuncts (such as 'possibly'), Comment Adjuncts (such as 'unfortunately'), Conjunctive Adjuncts (such as 'however') and Experiential Adjuncts (such as 'quickly'). Only Experiential Adjuncts can be Topical Theme; if another kind of Adjunct occurs first it should be marked Interpersonal Theme (in the case of Modal and Comment Adjuncts), or Textual Theme (in the case of Conjunctive Adjuncts).

The Wall Street Journal corpus, which was the only section of the Penn Treebank available for this research, contains very few Mood, Comment or Conjunctive Adjuncts, so the extent of this problem could not be properly measured.

6 Conclusion

This work is approximately ten years overdue, in the sense that that is how long the resources required to perform it have existed. The motivations for it are even older: corpus linguistics has been a pillar of systemic functional linguistic research since it began, and raw text corpora are inadequate for many of the questions systemic functional linguistics asks (Honnibal, 2004). The first effort to convert the Penn Treebank to another representation was presented within months of the corpus's completion (Wang et al., 1994). Since then, treebanks have been converted to several grammatical theories (cf. (Lin, 1998; Frank et al., 2003; Watkinson and Manandhar, 2001)). It is unclear why SFG has been left behind for so long.

A corpus of over two million words of SFG constituency analysed text, annotated with the most important clause rank interpersonal and textual systems and functions, is now available. This is an important resource for linguistic research, the development of SFG parsers, and research into applying systemic linguistics to language technology problems.

Acknowledgements

I would like to thank Jon Patrick for his useful feedback on this paper. I also owe thanks to the many people who have helped me on my honours thesis, from which this paper is mostly drawn. Christian Matthiessen and Canzhong Wu have both been wonderful supervisors. Paul Nugent helped with the graphics used in this paper and my thesis, and proof-read with invaluable diligence. James Salter has shown remarkable patience over the last year and half while teaching me to program. Finally, the language technology research group at Sydney Uni have all contributed sound advice and interesting discussions on my work.

References

A. Bies. 1995. Bracketing guidelines for Treebank II style. Penn Treebank Project.

Maria Herke Couchman and Casey Whitelaw. 2003. Identifying interpersonal distance using systemic features. In *Proceedings of the first*

Australasian Language Technology Workshop (ALTW2003).

Anette Frank, Louisa Sadler, Josef van Genabith, and Andy Way, 2003. *From Treebank Resources To LFG F-Structures - Automatic F-Structure Annotation of Treebank Trees and CFGs extracted from Treebanks*. Kluwer, Dordrecht.

Michael A. K. Halliday. 1966. The concept of rank: a reply. *Journal of Linguistics*, 2(1):110–118.

Michael Halliday. 1969. Options and functions in the english clause. Brno Studies in English.

Michael Halliday. 1976. *System and Function in Language*. Oxford University Press, Oxford.

Michael Halliday. 1994. *Introduction to Functional Grammar 2nd ed*. Arnold, London.

Ruqaiya Hasan. 1987. The grammarian's dream: lexis as most delicate grammar. In Halliday and Fawcett, editors, *New developments in systemic linguistics: theory and description*. Pinter, London.

Matthew Honnibal. 2004. Design, creation and use of a systemic functional grammar annotated corpus. Macquarie University.

Dekang Lin. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the 1994 Human Language Technology Workshop*.

Christian M. I. M. Matthiessen and John A. Bateman. 1991. *"Text generation and systemic-functional linguistics: experiences from English and Japanese"*. "Frances Pinter Publishers and St. Martin's Press", "London and New York".

Christian Matthiessen. 1995. *Lexicogrammatical Cartography*. International Language Sciences Publishers, Tokyo, Taipei and Dallas.

Robert Munro. 2003. Towards the computational inference and application of a functional grammar. Sydney University.

Jong-Nae Wang, Jing-Shin Chang, and Keh-Yih Su. 1994. An automatic treebank conversion algorithm for corpus sharing. In *Meeting of the Association for Computational Linguistics*, pages 248–254.

S. Watkinson and S. Manandhar. 2001. In *Proceedings of the workshop on evaluation methodologies for language and dialogue systems*.

Selectional Preference Based Verb Sense Disambiguation Using WordNet

Patrick Ye

Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
jingy@cs.mu.oz.au

Abstract

Selectional preferences are a source of linguistic information commonly applied to the task of Word Sense Disambiguation (WSD). To date, WSD systems using selectional preferences as the main disambiguation mechanism have achieved limited success. One possible reason for this limitation is the limited number of semantic roles used in the construction of selectional preferences. This study investigates whether better performance can be achieved using the current state-of-art semantic role labelling systems, and explores alternative ways of applying selectional preferences for WSD. In this study, WordNet noun synonym sets and hypernym sets were used in the construction of selectional preferences; Semcor2.0 data was used for the training and evaluation of a support vector machine classifier and a Naive Bayes classifier.

1 Introduction

Word Sense Disambiguation (WSD) is the process of examining word tokens in a given context and specifying exactly which sense of each word is intended in that context. It has many applications in natural language processing related areas such as document retrieval, question answering, and compositional sentence analysis (Jurafsky and Martin, 2000), to name a few.

WSD systems can be roughly divided into two categories based on how the disambiguation information is obtained and applied: knowledge based and corpus based. Knowledge based systems in general require certain existing linguistic information repositories which provides all the information that can be used by the disambiguation system to distinguish different senses of the same polysemous words based on the context. Examples of knowledge based systems include dictionary based systems (Lesk, 1986) and selectional preference based systems (Resnik, 1997).

Corpus based systems in general do not require any linguistic information, instead, they require a certain amount of training data (labelled or unlabelled), and a set of predefined disambiguation features which can be used by a statistical method to train a classifier which then is used in the disambiguation of previously unseen data. A corpus based system is described in (Yarowsky, 1995).

Selectional preferences between predicating words (verbs and adjectives) and their arguments (nouns) are a type of linguistic information which has previously been combined with statistical methods to perform word sense disambiguation, ((Resnik, 1997) and (McCarthy and Carroll, 2003)). A selectional preference is a function mapping semantic-role to noun type. The basic assumption made by all selectional preference based WSD systems is that the different senses of the same predicating word would have different selectional preferences with their arguments.

As will be discussed in section 2.2, selectional preference based WSD systems developed so far are limited in terms of coverage and accuracy. In my opinion, the most important cause of this limitation is these systems' inability to extract a sufficient number of semantic roles to be used in the construction of selectional preferences. For example, if a WSD system uses only the subject of the verbs in the selectional preferences, then it cannot be expected to correctly identify the appropriate sense of the verb "run" in "John ran a race" and "John ran a restaurant", since the distinguishing feature of these two senses of "run" comes from the objects they take.

Given the difficulty of semantic role labelling, it is not surprising that only a small set of semantic roles have been used in the literature on selectional preference based WSD. However, recent developments in semantic role labelling makes it possible to extract a much richer set of semantic roles from unrestricted text, thereby enabling more complex selectional preferences to be constructed.

The main objective of this study is to investigate whether the performance of selectional preference based WSD can be improved by using the current state-of-art semantic role labelling systems. This

paper is organised as follows: section 2 will give a formal description of the research problem presented in this paper; section 3 will provide a review of some related work; section 4 will discuss the statistical methods investigated in this study and how they are combined with selectional preferences; the results of this study will be presented in section 5; and section 6 gives a conclusion of this study and some avenues for further research.

2 Background

2.1 Selectional Preference

Selectional preferences (p) are verb-sense specific. It is possible for a particular sense of a verb to have more than one selectional preference. A selectional preference of a verb-sense (s) refers to the predicate-argument structure relationship between s and its arguments. Formally, a selectional preference is a function whose domain is the finite set of semantic roles (r) and whose range is a finite set of noun types (t):

$$p(r_i) = t_j$$

For example, the first sense of the verb "eat" (eat_1: *take in solid food*) in WordNet (Miller, 1995) would have a selectional preference that requires the **subject** of the verb to be nouns of the **animate** type and the **object** of the verb to be nouns of the **food** type; whereas the fourth sense of "eat" (eat_4: *use up (resources or materials)*) would have a selectional preference which allows the subject of the verb to be of both animate type and inanimate type.

Since there does not exist a set of commonly accepted noun types, it is common for different selectional preference based WSD systems to invent their noun types.

One can draw a parallel between verb selectional preferences of natural language and function overloading of the programming language Java. In Java, two or more functions can be declared with the same name, each of these functions will have a different argument list which the Java interpreter uses at runtime to select (disambiguate) the correct function. The argument list of Java functions is an ordered list of Java object *types*. Similarly, one can treat the different senses of any verb as different functions sharing the same name, and distinguish between them based on which type of nouns are used in which semantic role of the verb.

2.2 Related Work

Resnik (1997) describes a WSD system which uses selectional preferences to train an entropy based probabilistic model classifier. Resnik defines the *prior* distribution $Pr_p(t)$ as the probability of the noun-type t occurring in a particular selectional preference p. From the prior distribution, Resnik defines the *selectional preference strength* of a particular verb sense s with respect to a particular selectional preference p over a finite set of noun types T as:

$$\begin{aligned} St_p(s) &= D(Pr_p(t|s) \| Pr_p(t)) \\ &= \sum_{t \in T} Pr_p(t|s) \log \frac{Pr_p(t|s)}{Pr_p(t)} \end{aligned}$$

From the above equation, it is obvious that the selectional preference strength of a verb sense s depends on how much mutual information the noun types of its arguments share. In other words, verb senses which take a small set of nouns as arguments are easier to disambiguate.

With the selection preference strength, Resnik further defined the *selectional association* value between a verb sense s and a noun-type t as:

$$A_p(t,s) = \frac{1}{St_p(s)} Pr_p(t|s) \log \frac{Pr_p(t|s)}{Pr_p(t)}$$

The disambiguation of a polysemous verb v using Resnik's system is therefore achieved in the following way: Suppose the noun n is an argument to a polysemous verb v; Let $[s_1, s_2, ..., s_n]$ be v's senses; let $[ns_1, ns_2, ..., ns_k]$ be n's senses; and for each ns_j, let H_j be the set of WordNet synsets which are hypernyms of ns_j; compute the following for each s_i:

$$VA(s_i) = \max_{ns_j \in H_j} A_p(s_i, ns_j)$$

Then the verb sense(s) which maximise(s) the function VA will be chosen as the most appropriate sense(s) for v. Since Resnik's system is trained and evaluated on WordNet, he used a subset of WordNet noun synsets as the noun-types of his selectional preferences. Therefore, each ns_i is a noun type.

I believe this method of choosing noun types is a weakness of Resnik's system. It is not clear from his description whether this subset of noun synsets were hand picked or computed from the available data. If these synsets were hand picked (which is the likely scenario), then the resulting system could suffer from poor coverage because it was highly unlikely that the hand picked set of noun types were complete or compatible with the WordNet noun hypernym hierarchy. To illustrate this

problem, consider the verb-object relationship between $drink_1$ (*take in liquids* and its objects: if the noun-type **beverage** ($beverage_1$) is chosen as a noun type (as it was in Resnik's paper), and the sentences "John drank wine" and "Joe drank coffee" are in the training data, since *coffee*$_1$ and *wine*$_1$ both have **beverage** as hypernym then the probability of $Pr(beverage|drink_1)$ is very likely to be high. However, if in the testing data, the system encounters the sentence "John drank some water", then because "water" does not have **beverage** as a hypernym in WordNet, it would be unlikely for the system to identify the correct sense of "drink".

On the other hand, if the subset of noun types are computed from the training data, then all the hypernyms of the nouns in the training data would also be taken into account in the estimation of $Pr_p(t|s)$. Furthermore, since the hypernym of a noun n would always describe a more general concept than n, then it is natural that the noun types describing the most general concepts would produce the highest value for the estimation of $Pr_p(t|s)$. However, the more general the noun type is, the less distinguishing feature it would be able to provide, therefore such noun types would not be effective for the WSD task.

Another weakness of Resnik's system is that the selectional preferences used in this system were constructed with only a single semantic role, e.g. the object of the verb or the subject of the verb. Therefore, these selectional preferences could only provide limited features useful for sense disambiguation.

3 Methodology

3.1 System Architecture

The system developed in this study takes semantic-role-labelled sentences as inputs and trains a classifier which can be used for the disambiguation of verbs.

The system consists of two major components: the selectional preference construction module and the classifier training and disambiguation module. When the system is given a semantic-role-labelled sentence, it first constructs the selectional preferences from the labelled semantic roles and their head nouns. These selectional preferences are then passed to the statistical classifier for the training or the disambiguation of the verb. In this study, two types of statistical classifiers were investigated: a Support Vector Machine (SVM) classifier and a Bayesian classifier.

A state-of-the-art semantic role labelling system, "ASSERT" (Pradhan et al., 2004), was used for the task of semantic role labelling. The influence of the ASSERT will be discussed in section 4. In the remainder of this section, I will give details of the two main modules of the WSD system.

3.2 Selectional Preference Construction

In this study, the WordNet **noun hypernym hierarchy** (NHH) is used to generate the noun-types used to construct the selectional preferences. A noun synset ns_a is the hypernym of another noun synset ns_b if ns_a denotes a more general concept than ns_b. A hypernym hierarchy for a noun synset ns is the tree structure which includes all the direct and indirect hypernyms of ns.

Each path from the most specific node in the NHH to the most general node is treated as a separate noun-type. For example, the NHH of the first sense of "apple" ($apple_1$: *fruit with red or yellow or green skin and sweet to tart crisp whitish flesh*) would generate the following noun-types:

t_1: ($entity_1$, $substance_1$, $solid_1$, $food_2$, $produce_1$, $edible\_fruit_1$, $apple_1$)

t_2: ($entity_1$, $object_1$, $natural\_object_1$, $plant\_part_1$, $plant\_organ_1$, $reproductive\_structure_1$, $fruit_1$, $edible\_fruit_1$, $apple_1$)

t_3: ($entity_1$, $object_1$, $natural\_object_1$, $plant\_part_1$, $plant\_organ_1$, $reproductive\_structure_1$, $fruit_1$, $pome_1$, $apple_1$)

There are two advantages of using paths extracted from WordNet NHH as the noun-types. First, it eliminates the need for a set of hand generated noun-types which is most likely to be not as comprehensive as WordNet. Second, since the noun-types are set of noun synsets of varying degrees of generality, it is possible to compute partial equality between them, this partial equality will then be applicable to the comparison between selectional preferences, thereby increasing the coverage and potentially the accuracy of the system.

To illustrate how selectional preferences are constructed from semantic-role-labelled sentences, suppose we have the following sentence:

E1 [The monkey]$_{arg0}$ [ate]$_{target}$ [an apple]$_{arg1}$

The head nouns for **arg0** and **arg1** are "monkey" and "apple" respectively. Since the system does not know which senses of these words are being used here, it will have to consider all senses of both words. In WordNet, "monkey" and "apple" correspond to the NHHs shown in figure 1.

Each **path** in the above NHHs is a noun-type. As we can see, there are 3 potential noun-types for "monkey" (**arg0**), and 4 potential noun-types for

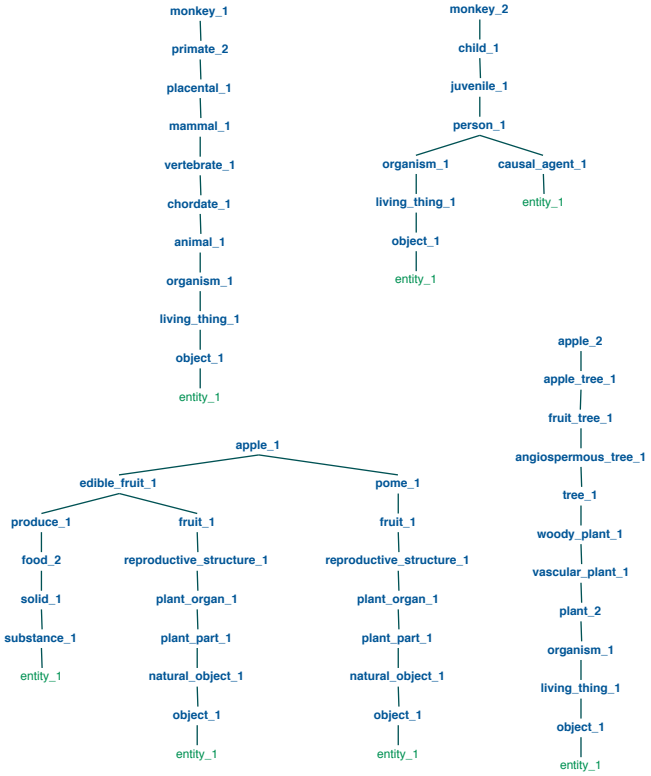


Figure 1: Example NHHs

"apple" (**arg1**). Therefore, the sentence E1 gives rise to 12 potential selectional preferences.

3.3 Training of the SVM Classifier

The SVM classifier is the first of the two classifiers investigated in this study. Since most verbs have more than two senses, the SVM classifier was trained to be multi-class, and each sense was treated as an independent class. Two types of multi-class classification were experimented: One-class-Against-the-Rest-of-the-classes and One-class-Against-One-other-class.

The attributes used in the classification are combinations of semantic role and WordNet noun synset. Recall that a selectional preference is a function mapping semantic roles to noun types; and each noun-type is a set of WordNet noun synsets. Each noun synset will be combined with its respective semantic role to form a feature. Therefore, if the total number of semantic roles is N_r and the total number of WordNet noun synsets is N_{ns}, the total number of dimensions or features is then $N_f = N_r \times N_{ns}$. During the training and the classification, all the selectional preferences generated for the **same** instance of a verb are used to create a single feature vector. If a synset ns_i appears in the noun-type for a particular semantic role r_j, then the feature corresponding to the (r_j, ns_i) tuple will have the value of 1.0, otherwise this feature will have the value of 0.0. Furthermore, all the selectional preference generated are always stored in the same feature vector. The total number of features may seem excessive, however, since the training data is unlikely to contain all the relevant selectional preferences, it is therefore necessary to include all the possible features during training and classification.

Two types of SVM kernels were experimented with in this study, linear and degree 2 polynomial.

3.4 Training of the Probabilistic Classifier

Since theoretically it is possible for a verb to take a large number of nouns for any of its semantic roles, the training of the probabilistic classifier would suffer from the data sparseness problem if no preprocessing is performed on the training data.

The preprocessing performed in this study is based on the theory of argument fusion (Jackendoff, 1990). Its main purpose is to extract common features from the noun types and give them appropriate mass in the probabilistic distribution. For example, suppose the training data consists of the following sentences for eat_1 (*take in solid food*).

S1 [The monkey]$_{arg0}$ [ate]$_{target}$ [an apple]$_{arg1}$

S2 John's dietitian allowed [him]$_{arg0}$ to [eat]$_{target}$ only [one slice of the $cake_3$]$_{arg1}$ at his birthday party.

S1 would generate the following selectional preferences:

S1_1 **arg0** $(entity_1, object_1, living\_thing_1, organism_1, animal_1, chordate_1, verebrate_1, mammal_1, placental_1, primate_2, monkey_1)$

arg1 $(entity_1, object_1, natural\_object_1, plant\_part_1, plant\_organ_1, reproduction\_structure_1, fruit_1, edible\_fruit_1, apple_1)$

S1_2 **arg0** $(entity_1, object_1, living\_thing_1, organism_1, animal_1, chordate_1, verebrate_1, mammal_1, placental_1, primate_2, monkey_1)$

arg1 $(entity_1, object_1, natural\_object_1, plant\_part_1, plant\_organ_1, reproduction\_structure_1, fruit_1, pome_1, apple_1)$

S1_3 **arg0** $(entity_1, object_1, living\_thing_1, organism_1, animal_1, chordate_1, verebrate_1, mammal_1, placental_1, primate_2, monkey_1)$

arg1 $(entity_1, substance_1, solid_1, food_2, produce_1, edible\_fruit_1, apple_1)$

S2 would generate the following selectional preferences:

S2_1 **arg0** $(entity_1, casual\_agent_1, person_1, male_2, man_1, John)$

arg1 $(entity_1, substance_1, solid_1, food_2 \; baked\_goods_1, cake_3)$

S2_2 **arg0** $(entity_1, object_1, living\_thing_1, organism_1, person_1, male_2, man_1, John)$

arg1 $(entity_1, substance_1, solid_1, food_2 \; baked\_goods_1, cake_3)$

It can be observed that some of these selectional preferences have partial overlappings among the noun-types of the same semantic roles. These overlappings capture what is in common between the examples from the training data. Intuitively, the overlappings are more suitable to be the selectional preferences than the individual training examples. For example, consider the selectional preferences generated by S1 and S2 for eat_1, one of the overlappings between them is:

S12_1 **arg0** $(entity_1, object_1, living\_thing_1, organism_1)$

arg1 $(entity_1, substance_1, solid_1, food_2)$

It is obvious that S12_1 captures almost exactly what eat_1's selectional preference really should be, namely that the subject of the verb has to be some living organism and the object of the verb has to be some kind of food. In the remainder of this paper, selectional preferences constructed through the process of argument fusion will be referred to as **fused selectional preferences**, and selectional preferences directly constructed from the training data will be referred to as **raw selectional preferences**. Since fused selectional preferences are more prototypical than the raw ones, it would make sense to give them greater mass in the final probability distribution.

Formally, the frequency of the selectional preferences are estimated in the following way:

Let rp_i be a raw selectional preference, its frequency ($C(rp_i)$) is the number of times rp_i appears in the training examples.

Let fp_j be a fused selectional preference, and let $[rp_1, rp_2, ..., rp_k]$ be the set of raw selectional preferences from which fp_j was derived, then fp_j's frequency is calculated as:

$$C(fp_j) = \sum_{i=1}^{k} C(rp_i)$$

Similarly, the conditional frequency of the selectional preference p_i given the verb sense s_j ($C(p_i|s_j)$) is estimated as the number of times p_i co-occurs with s_j.

The two frequency distributions are then used to construct the corresponding probability distributions which are then smoothed to allow far unseen data.

The classification of previously unseen data is not as simple as finding the verb sense s_i which maximises the probability of $Pr(s_i|p_j)$. Firstly, let P^c be the set of candidate selectional preferences $[p_1^c, ..., p_n^c]$ extracted with respect to an ambiguous verb v in a given context. Let P^t be the set of selectional preferences $[p_1^t, ..., p_m^t]$ from the training data. Suppose the set of senses of v is $S = [s_1, ..., s_k]$, then the most suitable sense(s) of v will be chosen in the following equation:

$$s_{max} = \underset{s_i \in S, p_j^c \in P^c}{\operatorname{argmax}} (max(Pr(s_i|p_j^c))$$

From Bayes' rule, $Pr(s_i|p_j^c)$ is calculated as follows:

$$Pr(s_i|p_j^c) = \frac{Pr(p_j^c|s_i)Pr(s_i)}{Pr(p_j^c)}$$

However, since it is very likely that p_j^c has not previously been seen in the training data, $Pr(s_i|p_j^c)$ is therefore estimated as follows:

$$Pr(s_i|p_j^c) = \max_{p_k^t \in P^t} (Pr(s_i|p_k^t) \cdot sim(p_j^c, p_k^t))$$

The function $sim(p_a, p_b)$ calculates the similarity between two given selectional preferences. Let $dom(p_a)$ and $dom(p_b)$ be the sets of semantic roles applicable to p_a and p_b respectively. Recall that in this study, a noun-type t is a **set** of WordNet noun synsets, then the function sim works in the following way:

$$sim(p_a, p_b) = \begin{cases} 0 & dom(p_a) \neq dom(p_b) \\ \sum_{r_i \in dom(p_a)} \cos(p_a(r_i), p_b(r_i)) & otherwise \end{cases}$$

4 Results

The system developed in this study was evaluated using the Semcor2.0 data and the Propbank data. Two types of baseline performances were used in the evaluation: the majority sense baseline (baseline 1) and the bag-of-synsets baseline (baseline 2).

The bag-of-synsets baseline works by first collecting the 10 nouns closest to and before the verb,

and the 10 nouns closest to and after the verb; then extracting the synsets from their WordNet noun hypernym hierarchy; and finally using these synsets as features to training a support vector machine classifier. The purpose of this baseline is to see whether the additional information provided by the semantic roles can indeed improve the performance of WSD.

Because of the diverse natural of verb selectional preferences and the different availabilities of the verb specific training data, the evaluation of the two classifiers was performed in a verb-by-verb fashion. The verbs selected for evaluation are: "bear", "eat", "kick", "look", "run", and "serve". As shown in table 1, these verbs are chosen because they represent a variety of transitivities, semantic role combinations, and different degrees of similarities between the senses. The senses of these verbs are defined in WordNet2.0.

Verb	Intran.[1]	Trans.[2]	Compl.[3]	NSR [4]
bear	no	yes	no	11
eat	yes	yes	no	2
kick	yes	yes	no	9
look	yes	no	yes	11
run	yes	yes	no	28
serve	no	yes	yes	9

Table 1: Semantic Properties of the verbs

The following classifiers were trained and evaluated:

C1 SVM classifier with a linear kernel

C2 SVM classifier with a degree 2 polynomial kernel

C3 Naive Bayes classifier using thematic role tag set

Table 2 shows the number of senses and the majority class baselines of the above verbs:

Verb	Majority Baseline 1	No. of senses
bear	31.58%	9
eat	76.27%	3
kick	45%	3
look	56.9%	8
run	33.75%	26
serve	27.81%	11

Table 2: Majority class baseline

[1]Intransitive

[2]Transitive

[3]Require Prepositional Complement

[4]Number of applicable semantic roles according to Propbank

Tables 3 to 5 show the results (Accuracy) of the above classifiers trained on 30%, 50%, and 80% of the training data:

Verbs	baseline 2	C1	C2	C3
bear	30.23	37.9	31.62	20.23
eat	75	61.5	64.5	70.25
kick	43.75	42.5	46.25	43.16
look	5.97	49.25	50.14	26.27
run	4.31	4.83	3.97	5.69
serve	12.5	36.5	38	16.58

Table 3: Classifiers accuracy(%) when 30% data was used in the training

Verbs	baseline 2	C1	C2	C3
bear	6.67	41.33	37.33	20.33
eat	7.14	60.36	63.57	65.36
kick	18.18	54.54	50.9	45.45
look	57.63	50.69	52.5	34.79
run	1.19	5	9.52	6.91
serve	12.79	4.88	4.88	14.42

Table 4: Classifiers accuracy(%) when 50% data was used in the training

Verbs	baseline 2	C1	C2	C3
bear	14.29	40.71	44.28	24.29
eat	8.33	58.33	63.33	63.33
kick	20	35	56	40
look	33.89	47.29	48.81	39.32
run	2.56	4.61	7.17	5.64
serve	7.89	8.95	7.89	13.16

Table 5: Classifiers accuracy(%) when 80% data was used in the training

The most significant feature of the results is that the three classifiers all performed below the majority class baseline. These poor results were caused by a combination of the following factors: complex sentence, poor semantic role labelling, inconsistent data, too finely defined verb senses and inadequate smoothing of the probability distributions.

The sentences used in the evaluation are generally longer than 20 words and contain embedded clauses, metaphors and ellipses. For instance, one of the examples for "eat" (eat_1) is the sentence: *"The dialogue is sharp witty and candid typical don't* **eat** *the daisies material which has stamped the author throughout her books and plays and it was obvious that the Theatre-by-the-Sea audience liked it"*. In this sentence, there is no subject/AGENT for "eat". Another example for "eat" (eat_3) is:

"No_matter that it is his troops who rape Western women and eat Western men". In this sentence, "eat" is clearly used in a metaphoric way therefore should not be interpreted literally. These complex sentences not only increase the amount of noise in the data, but also make semantic role labelling difficult. According to my estimation, less than 30% of the sentences were correctly tagged with semantic roles.

Another problem with the semantic role labelling is that it only labels noun phrases. The impact of this problem is shown by the very poor result on the verb "serve" most of whose senses require either a propositional phrase or a verb phrase as compliment. For example, the sentence "The tree stump serves as a table" is annotated as "[The tree stump]$_{agent}$ [serves]$_{target}$ as [a]$_{proposition}$ table" which is clearly wrong.

The problem caused by the excessively fine-grained senses is that these senses have very similar (sometimes identical) selectional preferences which cause inconsistency in the training data. Take eat for example, the definitions of its first and second senses are: *"take in solid food"*, and *"eat a meal; take a meal"* respectively. In the training data, in *" She was personally sloppy, and when she had colds would blow her nose in the same handkerchief all day and keep it soaking wet dangling from her waist and when she gardened she would* **eat** *dinner with dirt on her calves"*, *eat* is labelled as having the first sense, but it is labelled as having the second sense in *"Charlie* **ate** *some supper in the kitchen and went_into the TV_room to hear the news."*. This type of inconsistency causes the classifiers to sometimes behave almost randomly with respect to the relevant senses.

The problem caused by the inadequate smoothing of the probability distributions is more subtle. Given a very frequent senses s_a and a very infrequent verb sense s_b and a candidate selectional preference p_i^c , the conditional probabilities of $Pr(s_a|p_i^c)$ and $Pr(s_b|p_i^c)$ depends on the values of $Pr(p_i^c|s_a) \cdot Pr(s_a)$ and $Pr(p_i^c|s_b) \cdot Pr(s_b)$. It is often the case that there are so many selectional preferences applicable to s_a that $\frac{Pr(p_i^c|s_a)}{Pr(p_i^c|s_b)} < \frac{Pr(s_b)}{Pr(s_a)}$, thereby making the Bayes classifier assign s_b to instances of s_a. Currently, the Lidstone probability distribution with γ of 0.0001 is used by the Bayes classifier; further study is required to select a more suitable probability distribution.

Another interesting feature of the results is that the differences of the accuracy is relatively small with respect to the different amounts of data used in training. This feature is expected because one of the assumptions made by selectional preference based WSD is that each semantic role of any verb sense should be filled by nouns of similar type, in other words, nouns that have something in common. Therefore even though the amount of training data is different, the common features between the nouns of the same semantic role can still be captured and used for disambiguation.

Finally, the results also show that verbs with higher order of transitivity are easier to disambiguate. This is also not surprising because higher transitivity means more semantic roles which in turn provides more disambiguating features.

5 Conclusion and Future work

This paper has presented a study of whether the performance of selectional preference based WSD could be improved by using the current state-of-art semantic role labelling system. Although very little performance improvement was able to be achieved by the systems developed in this study, a few useful observations could be made.

First, selectional preference based WSD systems do not require a large amount of training data, as demonstrated by the previous section. Therefore, they may be more useful or more effective than corpus based WSD systems when the amount of training data is very limited or to act as a bootstrapping mechanism.

Second, to a very large degree, the performance of a selectional preference based WSD system depends on how finely the different senses of a verb are defined and the total number of semantic roles associated with the senses. As demonstrated by the "eat" example, the finer the senses are defined, the less effective selectional preference will be.

Third, the performance of selectional preference based WSD systems is heavily influenced by the quality of the semantic role identification. More importantly, it is not sufficient to only use semantic roles which can only be filled by noun phrases, as "serve" illustrated in the previous section; prepositions and verbal complements are also likely to be useful to selectional preference based WSD systems.

The results of this study also merit several further research topics. First, the focus of this research was on the disambiguation of verbs. However, the results of the disambiguation also contains the sense information of the nouns which are the arguments to the disambiguated verbs. Therefore, the next step of the current research is to assess how well the system developed in this work would perform on the nouns.

A further extension to the current WSD system

would be to incorporate extra information such as the prepositions and other open class words in the disambiguation. This extension may require a hybrid WSD system incorporating selectional preference based mechanisms and corpus based mechanisms.

Finally, as it was observed that the performance of a selectional preference based WSD system was heavily influenced by the quality of semantic role labelling; it might also be possible to use selectional preference as a crude measure of the performances of semantic role labelling systems on unlabelled data. This is because it is likely for a particular semantic role to be fi lled by nouns of similar types, therefore nouns correctly labelled for the same semantic role should exhibit a greater similarity than if incorrectly labelled.

Acknowledgements

I would like thank my supervisors, Professor Steven Bird and Dr. Adrian Pearce, for reviewing the paper. This research was sponsored by Australian Postgraduate Awards (APA) from the Australian Research Council (ARC).

References

Ray Jackendoff, 1990. *Semantic Structures*, chapter 2. The MIT Press.

Daniel Jurafsky and James H. Martin, 2000. *Speech and Language Processing*, chapter 17. Prentice Hall, 1 edition.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM Press.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654, December.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004), Boston, MA*, May 2–7.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?, Washington, April 4-5, 1997*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196.

Differentiating Types of Verb Particle Constructions

Jon Patrick and Jeremy Fletcher
Sydney Language Technology Research Group
School of Information Technologies
University of Sydney
Sydney, Australia, 2006
jonpat@it.usyd.edu.au, jfletch1@it.usyd.edu.au

Abstract

A verb particle construction (VPC) classification scheme gleaned from linguistic sources has been used to assess its usefulness for identifying issues in decomposability. Linguistic sources have also been used to inform the features suitable for use in building an automatic classifier for the scheme with a series of good performance results. The notions of how to define the task of computing phrasal verbs are discussed and new proposals are presented.

1 Introduction

Our area of research focuses on verb-particle constructions (VPCs), a sub-section of multi-word expressions (MWEs). MWEs are a generic term for the group of expressions that include idioms (e.g. *over the moon*), lexically fixed expressions (e.g. *ad hoc*), light verb constructions (e.g. *make a mistake*), institutionalised phrases (e.g. *kindle excitement*), and verb-particle constructions (e.g. *run away*). All these expressions have in common the occurrence of words adjacent to each other that would be more frequent than if they were simply random words put together. Hence the words which constitute them have some particular meaning together that they would not have apart.

VPCs consist of a simplex (single-word) verb, and a particle, whether preposition or adverb. A particular subset of interest are "phrasal" verbs which are considered to be non-decomposable structures where the meaning is in the whole and not the parts of the phrase (Dixon, 1978).

Previous research into the area of MWEs, shows a number of places in which there has been a lack of research. The area which is of interest to us is the approach of bringing reliable resources and specific encodeable linguistic knowledge for use in feature selection in supervised learning tasks of categorisation and WSD.

2 Previous Research

There has been only limited research done in the field of MWEs in computational linguistics, mostly focusing on VPCs; although much research (Abeillé, 1988, 1995; Barkema, 1994; Wehrli, 1998) has been done on such MWEs as idioms (Abeillé, 1995; Barkema, 1994; Wehrli, 1998) and light verbs (Abeillé, 1988) within the traditional linguistics field. MWEs are very idiosyncratic constructs making progress in their computation difficult. They have been called "a serious problem for many areas of language technology" (Copestake, Lambeau, Villavicencio, Bond, Baldwin, Sag, Flickinger, 2002) "unpredictable" (Baldwin & Villavicencio, 2002) and "a pain in the neck" (Sag, Baldwin, Bond, Copestake and Flickinger. 2002), and thus many applications of computational linguistics such as machine translation put multiword expressions in the "too-hard" basket. However, the use of MWEs in natural language is widespread, and thus comes about the need for what has been called "a robust, structured handling [of MWEs]" (Baldwin & Villavicencio, 2002, Calzolari, Fillmore, Grishman, Ide, Lencu, MacLeod, and Zampolli. 2002).

Much of the previous work on VPCs has revolved around tasks besides WSD, focused on the extraction of multiple word constructs from corpora (Abeillé,1988, Baldwin & Villavicencio, 2002, Maynard & Ananiadou, 1999.) which we will discuss in some detail below, or what is termed the "decomposability" or "compositionality" of VPCs.

2.1 Extraction of MWEs from Corpora

One of the important components of research in MWEs is their automatic extraction from some corpora. The best results for precision and recall on this task is clearly the work of Baldwin and Villavicencio (2002), who used a combination of part-of-speech (POS)-based extraction (using Brill's POS tagger (Brill, 1995)), chunk-based

extraction, and chunk-grammar-based extraction, to extract VPCs from the Wall Street Journal section of the Penn Treebank.

They report precision of 0.889 and recall of 0.903 ($F\beta1 = 0.896$) for this task. Although they cite some other research into a similar area, there had been very little research in this specific area of extracting VPCs automatically from corpora, although some studies without quantitative analysis had been done previously (Kaalep & Muischnek, 2002, Krenn & Evert, 2001), along with work on the extraction of other collocations other than MWEs (Abeillé, 1988, Basili, Pazienza and Velardi, 1993, Maynard and Ananiadou, 1999).

VPC Example	Verb Contributes to Meaning	Particle Contributes to Meaning
1. Peter put the picture up	Yes	Yes
2. Susan finished up her paper	Yes	No
3. The thief made away with the cash	No	Yes
4. Barbara and Simon made out	No	No

Table 1. Summativity of VPCs; whether individual elements determine the meaning of the construction.

2.2 Determining the Decomposability of MWEs

As the semantics of most MWEs are more difficult to ascertain than the semantics of simplex words (even taking into account the problem of disambiguating between different senses of a simplex word), there has been some research done into what is termed the "compositionality" or "decomposability" of VPCs. In Bannard, Baldwin & Lascarides (2003) some examples are given of VPCs which illustrate one way of describing "summativity" (See table 1). In these examples, 1) "put up", is entirely composed of its constituent parts, as at the end of the action, the painting is both "put" somewhere, and is now "up". In 2) "finished up", the paper is "finished", but nothing is "up", in 3) "made away", the thief is "away", but nothing is "made", and in 4) nothing is either "made" or "out".

Lin (1999) discusses the principle of decomposability with regards to constructs other than VPCs, (e.g. "red tape"), and conducted an experiment whereby individual words in the collocation are replaced by words with similar meanings taken from a thesaurus.

This word-substitution technique is transferred to VPCs in Bannard (2002) where he suggests that a similar approach could be used to determine the "decomposability" of VPCs. He obtains disappointing results ranging from precision of 0.516 and recall of 0.739 (for the largest class) to precision of 0.286 and recall of 0.083 (for the smallest class).

Bannard et al (2003) describe a statistical distribution modelling framework for determining whether specific VPCs are "decomposable", and hence automatically infer their meaning.

They conducted an internet-based experiment whereby non-expert native English speakers were asked whether certain VPCs, in sets of 5 sentences for each exemplar VPC, entailed the meaning of the simplex verb and/or the meaning of the preposition. They used this as their gold standard test data to evaluate their results. This contrasts with the approach taken by Lin (1999), in that their evaluation is based on a more intuitive level, although this could in fact lead to their results becoming affected by subjective judgments. They then encapsulated the problem as a classification task, to classify VPCs into classes depending on whether or not they were composed of their individual elements. They achieved results that improved on a baseline of classifying everything as the most common class, with results ranging from 0.735 for precision and 0.892 for recall ($F\beta1= 0.810$) to 0.303 for precision and 0.769 for recall ($F\beta1=0.435$) by using their distribution modelling approach.

2.3 Word Sense Disambiguation (WSD)

There has been much general research done in the field of WSD (Krovetz & Croft, 1989, Maynard & Ananiadou, 1998, Yarowsky, 1992, 1995, for example), although very little relating specifically to disambiguating MWEs. O'Hara and Wiebe, 2003 do perhaps what is the most relevant research, on the task of disambiguating prepositions as having a locative, temporal or other meaning. They report average accuracy of 0.703, although they don't provide precision and recall scores, so it is difficult to ascertain the particular shortcomings of the system, and where improvements could be made. However, we can make the conjecture that had they determined the compositionality of the VPC to which the prepositional particle belongs, their reported accuracy would be higher.

3 Method

Our research goal is to create a system that can accurately disambiguate between the different semantics of different instances of VPCs. It is interesting to note that the task of disambiguating VPCs has not been undertaken by other researchers working on MWEs possibly for a number of reasons.

Firstly, there is no readily available corpus that has VPCs tagged for semantic differences. Hence, a major part of the work described is to collect a suitable subset of a given corpus – in this case, we use the British National Corpus (BNC) – and manually tag target VPCs as having certain semantic features.

Another reason why semantic disambiguation has not been undertaken to such a fine degree for VPCs is that there is, in fact, no comprehensive electronic resource which contains different senses of a given verb-particle construction, although there are large, readily available resources for different simplex verbs (the most obvious being WordNet (G. Miller, Beckwith, Fellcaum, Gross, and K. Miller. 1990).

Hence, the main difference between the research being undertaken in this project and that which has been done previously is the exploitation of a lexical database of phrasal verbs (constructed from Meyer, 1975). This will be used to increase the accuracy of our task on VPCs, and provide us with a compendium of "valid" verb-particle constructions. Also, whilst most semantic identification tasks relate to simplex words, there has been little research on disambiguation of MWEs. Disambiguation in this sense is most applicable to VPCs, as most other idiomatic MWEs have a single, fixed (if still metaphorical) meaning. For example, the idiom "kick the bucket", once it has been identified as a MWE has only the sense of "dying". Although the phrase itself could also have a literal sense of kicking, once it has been extracted and identified as an idiom, only this idiomatic, metaphoric sense applies. It is a similar situation with other idioms.

However, with VPCs, there are many examples that have different senses. For example the phrasal verb "to check out" has the sense of leaving a hotel, and the sense of checking something to make sure it is correct. Other examples of VPCs with multiple senses are "pick up" (understand/comprehend, retrieve from the ground, hook up with someone of the opposite sex), "look out" (look out of a window, watch out for) and "set up" (put into position, furnish with money/resources, establish, etc…).

4 Motivation

Unlike the current trend in other works that focus on recognition of VPCs from thesaural expansion of context whether automatically (Lin, 1999, Bannard et al, 2003, McCarthy et al, 2003) or using WordNet (Bannard et al 2003) and the establishment of gold standards by survey (Bannard et al, 2003, McCarthy et al, 2003) we prefer a principled method using the understandings developed by linguistic studies. The identification of (virtually) all candidate VPCs have been captured in 3 Phrasal Verb dictionaries (Collins, Oxford, and Meyer) so as a resource to determine the definitions of VPCs we use the dictionary of Meyer (1975). As well as the dictionaries, a variety of linguistic studies provide an extensive analysis of the features of VPCs. Our procedure is to use the dictionaries to guide the compilation of our gold standard and the studies to guide feature selection for automated classification to both establish the parameterisation of a generative model for recognising VPCs and to identify the strengths of the various features established by linguists.

The most comprehensive analysis of this problem has been completed by Dixon, (1982) and we use his classification scheme to guide the development of our own classification scheme and much feature selection has been gleaned from his work. For the definition of verbs we use the new Shorter Oxford English Dictionary (1993) and for particle description Lindstromberg, (1998). Dixon produced a 5 class classification scheme:

A. Literal usage of all VPC components, e.g. *John walked on the grass,*
B. Like A but with missing arguments that are reasonably inferable e.g. *He ran down (the bank) to the railway line,*
C. Obvious metaphorical extensions form literal phrases e.g. *the firm went under,*
D. non-literal constructions that cannot obviously be related to the literal form e.g. *They are going to have it out;*
E. Full idioms, e.g. *turn over a new leaf.*

This scheme is more extensive than that used in any computational study we have found, most of which attempt to resolve the compositional/non-compositional dichotomy of a VPC or at best provide a graded scale. Whilst the Dixon scheme is more detailed than other schemes we are not entirely satisfied with it. In our own studies we have come to recognise that there is a dimension of diversity in VPCs not captured by it. Some VPCs have become so established that they

Class	Description	Example(s)
N – Non-decomposable VPCs (Phrasal Verbs)	Verb-preposition pairs which are semantically related, and whose meaning is somewhat or wholly idiomatic.	*"Leeds United carried off a massive victory."* *"John and Julie made out."*
D – Decomposable VPCs	Verb-preposition pairs which are semantically related, but whose meaning is literal, or where the preposition is redundant.	*"The bee carried the pollen off to another flower"*
I – Independent verb-preposition pairs	Verb-preposition pairs which have no semantic relationship.	*"The cables carry around 1,000 volts"*

Table 2. Description of NDI classification scheme.

have a metaphorical sense derived from the original components and are found in the dictionaries catalogues as such, *have it off, kick off*. However they, like fully compositional VPCs, are used in both literal and metaphorical contexts based on the literality of the phrasal arguments, leading to the perception that compositionality is a continuum. In the case of *kick off*, our own experience is that even with literal arguments it feels metaphorical, e.g. *The game kicks off at 7pm*.

We believe that individual assessment of the literality of the arguments will establish a more reliable predictor of metaphoricity and thereby compositionality. For example, *the Government is driving down the road of disaster* has one more metaphorical argument than, *the man is driving down the road of disaster*. Furthermore, the relationship between the VPC and the head of the object is literal, and the metaphorical meaning is only created by the head modifier. Due to the manual effort in calibrating a corpus with this level of detail we have not incorporated such an analysis here, but we foreshadow it as future work.

A further deficit in this scheme is that it doesn't permit the distinction between full literal usage (class A) and literal usage of the compound where the particle is non-contributing (*do you think he will end up getting married*) or may even be removed (*are the computers linked/hooked (up) to the network*).

Further complications arise from features that can exist as one value at the sense level in the way you might define it in a dictionary but the context of usage changes those values, e.g. blow off.

Thus, we use the principles behind Dixon's classifications, along with our analysis of the problem to create a new 3-class classification scheme, which reflects more closely the real-life problem of identifying phrasal verbs.

The description of our NDI classification scheme appears in Table 2.

5 Determining Gold-standards

The work of Bannard et al (2003) determined a gold standard by randomly collecting 5 examples of a given VPC and as a collection asking non-experts to classify the components as compositional/non-compositional. We prefer an alternative approach where each individual sample sentence is assessed for all classifications of interest. Our work in this study has shown a significant diversity in random sampling so that we don't believe it can lead to a consistent result. This random sampling leads to imprecision in the classification task, due to a failure to create a homogenous data set. Indeed the poor inter-rater reliability in their study is testimony to this problem. Rather we have categorised each sample sentence extracted from the BNC.

We perceive that the notion of sense for VPCs appears at three major levels without restricting granularity within those levels. The first level is the compositional sense brought together by the union of the components, the second is the intrinsic sense that is more (or not) than the sum of the parts and conventionally recorded in a phrasal dictionary, and the third is the contextual sense that varies either of the other two senses in language usage. Hence automatic WSD will only be achieved by identifying each of these types of meaning making through fixed resources like dictionaries, manual analysis of the idiosyncratic usage in real language examples, that is corpus tagging, and the machine learning methods

appropriately parameterised for all variables of a linguistically motivated model. The current work is a beginning on the larger task of WSD for VPCs.

6 Data Selection

The data selection process consisted of firstly constructing a lexical database of Meyer's phrasal verb dictionary (1975) (denoted PV-Lex-Meyer) (Only entries A-O have been completed). The text was scanned and OCRed and then converted from a Word file into an XML database using the Ferret software (Patrick, Palko, Munro, Zappavigna, 2003). All VPCs in the database were extracted and all matching examples in the BNC retrieved on the criteria the verb and the particle had no intervening verb. This yielded over 600,000 sample sentences. To cut down the examples used, we sampled those VPCs which had medium density in the corpus; those verb-particle pairs which occurred in more than 10, but less than 40 sentences. There were approximately 70 such verb-particle pairs, giving us a reduced corpus of approximately 6000 sentences.

We used this reduced corpus as the basis for our initial classification task over the 3 possible tags for particles which occur in the BNC's CLAWS tag set, AV0 (adverb), AVP (adverbial preposition), PRP(general preposition). Our results for this experiment are shown in Table 3.

Class	Precision	Recall	F-Score
AV0	0.975	0.963	0.969
AVP	0.984	0.996	0.990
PRP	0.911	0.767	0.833

Table 3. Precison, Recall and F scores for estimating POS tags of particles.

The features used in this preliminary experiment were the POS tags of the words on either side of the verb and particle, the distance between the verb and particle, the particle's value. This experiment provided the separation of the data into the Clean and Noisy data sets, where Clean is correctly classified and Noisy is incorrectly classified.

Once we had achieved our best accuracy, we had a list of approximately 400 sentences which were not correctly identified by our classifier. We then manually tagged these 400 sentences using our VPC classification schemes.

This set provided examples of VPCs used in an atypical fashion. We use this set in our classification task, as the value of linguistic analysis is not tested by an entirely compliant data set. So although this sample is not representative of general usage, if gives a better measure of the performance of the linguistic features.

To complement these 400 sentences, and give us a rich problem to solve we sampled across the 5600 correctly classified sentences, to extract another 400. Our goal was to extract a sample of sentences which were representative of the trends in preposition usage (i.e. the POS tag of the particle), while still maintaining as wide a spectrum of verb-particle pairs as possible.

Of the extracted sentences, around 80% had particles labelled as adverbial particles, 10% as general adverbs and 10% as general prepositions. Thus, to maintain this distribution, we extracted 320 sentences which contained adverbial particles, 40 with general adverbs, and 40 sentences with general prepositions.

The representation of different VPCs according to CLAWS tags are: AV0-28, AVP-64, PRP-65. We thus targeted extraction numbers of 5 sentences for each VPC with an AVP tag, 2 sentences for each with an AV0 tag, and 1 sentence with a PRP tag. Given that some verbs occurred in less than the targeted number of sentences, we actually extracted a total of 423 example sentences.

We then manually tagged these 423 examples using our VPC schemes which was reduced to 376 when unusable sentences were deleted (principally for incorrect pre-processing, and in a few cases for being unintelligible). This set of approximately 800 examples was then used for our classification tasks. Our aim was to have a representative sample of the corpus, while maximising those examples where the classification was less obvious, and the number of different VPCs. Thus we included those examples where the use of the particle was more difficult to classify.

7 Results

The compositionality experiments are based on the NDI classification scheme, decomposable (D) (~44%), non-decomposable (N) (~36%) and Independent (I) (~19%), as this recognises the processing situation of identifying phrasal verbs in real text, rather than the somewhat artificial task of just discriminating between summative and non-summative constructs. The D class is the largest class and so sets the baseline as P=0.472, R=1, F=0.641 on the Noisy data set, P=0.412, R=1, F=0.584 on the Clean data set and P=0.442, R=1, F=0.613 on the Combined. Based on our initial linguistic analysis of the problem, we looked for a number of features within the target sentence. In

the first experiment (Table 4), the features included the particle being used, the distance between the verb and the particle in the VPC, the number of words this sentence has in common with example sentences extracted from our resource for this VP pair, whether this particle was part of one of the compound particles extracted during the manual annotation of the corpus, whether this VP pair has more senses in which it is transitive or intransitive (or neither, if there are an equal number of transitive and intransitive senses) also extracted from PV-Lex-Meyer, and the Dixon sub-categorisation frame.

In experiment 2 (Table 4) we constructed a more fine-grained scale for the transitivity measure, distinguishing between those PVs which only have transitive or intransitive senses, and those which have more transitive than intransitive senses.

	Exp 1 - Noisy			**Exp 2 - Noisy**		
	P	**R**	**F**	**P**	**R**	**F**
N	0.500	0.534	0.516	0.523	0.585	0.552
D	0.579	0.565	0.572	0.612	0.554	0.581
I	0.420	0.420	0.433	0.507	0.522	0.514
	Exp 1 - Clean			**Exp 2 - Clean**		
	P	**P**	**P**	**P**	**R**	**F**
N	0.502	0.502	0.502	0.513	0.649	0.573
D	0.461	0.461	0.461	0.485	0.406	0.442
I	0.550	0.550	0.550	0.508	0.411	0.455
	Exp 1 - Combined			**Exp 2 - Combined**		
	P	**R**	**F**	**P**	**R**	**F**
N	0.459	0.609	0.523	0.502	0.605	0.549
D	0.506	0.390	0.441	0.545	0.452	0.494
I	0.550	0.452	0.561	0.556	0.556	0.556

Table 4. Performance statistics for experiments 1 and 2.

In Experiment 3a the length of the verb was used as a feature, given that, as Dixon says, "phrasal verbs are almost exclusively based on monosyllabic verbs of Germanic origin".

We considered that this length would give us a reasonable estimate of the number of syllables. This result looked encouraging, so we manually annotated each of the verbs in our resource for the number of syllables in the word.

In experiment 3b we used this number instead of the verb length. This showed an increase in two of the classes, but a decrease in the largest class. We hence included both these features in experiment 3c (see Table 5).

Some further analysis of the linguistics of these constructs led to the creation of a measure of the differences of the arguments of the verb. To do this, in experiment 4a we identified the POS tag of the head of each of the noun phrases surrounding the preposition. In experiment 4b we then generalised this distinction, to being either a named entity, or a general noun, given that this distinction occurs within the CLAWS tag set. These features were added to the features from experiment 3c. The results are shown in Table 6.

Following on from the potential phonological interpretation of Dixon's "Germanic origin" thesis, we also used the last three letters of the verb, each as an individual feature in experiment 5. However, we appreciate this is a primitive representation of the underlying linguistic model, and needs further maturity. See Table 7.

8 Conclusions

We have shown in the research a classification system built on encodeable linguistic knowledge can predict certain semantic information about a given instance of a VPC, to a level of accuracy comparable to that which has been achieved on the more coarse-grained approach of dealing with each verb-particle pair as a unit whose semantics remain the same in different contexts.

While VPCs in different contexts have vastly different semantic properties, we have also shown that it is possible to compute these semantics from features such as the syntactic structure of the VPC, and features of the arguments of the VPC.

Although the results are not presented here, there is evidence to suggest that if the distinction between these classes could be predicted reliably we could also get good results on predicting Dixon's class assignment of a given instance of a phrasal verb.

Our approach has come from grounding in the linguistic features of VPCs, to determine what the key distinctions between VPCs with different semantics are. Whilst no direct comparison can be drawn between the work presented here and the previous studies done on the "decomposability" of VPCs, our results show more stable solutions on a more complex task, which is also closer to realistic language processing.

	Experiment 3a Noisy			Experiment 3a Clean			Experiment 3a Combined		
Class	**Prec**	**Recall**	**F-Score**	**Prec**	**Recall**	**F-Score**	**Prec**	**Recall**	**F-Score**
N	0.570	0.619	0.593	0.533	0.655	0.588	0.502	0.545	0.523
D	0.627	0.619	0.623	0.537	0.465	0.498	0.533	0.520	0.527
I	0.443	0.391	0.415	0.550	0.452	0.496	0.583	0.521	0.550
	Experiment 3b Noisy			**Experiment 3b Clean**			**Experiment 3b Combined**		
	Prec	**Recall**	**F-Score**	**Prec**	**Recall**	**F-Score**	**Prec**	**Recall**	**F-Score**
N	0.536	0.627	0.578	0.548	0.689	0.611	0.522	0.624	0.568
D	0.664	0.589	0.625	0.519	0.439	0.476	0.564	0.480	0.518
I	0.544	0.536	0.540	0.525	0.425	0.470	0.565	0.549	0.557
	Experiment 3c Noisy			**Experiment 3c Clean**			**Experiment 3c Combined**		
	Prec	**Recall**	**F-Score**	**Prec**	**Recall**	**F-Score**	**Prec**	**Recall**	**F-Score**
N	0.579	0.619	0.598	0.579	0.682	0.623	0.535	0.632	0.579
D	0.667	0.655	0.661	0.576	0.523	0.549	0.579	0.511	0.543
I	0.547	0.507	0.526	0.550	0.452	0.496	0.553	0.514	0.533

Table 5. Performance statistics for experiments 3 a,b,c

	Experiment 4a - Noisy			Experiment 4b - Noisy		
	P	**R**	**F**	**P**	**R**	**F**
N	0.522	0.649	0.578	0.525	0.635	0.757
D	0.549	0.503	0.525	0.511	0.432	0.469
I	0.600	0.411	0.488	0.515	0.466	0.489
	Experiment 4a - Clean			**Experiment 4b – Clean**		
	P	**R**	**F**	**P**	**R**	**F**
N	0.586	0.576	0.581	0.578	0.627	0.602
D	0.670	0.702	0.686	0.671	0.667	0.669
I	0.540	0.493	0.515	0.533	0.464	0.496
	Experiment 4a - Combined			**Experiment 4b – Combined**		
	P	**R**	**F**	**P**	**R**	**F**
N	0.538	0.583	0.560	0.544	0.583	0.563
D	0.566	0.554	0.560	0.564	0.560	0.562
I	0.520	0.465	0.491	0.520	0.458	0.487

Table 6. Performance statistics for experiments 4 a and b

	Experiment 5 - Noisy		
	P	**R**	**F**
N	0.675	0.703	0.689
D	0.715	0.762	0.738
I	0.566	0.435	0.492
	Experiment 5 - Clean		
	P	**R**	**F**
N	0.618	0.689	0.652
D	0.610	0.535	0.570
I	0.480	0.493	0.486
	Experiment 5 - Combined		
	P	**R**	**F**
N	0.683	0.665	0.674
D	0.647	0.619	0.633
I	0.515	0.592	0.551

Table 7. Performance statistics for experiment 5

9 Future Work

The work discussed here is a precursor to the rich, and perhaps more computationally difficult task of sense-disambiguation of VPCs. While we have gone some way to computing differences in the semantics of different VPCs, there is a far greater level of sophistication required before all the semantic properties of these anomalous constructs can be computed.

There is also scope for an improvement of the results presented here, through a deeper linguistic analysis of the structure of these VPCs, in particular looking at the features of the arguments.

We have also identified other semantic properties of VPCs (such as the dichotomy of whether the verb and preposition are being used in a literal or metaphoric sense), which in the future, may also form an independent basis for computational classification tasks. Whether these tasks can be performed to any high level of accuracy is left as an open question.

10 References

Abeillé, A. 1988. Light verb constructions and extraction out of NP in a tree adjoining grammar. In *Papers of the 24th Regional Meeting of the Chicago Ling. Soc.*

Abeillé, A. 1995. The flexibility of French idioms: A representation with Lexicalised Tree Adjoining Grammar. In M. Everaert, E-J. van der Linden, A. Schenk, and R. Schreuder, editors, *Idioms: Structural and Psychological Perspectives*, chapter 1. Lawrence Erlbaum Associates.

Baldwin, T. and Villavicencio, A. 2002. Extracting the Unextractable: A case study on verb-particles. In *Proc. of the 6th Conference on Natural Language Learning (CoNLL-2002),* Taipei, Taiwan

Bannard, C., Baldwin, T. and Lascarides, A.. 2003. A Statistical Approach to the Semantics of Verb-Particles. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, Sapporo, Japan, pp. 65-72.

Bannard, C. 2002. Statistical techniques for automatically inferring the semantics of verb-particle constructions. *LinGO Working Paper No. 2002-06.*

Barkema, H. 1994. The idiomatic, syntactic and collocational characteristics of received NPs: some basic statistics. *Hermes* 13: 19-40. Aarhus School of Business.

Basili, R., Pazienza, M. and Velardi, P. 1993. Semi-automatic extraction of linguistic information for syntactic disambiguation. *Applied Artificial Intelligence*, 7:339-64.

Blaheta, D. and Johnson, M. 2001. Unsupervised learning of multi-word verbs. In *Proc. of the ACL/EACL 2001 Workshop on the Computational Extraction, Analysis and Exploitation of Collocations*, pp 54-60.

Brill, E. 1995. "Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging", *Computational Linguistics*. 21:543-65.

Calzolari, N., Fillmore, C., Grishman, R., Ide, N., Lencu, A., MacLeod, C. and Zampolli, A. 2002. Towards best practice for multiword expressions in computational lexicons. In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002), pp 1934-40*

Copestake, A., Lambeau, F., Villavicencio, A., Bond, F., Baldwin, T., Sag, I.A. and Flickinger, D. 2002. Multiword expressions: linguistic precision and reusability. In *Proc. Third conference on Language Resources and Evaluation (LREC-2002)*, pp. 1941—1947. Las Palmas, Canary Islands.

Dixon, R. M. W. 1982. The Grammar of English Phrasal Verbs. *Australian Journal of Linguistics*, 2:149-247.

Kaalep, K. and Muischnek, K. 2002. Using the text corpus to create a comprehensive list of phrasal verbs. In *Proc. of the 3rd International conference on Language Resources and Evaluation (LREC 2002),* pp 101-5.

Krenn, B. and Evert, S. 2001. Can we do better than frequency? A case study on extracting PP-verb collocations. In *Proceedings of the ACL Workshop on Collocations*, Toulouse, France, pp 39-46.

Krovetz, R. and Croft, W.B. 1989. Word sense disambiguation using machine-readable dictionaries. In *Proc. Annual ACM Conference on Research and Development in Information Retrieval 1989.*

Lin, D. 1999. Automatic Identification of Non-compositional Phrases. In *Proc. 37th Annual Meeting of the ACL,* pp 317-24.

Lindstromberg, S. 1998. English Prepositions Explained. Amsterdam: John Benjamins.

Maynard, D. and Ananiadou, S. 1999. Identifying contextual information for multi-word term extraction. In *5th International Congress on Terminology and Knowledge Engineering (TKE 99)*, pp 212-21.

Maynard, D. and Ananiadou, S. 1998. Acquiring contextual information for term disambiguation. In *Proc. of 1st Workshop Computational Terminology, Computerm '98,* Montreal, Canada.

Meyer, G. A. 1975. The Two-Word Verb, A Dictionary of the Verb-Prepositional Phrases in American English. The Hague: *Mouton.*

Patrick, J., Palko, D., Munro, R., Zappavigna, M. 2003. Inferring semantic structure from format, Computing Arts: Digital Resources in the Humanities, (ed) C. Cole & H. Craig, Sydney: The Uni of Sydney, pp150-168.

McCarthy, D., Keller, B., Carroll, J. 2003, Detecting a Continuum of Compositionality in Phrasal Verbs. In *Proceedings of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, Sapporo, Japan, pp. 65-72.

Miller, G.A., Beckwith, R., Fellcaum, C., Gross, D. and Miller, K.J. 1990. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235-44.

O'Hara, T and Wiebe J. 2003. Preposition Semantic Classification via PENN TREEBANK and FRAMENET. In *Proc. of Computational Natural Language Learning-2003 (CoNLL-03).*

Sag, I.A., Baldwin, T., Bond, F., Copestake, A. and Flickinger, D. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pp 1-15, Mexico City, Mexico.

Smadja, F. 1993. Retrieving collocations from text: Xtract. In *Computational Linguistics*, 19(1):143-78.

Wehrli, E. 1998. Translating idioms. In *Proc. of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics: COLING/ACL-98*, pp 1388-92, Montreal, Canada.

Yarowsky, D. 1992. Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In *Proceedings, COLING-92.* pp 454-460.

Yarowsky, D. 1995. Unsupervised Word Sense Disambiguation Rivalling Supervised Methods. *Proc 33rd Ann. Meet.of the ACL.*