Australasian Language Technology Workshop 2007

Proceedings of the Workshop

Workshop Chairs:
Nathalie Colineau
Mark Dras

10–11 December 2007
Melbourne Zoo
Melbourne, Australia

Proceedings of the Australasian Language Technology Workshop 2007
URL: `http://www.alta.asn.au/events/alta2007/`

Sponsors:

NICTA

CSIRO

APPEN

inferencecommunications

THE UNIVERSITY OF MELBOURNE

ISSN 1834-7037 (for the online proceedings)

Preface

This volume contains the papers accepted for presentation at this year's Australasian Language Technology Workshop (ALTA2007), held at the Melbourne Zoo, Melbourne, Australia, on December 10–11, 2007. This is the fifth annual installment of the workshop in its most recent incarnation, and the continuation of an annual workshop series that has existed under various guises since the early 90s.

The goals of the workshop are:

- to bring together the growing Language Technology (LT) community in Australia and New Zealand and encourage interactions;
- to encourage interactions between this community and the international LT community;
- to foster interaction between academic and industrial researchers;
- to encourage dissemination of research results;
- to provide a forum for the discussion of new and ongoing research and projects;
- to provide an opportunity for the broader artificial intelligence community to become aware of local LT research; and, finally,
- to increase visibility of LT research in Australia, New Zealand and overseas.

This year has seen quite some activity in language technology in Australia, with PACLING 2007, the 10th Conference of the Pacific Association for Computational Linguistics, being hosted in September also in Melbourne. In addition, this year ALTA is co-located with the Australian Document Computing Symposium (ADCS), including a joint session of papers and talks of interest to both communities.

This year's Australasian Language Technology Workshop includes regular talks as well as poster presentations and student posters. Of the 23 papers submitted, 16 papers were selected by the program committee for publication and appear in these proceedings. Of these, 10 are oral presentations and 6 are poster presentations. Additionally, we have included 3 student posters to encourage feedback on early results. Each full-length submission was independently peer reviewed by at least two members of the international program committee, in accordance with the DEST requirements for E1 conference publications.

We would like to thank all the authors who submitted papers; the members of the program committee for the time and effort they contributed in reviewing the papers; and our invited speakers, Sophia Ananiadou, Nick Thieberger and Justin Zobel. Our thanks also go to local organisers Nicola Stokes and Lawrence Cavedon, to members of the ALTA executive for their support in organising the workshop, and to our sponsors (NICTA, CSIRO, Inference Communications, Appen, and the University of Melbourne), who enabled us in particular to support student participation in the event.

Nathalie Colineau and Mark Dras
Programme co-chairs

Table of Contents

Workshop Programme

Day 1 — 10 December, 2007

09:15 – 09:30 Welcome and Opening

09:30 – 10:00 *An Empirical Investigation into Grammatically Constrained Contexts in Predicting Distributional Similarity*
Dongqiang Yang and David Powers

10:00 – 10:30 *TAT: An Author Profiling Tool with Application to Arabic Emails*
Dominique Estival, Tanja Gaustad van Zaanen, Son Bao Pham, Will Radford and Ben Hutchinson

10:30 – 11:00 Coffee Break

11:00 – 11:30 *Extending Sense Collocations in Interpreting Noun Compounds*
Su Nam Kim, Meladel Mistica and Timothy Baldwin

11:30 – 12:00 *Dictionary Alignment for Context-sensitive Word Glossing*
Willy Yap and Timothy Baldwin

12:00 – 12:15 Lightning preview of posters

12:15 – 13:45 Lunch

13:45 – 15:15 Poster Session
Exploring Approaches to Discriminating among Near-Synonyms
Mary Gardiner and Mark Dras
Practical Queries of a Massive n-gram Database
Tobias Hawker, Mary Gardiner and Andrew Bennetts
Named Entity Recognition in Question Answering of Speech Data
Diego Mollá, Menno van Zaanen and Steve Cassidy
Experiments in Mutual Exclusion Bootstrapping
Tara Murphy and James Curran
Two-Step Comprehensive Open Domain Text Annotation with Frame Semantics
Bahadorreza Ofoghi, John Yearwood and Liping Ma
Question Prediction Language Model
Luiz Augusto Pizzato and Diego Mollá
Exploring Extensions to Machine-Learning Based Gene Normalisation
Benjamin Goudey, Nicola Stokes and David Martinez
Distributional Similarity of Multi-Word Expressions
Laura Ingram and James Curran
Extending CCG*bank with Quotes and Multi-modal* CCG
Daniel Tse and James Curran

15:15 – 15:30	Walk through zoo to joint session with ADCS
15:30 – 16:00	Coffee Break
16:00 – 16:45	JOINT SESSION WITH ADCS *Exploring Abbreviation Expansion for Genomic Information Retrieval* Nicola Stokes, Yi Li, Lawrence Cavedon and Justin Zobel *A Bottom-Up Term Extraction Approach for Web-Based Translation in Chinese-English IR Systems* Chengye Lu, Yue Xu and Shlomo Geva (ADCS authors) *Automatic Thread Classification for Linux User Forum Information Access* Timothy Baldwin, David Martinez and Richard B. Penman (ADCS authors)
16:45 – 17:45	*Measures of Measurements: Robust Evaluation of Search Systems* Invited Speaker – Justin Zobel
18:00 – 19:00	Canapes and Drinks

DAY 2 — 11 DECEMBER, 2007

09:15 – 10:15	*Text Mining Techniques for Building a Biolexicon* Invited Speaker – Sophia Ananiadou
10:15 – 10:45	*Measuring Correlation Between Linguist's Judgments and Latent Dirichlet Allocation Topics* Ari Chanen and Jon Patrick
10:45 – 11:15	Coffee Break
11:15 – 11:45	*Charting Democracy Across Parsers* Scott Nowson and Robert Dale
11:45 – 12:15	*Parsing Internal Noun Phrase Structure with Collins' Models* David Vadas and James R. Curran
12:15 – 13:30	Lunch
13:30 – 14:15	ALTA Annual General Meeting
14:15 – 14:45	*Entailment due to Syntactically Encoded Semantic Relationships* Elena Akhmatova and Mark Dras
14:45 – 15:15	*Statistical Machine Translation of Australian Aboriginal Languages: Morphological Analysis with Languages of Differing Morphological Richness* Simon Zwarts and Mark Dras
15:15 – 15:45	Coffee Break
15:45 – 16:45	*Does Language Technology Offer Anything to Small Languages?* Invited Speaker – Nick Thieberger
16:45 – 17:00	Awards and Closing Remarks

Text Mining Techniques for Building a Biolexicon

Sophia Ananiadou
School of Computer Science
The University of Manchester
sophia.ananiadou@manchester.ac.uk

My talk will focus on building a biolexicon by leveraging existing bio-resources, combining them within a common, standardized lexical, terminological, conceptual representation framework and employing advanced NL technologies to discover new terms, concepts, relations and linguistic lexical information from text. In particular I will discuss term normalisation techniques, named entity recognition and a smart dictionary look up. This research forms part of the National Centre for Text Mining (www.nactem.ac.uk) and the project BOOTStrep.

Does Language Technology Offer Anything to Small Languages?

Nick Thieberger
PARADISEC, University of Melbourne/
University of Hawai'i at Manoa
thien@unimelb.edu.au

The effort currently going into recording the smaller and perhaps more endangered languages of the world may result in computationally tractable documents in those languages, but to date there has not been a tradition of corpus creation for these languages. In this talk I will outline the language situation of Australia's neighbouring region and discuss methods currently used in language documentation, observing that it is quite difficult to get linguists to create reusable records of the languages they record, let alone expecting them to create marked-up corpora. I will highlight the importance of creating shared infrastructure to support our work, including the development of Pacific and Regional Archive for Digital Sources in Endangered Cultures (PARADISEC), a facility for curation of linguistic data.

Proceedings of the Australasian Language Technology Workshop 2007, pages 3-3

Measures of Measurements: Robust Evaluation of Search Systems

Justin Zobel
National ICT Australia
justin.zobel@nicta.com.au

A good search system is one that helps a user to find useful documents. When building a new system, we hope, or hypothesise, that it will be more effective than existing alternatives. We apply a measure, which is often a drastic simplification, to establish whether the system is effective. Thus the ability of the system to help users and the measurement of this ability are only weakly connected, by assumptions that the researcher may not even be aware of. But how robust are these assumptions? If they are poor, is the research invalid? Such concerns apply not just to search, but to many other data-processing tasks. In this talk I introduce some of the recent developments in evaluation of search systems, and use these developments to examine some of the assumptions that underlie much of the research in this field.

Proceedings of the Australasian Language Technology Workshop 2007, pages 4-12

Entailment due to Syntactically Encoded Semantic Relationships

Elena Akhmatova
Centre for Language Technology
Macquarie University
Sydney, Australia
elena@ics.mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, Australia
madras@ics.mq.edu.au

Abstract

The majority of the state-of-the-art approaches to recognizing textual entailment focus on defining a generic approach to RTE. A generic approach never works well for every single entailment pair: there are entailment pairs that are recognized poorly by all the generic systems. Automatic identification of such entailment pairs and applying to them an RTE algorithm that is specific to them could thus increase an overall performance of an entailment engine (that in this case will combine a generic RTE algorithm with a number of RTE algorithms for the problematic entailment pairs). We identify one subtype of entailment pairs and develop a two-part probabilistic model for their classification into true and false entailments and evaluate it relative both to a baseline and to the RTE systems. We show that the model performs better than the baseline and the average of the systems from the RTE2 on both the balanced and unbalanced datasets we have created for evaluation.

1 Introduction

Recognizing Textual Entailment (RTE) is a task where, given two text snippets, the goal is to determine whether the meaning of one text snippet can be inferred from the meaning of the other (Dagan et al., 2005). The first of the text snippets in such a pair is referred to as *the text* and the other one as *the hypothesis*. The pair of text and hypothesis is called a *text-hypothesis pair* or *entailment pair*, with the two names considered to be synonymous. The text is usually much longer than the hypothesis. It can be represented by one or more coherent sentences, while the hypothesis is usually one short sentence. It is the meaning of the hypothesis that might or might not be entailed from the text. Thus, given a text-hypothesis pair, we recognize the relation between the meanings of the text and the hypothesis in the pair as *a true entailment* if the meaning of the hypothesis is entailed from the meaning of the text. Otherwise, we recognize the relation between the meanings of the texts as *a false entailment*.

There are several datasets for RTE. They contain text-hypothesis pairs marked *yes* if there is a relation of true entailment in a pair and *no* otherwise. These datasets are manually created annually for the RTE Challenges[1] and are freely available.

Most state-of-the-art approaches to RTE seek a generic approach to the task and do not differentiate between text-hypothesis pairs. However, a possible alternative is to consider subclasses of entailment pairs and build models to handle these specialties. An instance of this idea is proposed in Vanderwende and Dolan (2005), where the complete set of entailment pairs is divided in two: those whose categorization could be accurately predicted based solely on syntactic cues and those where it is not the case. Their subsequent work (Vanderwende et al., 2006) presents an RTE system based on this work.

The broader context of our work is to investigate different ways of subclassifying entailment pairs. In this framework, a generic system would have additional special components that take care of the special subclasses of entailment pairs. Such a component is involved when a pair of its subclass is recognized. Note that we do not envisage classifying all the entailment pairs to give a partitioning of the space, a probably infeasible task. We suggest dividing into classes the entailment pairs that are problematic for all the state-of-the-art generic systems and develop separate RTE algorithms for these par-

[1] http://www.pascal-network.org/Challenges/RTE/

ticular classes. The broad question that we aim to answer is whether this will improve the overall performance of the RTE engine.

In this paper we are looking at one subtype of entailment pairs where a semantic relation expressed in the hypothesis is implicitly represented by a syntactic construction in the text. There are several reasons to work with this type of entailment pairs. First, it proves possible to recognize them well automatically and distinguish them from other entailment pairs using machine learning. Second, narrowing down the entailment pairs to this subset allows us to draw an analogy with, and develop an algorithm related to, the work by Lapata (2001) that finds the implicit relation between attributes to a head noun in the noun group. That together with a conditional probability model in a parallel with SMT will be taken as the basis of an algorithm for classification of entailment pairs of the chosen type. We evaluate the approach on the RTE2 annotated dataset.

The layout of the paper follows the general flow of the research. Section 2 defines the chosen type of entailment pairs. Section 3 describes an automatic classifier which distinguishes the desired type of the entailment pairs. Section 4 describes an algorithm for recognizing true and false entailments for the entailments of the chosen type, and gives some experimental results comparing our algorithm against a number of baselines. Section 5 presents the evaluation results and section 6 concludes the work.

2 Entailment types

We looked through the RTE2 test set and partitioned the set into several groups of entailments. Though the entailment pairs are different, for every word in the hypothesis there is often a word in the text from which it is entailed. It is not always so and we focus on the entailment pairs where this is not the case.

2.1 Syntactically encoded semantics

The entailment relationship we are focusing on is named *an Entailment due to Syntactically Encoded Semantic Relationships (ESESR)*, as a specific syntactic construction in the text encodes a semantic relationship between its elements that is explicitly shown in the hypothesis.

Being more precise, the text-hypothesis pairs of interest have the following characteristics:

1. The hypothesis is a simple sentence. That is a sentence that consists of a subject, a predicate, and an object, and has no subordinate clauses.

2. Both subject and object of the hypothesis (or their morphological variants) are found in the text.

3. The predicate of the hypothesis has no match with anything in the text that is linked to the matches of the subject and the object of the hypothesis.

4. The matches of the subject and the object in the hypothesis can be linked to each other in the text by any syntactic relationship except depending from the same verb or a derivative of it.

Thus, the predicate of the hypothesis is the semantic relationship between its subject and object that is not explicitly defined in the text but is implicitly presented in the syntactic relationship between the matches of the subject and object of the hypothesis in the text.

The most frequent syntactic relationships between the matches of the subject and the object of the hypothesis in the text in the RTE2 dataset are apposition,[2] a noun group and its prepositional attachment, and attributive relation within a noun group.

Consider the examples of the entailments of the described type:

(1) *Text*: From Les Combes, in the Italian Alps, yesterday, where the Pope is on vacation, the Vatican's Press Office Director, Joaquin Navarro Valls, responded with a written statement to the accusations made by the Israeli government against Benedict XVI.

Hypothesis: Les Combes is located in the Italian Alps.

The location *Les Combes* is in the relation of apposition to *the Italian Alps*. This syntactic relation implicitly encodes the semantic relation represented by the words *is located in* between the noun groups.

[2]We use the definition of Quirk et al. (1985) here.

(2) *Text*: Lt. Jim Bowell of the Butler Township Fire Department said the 4:45 a.m. accident set fire to about 100 yards of woods.

Hypothesis: Jim Bowell is engaged by the Butler Township Fire Department.

Lt. Jim Bowell is connected syntactically to *the Butler Township Fire Department* via a preposition. That implicitly encodes a relation between the person and organization, *to be engaged by*.

(3) *Text*: Japan's Kyodo news agency said the US could be ready to set up a liaison office—the lowest level of diplomatic representation—in Pyongyang if it abandons its nuclear program.

Hypothesis: Kyodo news agency is based in Japan.

The attributive relationship between *Kyodo news agency* and *Japan* suggests but does not state explicitly the relationship *is based in* between them. The Kyodo news agency *is based in* Japan is entailed from the attributive relationships between the nouns.

2.2 Recognition of the entailment types by RTE2 Challenge participants

The fact that most entailment engines rely on high word overlap, longest common substring and other features[3] implies an assumption that there must be a word in the text for every word in the hypothesis. That in its turn suggests the ESESR entailment pairs may not be recognized well.

The RTE2 results confirm that. The mean recognition of the entailments of this subtype is 61.9% among all the 41 system submissions. This places the type we have defined around the middle: difficult enough to be a challenge, but not so difficult as to be infeasible. The agreement on the recognition of the true entailments is around 86%, and the false entailments are recognized correctly with an accuracy of less than 25%. The features mentioned above tend to guess the true entailment as the matches of the subject and the object of the hypothesis in the text give a good score for word overlap, longest common substring and dependency tree matches. The false entailment is not found as the predicate of the hypothesis, important in this case, is not taken into account by these generic features.

[3]See, for example, system descriptions in the proceedings of the RTE1 and RTE2 Challenges at http://www.cs.biu.ac.il/~glikmao/rte05 and http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/ respectively.

3 Classification

In this section we want to verify that entailment pairs of the ESESR subtype can be recognized. To do this we construct a machine learner. It marks entailment pairs as true if they are of the ESESR type and false otherwise.

To extract the features we build first the word-to-word alignment between the words of the text and hypothesis, based on WordNet.[4] The features for the machine learner are based on the syntactic and semantic relationships between the aligned parts of the text and the hypothesis. We build two sets of features: ones that tell that the entailment is of a given type, and ones that tell that the entailment is not of the given type.

The syntactic features:

for: The syntactic features that are in favour of the ESESR type are the existence of a particular syntactic relationship between the matches of the subject and the object of the hypothesis in text, namely apposition, being within the same noun group, representing a noun group and its prepositional attachment or the combination of the above.

against: The syntactic features that indicate that the entailment pair is not of the ESESR type show that the aligned parts of the hypothesis in the text are connected in the text by a predicate or represent the predicate themselves.

The semantic features: For the semantic description of the text and the hypothesis we are inter-

[4]Two words are aligned if there is a path bewteen them in WordNet of length $\leq$ 3. The Cartesian product of the set of the words of the text and the set of the words of the hypothesis yields a set of the candidate word pairs. We used WordNet 2.0 and the C++ API provided by the WordNet developers to look for the paths between the words. We consider the path *travel#v#1* – *walk#v#1* as a path of length 2, where *walk#v#1* is a hyponym of *travel#v#1*, *teakettle#n#1* – *kettle#n#1* – *pot#n#1* is a path of length 3. There can be any WordNet relationships between the nodes in the path except antonyms.

ested in the number of the aligned words, predicates and named entities.

We have 16 features all together. For a more detailed description of the features please refer to Akhmatova and Dras (2007).

The RTE2 test set consists of 800 entailment pairs. Only approximately one tenth of those pairs are ESESR entailments. To build the classifier we have duplicated all the ESESR entailment pairs several times to make the distribution of the entailment pairs equal. (We indeed took care later for the cross-validation that the examples on which we test are not in the training set in this case.) The reason for this is that we are interested in true positives to apply to them an algorithm in section 4. Having only a small proportion of the set being of the ESESR type leads the machine learner to underweight these in the attempt to maximize the overall accuracy and gives a low TP, true positive, rate, which is the one we are interested in. We ran the J48 classifier on the dataset with the one-leaf-out cross validation test mode using the WEKA ML API (Witten and Frank, 1999). The overall accuracy is 75% (see table 1).

4 Model

The problem of assigning a value of true or false can be thought of probabilistically, evaluating the conditional probability of the hypothesis h given the text t, $P(h|t)$. We can rewrite this using Bayes Rule as

$$P(h|t) = \frac{P(t|h) \times P(h)}{P(t)}$$

An analogy with Statistical MT can be drawn here. As in SMT[5] we divide the calculation of $P(h|t)$ into two parts, each of which we are able to estimate. One difference is that in SMT we find the argmax of this function to find the best target sentence for the source sentence. This allows us to ignore the denominator. In entailment we must find a threshold that will divide the true entailment pairs from false, so $P(t)$ will constitute at least a scaling factor. It is true that $P(t)$ may be different for each text, so whether the common threshold can be found is not obvious. However the related work of Glickman et al. (2005) on defining probabilistic textual entailment shows that such a threshold is possible. In this paper we regard it as an empirical question; we discuss it further in Section 4.3.

In SMT $P(t|h)$ is generally referred to as the translation probability and $P(h)$ as the language model; but $P(h)$ is more generally speaking just a prior distribution, the knowledge available in the absence of the more detailed information. In the context of this work, when we know nothing about the extra semantic or syntactic relationships between the words of the text and the hypothesis, the estimation of the probability of the hypothesis sentence is a prior probability of the entailment relation in a pair.

For example, if the text sentence contains *Samuel L. Husk, executive director of the Council of Great City Schools, …* (see example (4)) then it is more likely in the absence of other knowledge to entail that *Samuel L. Husk works for the Council of Great City Schools*, than that *Samuel L. Husk threw a party in the Council of Great City Schools*. Thus, our expectation is that the former sentence is a more probable sentence in the language than the latter, and that it can be supported by corpus statistics.

4.1 Model: part I

To calculate a prior probability of the entailment relation, $P(h)$, we adapt the work of Lapata (2001). She was interested in disambiguation of a relationship between an adjective and a noun inside a noun group. Using corpus statistics it was estimated that the adjective *fast* and a noun *planes* in a noun group *fast planes* are much more probable to be in a relationship represented by the word *to fly* (*the planes that fly fast*) than in relationships *to break* or *to land* (*the planes that break fast* or *the planes that land fast*). Similar to that, we want to estimate that, if it is not stated otherwise, the most probable relationship between a person *Samuel L. Husk* and a company *the Council of Great City Schools* is *to work for*.

Thus, similar to Lapata (2001), we calculate the probability of the hypothesis sentence as a probability of a triple consisting of a subject of the hypothesis sentence, NE_1, its predicate, R, and a direct or indirect object, NE_2, that is the probability $P(NE_1, R, NE_2)$. We had to take named entities instead of the actual subject and object, as firstly, subject and object very often belong to the set of

[5]See, for example, "A Statistical MT Tutorial Workbook," unpublished, August 1999 at http://www.isi.edu/~knight/.

TP Rate	FP Rate	Precision	Recall	Class
0.87	0.39	0.69	0.87	FALSE
0.61	0.13	0.84	0.61	TRUE

Table 1: The result of the J48 classifier

standard named entities, such as Person, Location, Organization, JobTitle; and secondly, actual subjects and objects will be rare in the corpus, therefore not allowing us to gather reliable statistics about them.

$$\begin{aligned} P(h) &:= P(NE_1, R, NE_2) \\ &= P(NE_1|R, NE_2) \times P(R, NE_2) \\ &= P(NE_1|R, NE_2) \times P(R) \times P(NE_2|R). \end{aligned}$$

We will make an approximation assuming that NE_1 is independent of NE_2

$$P(NE_1|R, NE_2) \approx P(NE_1|R), \text{thus}$$
$$P(h) = P(NE_1|R) \times P(R) \times P(NE_2|R).$$

We estimate the individual probabilities by corpus frequency counts ($C(x)$ represents the counts of x)

$$\begin{aligned} P(h) &= \frac{C(NE_1, R)}{C(R)} \times \frac{C(R)}{\sum_{i=1}^{n}(C(R_i))} \times \frac{C(NE_2, R)}{C(R)} \\ &= \frac{C(NE_1, R) \times C(NE_2, R)}{C(R) \times \sum_{i=1}^{n}(C(R_i))}. \end{aligned}$$

These probabilities have been calculated pairwise for Location, Person, JobTitle and Organization. The corpus was the first 500,000 sentence of the Wikipedia XML corpus (Denoyer and Gallinari, 2006) parsed using the Minipar parser (Lin, 1998) and Annie plug-ing of the GATE development environment (Cunningham et al., 1996). Table 2 shows a selection of the relations found in the RTE2 dataset. So, for example, *Person work(s) in Location* (at rank 93, with a $-\log_2(P(h))$ of 10.25) is much more frequent than *Person represent(s) Location* (at rank 775, with a $-\log_2(P(h))$ of 13.60).

Person-Location			Person-Organization		
live	182	11.15	attended	4	6.40
resides	711	13.40	works	609	13.70
represents	775	13.60	related	681	14.03
comes	331	12.00	engaged	714	14.18
worked	93	10.25	is player	493	13.31
Organization-Location			writes	242	11.98
operates	36	10.09	command	1206	16.67
based	130	11.41	is representative	206	11.78
located	7	8.13	is head	258	12.13
attended	543	13.93	heads	115	10.93
published	56	10.51	is member	11	8.20
Organization-Organization			occupied	776	14.40
owns	43	10.41	employed	884	14.90
Location-Location			JobTitle-Organization		
located	5	6.15	attended	31	10.53
situated	33	8.87	works	470	15.21
lies	32	8.87	is player	429	14.95
Location-Organization			writes	762	17.40
is subordinate	162	11.86	is head	9	9.12
			heads	73	11.56
			employed	681	16.62

Table 2: Some relations extracted from the first 500,000 sentences of the Wikipedia XML corpus. The three columns give the relation, its rank in a sorted list, and the value $-\log_2(P(h))$ respectively.

4.2 Model: part II

Whereas $P(h)$ is a prior probability looking only at the relationship between subject and object in the hypothesis, $P(t|h)$ looks at the aspects of the text that might suggest the entailment relationship. Consider example 4 below.

(4) *Text*: “Relative size and the power of the purse are certainly key factors,” says Samuel L. Husk, executive director of the Council of Great City Schools.

Hypothesis: Samuel L. Husk *works for* the Council of Great City Schools.

There is a direct syntactic connection between *Samuel L. Husk* and *executive director of the Council of Great City Schools*. By contrast, consider example 5.

(5) *Text*: Both aftershocks had their epicentre around the Nicobar island group in the south of archipelago that lies close to *Indonesia*, *India’s Meteorological Department* said.

Hypothesis: *India’s Meteorological Department* operates from *Indonesia*.

There is no syntactic relationship between *India’s Meteorological Department* and *Indonesia*, suggesting the hypothesis is not a valid entailment.

Our approach to estimating $P(t|h)$, then, is to decide whether particular relatioships in the text hold. To do this we built a classifier with various classes of features.

Features 1 and 2 syntactic structure of the text sentence: presence or absence of the syntactic connection between the aligned elements; type of the syntactic relationship, if present.

Features 3 – 6 alignment: number of non-aligned words between the aligned noun groups, number of the non-aligned head elements of the aligned noun groups.

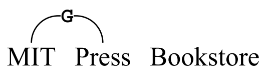
Features 7 and 8 syntactic structure of the aligned noun groups.

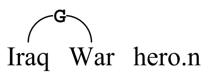
Feature 9 paraphrases.

We have already briefly mentioned above the importance of the syntactic dependencies between the matches of the subject and object of the hypothesis in the text.

The alignment features capture the fact that if there are too many missed words in the aligned noun groups then the hypothesis might have aquired different meaning from the one expressed in the text. Non-aligned head elements of the noun groups greatly increase the possibility of the meaning altering.

In determining the existence of syntactic relationships within the text, we use the Link Grammar Parser (Sleator and Temperley, 1991). To give an example for the features 7 and 8, the link G, for example, connects proper noun words together. For example, *MIT* and *Press* in the *MIT Press Bookstore*, see example (6), as well as *Iraq* and *War* (see example (7)), will be connected by the link G. We would say that the hypothesis is closer to the text if from the noun groups *MIT Press Bookstore* and *the Iraq War hero* the whole parts *MIT Press* and *Iraq War* were present in the hypothesis, rather than just *MIT* or *Iraq*, for example. If it is not the case and one can see only the first parts of the *MIT Press* and *Iraq War* components of the text sentence, then we say that the G link is ‘broken’. A broken G link reduces the probability of the true entailment between the text and the hypothesis.





In the examples (6) and (7) the G relation in the noun groups was broken. *The MIT Press* was substituted with *The MIT*, *Iraq War* with *Iraq*. That led to the hypotheses that the meaning of the text is not entailed correctly.

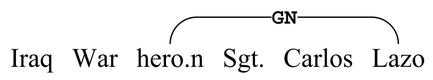
(6) *Text*: The MIT Press Bookstore stocks most of the books and journals published by The MIT Press as well as the best of other publishers books in related fields.

Hypothesis: The MIT is a book store.

(7) *Text*: The State Department is making the unusual offer of giving expedited visas to the Cuban sons of Iraq War hero Sgt. Carlos Lazo, so they can visit him in the United States, people familiar with the case said Friday.

Hypothesis: Sgt. Carlos Lazo worked in Iraq.

The link GN connects a proper noun to a preceding common noun which introduces it.



MX connects modifying phrases with commas to preceding nouns. Thus, *Sgt. Carlos Lazo* is connected to the *Iraq War hero* in *Iraq War hero Sgt. Carlos Lazo* by the GN link. It is the same for the *Maricopa County Superior Court Judge* and *Lindsay Ellis* in the *Maricopa County Superior Court Judge Lindsay Ellis*, see example (8). In case the *Iraq War hero* and *Sgt. Carlos Lazo* were in the sentence in the relation of apposition, for example, *Sgt. Carlos Lazo, an Iraq War hero*, they would be connected by an MX link. That makes GN and MX links to be equivalent for us here. The parts connected by the links GN and MX are substitutable, *Sgt. Carlos Lazo* is a *hero*, *Lindsay Ellis* is a *judge*. Thus, if the head nouns in *Maricopa County Superior Court Judge* and *Iraq War hero* are not aligned the hypothesis still might be true.

(8) *Text*: Maricopa County Superior Court Judge Lindsay Ellis also ordered Miss Bickel to pay $5,000 in restitution to Miss Tomazin's family and to perform 40 hours per week of community service indefinitely.

Hypothesis: Lindsay Ellis occupies a post at the Superior Court.

Feature 9 is the number of paraphrased phrases.

(9) *Text*:Mahmoud al-Zahar , a Hamas leader in Gaza, said so explicitly, dismissing Mr. Abba's arguments: History has proven that the rockets have been in the Palestinian interest.

Hypothesis:Mahmoud al-Zahar is a member of Hamas.

Leader and *member* are not synonyms, but they will be found to be paraphrases of each other by the algorithm proposed in Bannard and Callison-Burch (2005). To acquire the paraphrases we used the PhraseBuilder[6] on English and Dutch corpuses of Europarl.

[6]we have used the PhraseBuilder by Simon Zwarts http://www.ics.mq.edu.au/~szwarts/Downloads.php

4.2.1 Deriving a Probability

We have selected the k-nearest neighbours method, which has quite a transparent method of calculating the probability for an instance to belong to a particular class (Mitchell, 1997). We used WEKA API k-nearest neighbours method implementation for our work.

We then derive a probability from our classifier. In classification, classified instances will fall at varying distances from the boundaries which define the class spaces. This can correspond, for example, to the certainty of classification, and various classification methods have a derived probability of classification. In our case, with classes being true entailment and false entailment, we can use this as an estimate of $P(t|h)$.

The accuracy of the machine learner built on these features with $k = 5$ is not high, 54%, on the one-leaf-out approach. We are interested here though in the probabilities of belonging to a particular class rather than in the classification. $P(true|instance) = 0.49$ is the same for us here as $P(true|instance) = 0.51$. That means that the algorithm is not actually sure to which class the instance belongs. That the $P(true|instance)$ is greater than, say, 80% would be an important clue in the class prediction.

4.3 Combining part I and part II

For calculating our $P(h|t)$, as defind at the start of the Section 4, we have estimates of $P(h)$ and $P(t|h)$. We will assume that $P(t)$ is a constant for all entailment pairs and acts as a normalizing factor. (This may not be true, but we treat it here as an empirical question.)

We want then to find a threshold H for $P(h|t)$, such that where $P(h|t) \geq H$ the entailment pair is true, and false otherwise. The threshold H then incorporates the normalizing factor $P(t)$.

We have created a balanced corpus of the *true* and *false* examples of the ESESR entailment pairs from the RTE2 dataset. Then, as the one-leaf-out approach suggests, for every instance (that is, for every entailment pair) we created a separate dataset not containing it to build the k-nearest neighbours classifier. The probability of the instance being a true entailment on this classifier is the outcome of the

baseline	unbalanced dataset performance	balanced dataset performance
41 submissions mean	61.9%	50%
best performing on ESESR system	86%	73%
secondbest system	74%	55%
default “yes”	78%	50%

Table 3: Baselines and their performance on the balanced and unbalanced datasets

classification process, see the section 4.2.1. Then this probability is combined with the probability of the hypothesis $P(h)$, described in the section 4.1. This process is repeated for every entailment pair. Thus, as a result, every entailment pair is associated with a value of the probability $P(h|t)$.

One possibility to find a good value of such an H is to carry out a search over possible values on a development set. As an alternative we used a machine learner again, a decision tree, with the single feature being the combined probability. The top node of the decision tree is the best split of data. Due to the fact that the probabilities $P(h)$ are quite small numbers, we used as a feature for the decision tree also the product of the logarithms base two of the probabilities. Even though this is not strictly derivable from our model, it is still a ranking and we get a good threshold. The threshold $H = 3.41$ fits the training set best of all.

5 Evaluation

We compare the results of the approach on two datasets, *an unbalanced dataset* consisting of all the ESESR entailments from the RTE2 corpus; and *a balanced dataset*, the set of 50000 random balanced subsets of the unbalanced dataset containing all the false entailments and the same number of randomly chosen true entailments (refer to section 2.2).

We take four baselines as a comparison for our approach:

1. the mean of the accuracy of all the 41 submissions to the RTE2 Challenge;
2. the best performing on the ESESR entailment pairs system;
3. the second best system on ESESR entailment pairs; and
4. the default algorithm that gives “yes” to all the entailments, due to the fact that the majority of the ESESR entailment pairs in the RTE2 test set are true entailments

Refer to the Table 3 to find the evaluation of the performance with respect to the baselines.

We are particularly interested in the balanced dataset, as we do not know the proportion of the true and false entailments of a given type in an arbitrary context.

Our system gets 80% accuracy on the unbalanced dataset and 59% accuracy on the balanced dataset. That means that our method performs noticeably better than the average of the methods from RTE2 Challenge and the “yes” to all baseline on both datasets. It scores about 18% higher than the average and 2% higher than the “yes” to all algorithm on the unbalanced dataset; and 9% higher than these two algorithms on the balanced dataset. Further, our results are higher for all but the best system in the Challenge for this subtype.

6 The conclusions and future work

In the current work we have identified a subtype of entailment pairs; presented a machine learner that distinguishes the subtype among the entailment pairs; and presented a probabilistic model that evaluates the conditional probability of the hypothesis given the text. We then evaluated the algorithm against a baseline and two other systems. The result is that the algorithm performs significantly better than the baseline (from 9% up to 18% better) and all but the best system in the Challenge for the type of entailment pairs we are interested in.

We plan to address other subtypes similar to ESESR entailment groups thus contributing more to the recognizing specific types of entailments.

References

Elena Akhmatova and Mark Dras. 2007. Syntactically encoded semantic relationships type of entailment pairs. Available from http://www.ics.mq.edu.au/~elena/pub.html.

Colin J. Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL*.

Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. 1996. Gate-a general architecture for text engineering. In *COLING*, pages 1057–1060.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *MLCW*, pages 177–190.

Ludovic Denoyer and Patrick Gallinari. 2006. The wikipedia xml corpus. *SIGIR Forum*, 40(1):64–69.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. A lexical alignment model for probabilistic textual entailment. In *MLCW*, pages 287–298.

Maria Lapata. 2001. A corpus-based account of regular polysemy: The case of context-sensitive adjectives. In *NAACL*, pages 63–70.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems*.

Tom Mitchell. 1997. *Machine Learning*. McGraw Hill.

Randolth Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A grammar of contemporary English*. Longman, Singapore.

Daniel Sleator and Davy Temperley. 1991. Parsing english with a link grammar. Available at http://www.link.cs.cmu.edu/link/papers/index.html.

Lucy Vanderwende and William B. Dolan. 2005. What syntax can contribute in the entailment task. In *MLCW*, pages 205–216.

Lucy Vanderwende, Arul Menezes, and Rion Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *2nd PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Ian H. Witten and Eibe Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Proceedings of the Australasian Language Technology Workshop 2007, pages 13-20

Measuring Correlation Between Linguists' Judgments and Latent Dirichlet Allocation Topics

Ari Chanen and Jon Patrick
School of Information Technologies
University of Sydney
Sydney, Australia, 2007
{ari,jonpat}@it.usyd.edu.au

Abstract

Data that has been annotated by linguists is often considered a gold standard on many tasks in the NLP field. However, linguists are expensive so researchers seek automatic techniques that correlate well with human performance. Linguists working on the ScamSeek project were given the task of deciding how many and which document classes existed in this previously unseen corpus. This paper investigates whether the document classes identified by the linguists correlate significantly with Latent Dirichlet Allocation (LDA) topics induced from that corpus. Monte-Carlo simulation is used to measure the statistical significance of the correlation between LDA models and the linguists' characterisations. In experiments, more than 90% of the linguists' classes met the level required to declare the correlation between linguistic insights and LDA models is significant. These results help verify the usefulness of the LDA model in NLP and are a first step in showing that the LDA model can replace the efforts of linguists in certain tasks like subdividing a corpus into classes.

1 Introduction

Since linguists are expensive to employ, there is a preference in most NLP projects to use automatic processes especially where it can be shown that the automatic process approaches the performance of the linguists. Several linguists were used on the ScamSeek project (Patrick, 2006). ScamSeek was created for the Australian Securities and Investments Commission (ASIC) government agency to identify financial scam websites based on the linguistic properties of the webpage content. A major task they performed by the project linguists was to partition the corpus into classes. Besides defining the classes in terms of the documents assigned to them, the linguists also identified phrases they believed were indicative of each class.

The LDA corpus model (Blei, 2004) can automatically generate a likely set of corpus topics and subdivide the corpus words among those topics. We will show that there are similarities between the task the LDA performs and the tasks the ScamSeek linguists performed. This paper attempts to determine to what degree LDA topics correlate with the judgments of linguists in partitioning a corpus into document classes.

Formally, we set a null hypothesis, H_0, to claim that the relationship between the linguists' document classes and LDA topics is random. The alternative hypothesis, H_a, claims those document classes and the topics have a significant amount of correspondence or correlation between them. In order to measure how significant the correlation is, principled methods of measuring the statistical significance of the correlation must be found. If the p-value for the correlation between a document class and the **best** correlating topic for that class is less than $\alpha = 0.05$, then H_0 will be rejected in favor of H_a. The determination of the p-values are discussed in the Methods section.

2 Background

2.1 LDA Model

The LDA is a Bayesian, generative corpus model which posits a corpus wide set of k topics from which the words of each document are generated. In this model, a topic is a multinomial distribution over terms. According to the LDA model, an author first determines, through a random process, the topic proportions of a new document. Thereafter, the author chooses a topic for the next word and then draws that word randomly according to the chosen topic distribution.

The LDA model can be represented as a graphical model as shown in figure 1. Graphical models represent the dependencies between probabilistic model hyper-parameters and variables. A good introduction can be found in (Buntine, 1995). The LDA model includes two hyper-parameters, α and β as well as three random variables (RV's), $\theta_{1:D}$, z and w, where D is the number of corpus of documents.

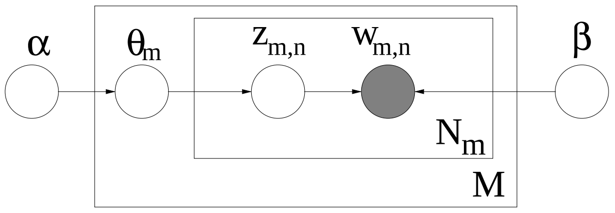


Figure 1: The LDA graphical model

α takes a scalar value that affects the amount of smoothing of the symmetric Dirichlet (*dir*) distribution that produces the multinomial (*multi*) distributed θ_m, representing the topic proportions for document m. The hyper-parameter β is a $k \times V$ matrix of probabilities where V is the size of the corpus vocabulary. Each row of β is a topic multinomial where $\beta_{ij} = p(w = j|z = i)$. The RV z is an index variable that indicates which topic was chosen for each document word (Steyvers and Griffiths, 2005)(Blei, 2004).

Formally, each document m is assumed to be formed by the following generative steps:

1. Choose proportions $\theta_m|\alpha \sim Dir(\alpha)$.
2. For $n \in \{1, \cdots, N_m\}$:
   (a) Choose topic $z_{m,n} \sim Multi(\theta_m)$
   (b) Choose word $w_{m,n}$ from $p(w_{m,n}|z_{m,n}, \beta_{z_{m,n}})$

where N_m is the number of words in document m.

Under graphical model notation, shaded elements are observed and unshaded elements are latent. Thus, the circle denoting the w element, representing the words of a document, is the only observed element. The other elements are latent. In order for the LDA model to be useful in practical settings, these latent RV's and hyper-parameters need to be estimated.

If α and β are assumed fixed, then the posterior probability w.r.t. θ and z can be expressed as follows:

$$\begin{aligned} p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) &= \frac{p(\mathbf{w}|\theta, \mathbf{z}, \alpha, \beta)p(\theta, \mathbf{z}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)} \\ &= \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{\int_\theta \sum_{\mathbf{z}} p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)\, \mathrm{d}\theta} \end{aligned}$$

Unfortunately, this posterior probability is intractable to calculate due to the integral over the Dirichlet variable. There are several methods for approximating θ and z. The LDA topic data used in this research was induced using the mean field variational method which is an iterative algorithm that converges on estimates of θ and z for each document and each word in those documents. Once these estimates have been obtained, then estimates for α and β can be obtained by holding the values of θ and z fixed and using an empirical bayes estimation technique. By alternating between the mean field variational estimation and the empirical bayes estimate the values of the latent elements are guaranteed to eventually converge to stable values. For further details on this latent element estimation technique see (Blei, 2004).

In the experiment section the topic proportions $\theta_{1:D}$ of each document and the topic rows of β will be compared to similar data produced by linguists.

Table 1 shows the 25 top terms from four sample topics induced from the ScamSeek corpus for a 64 topic model. The top terms are constructed by sorting a topic's multinomial terms by term probability in descending order. The first row of the table shows the name of the linguists' document class that is most correlated[1] with the topic terms shown in the rows below. The last row shows the cumulative probability mass that the top 25 topic words account for. Three of these example topics are most associated with scam classes. Only the topic most associated with the **Licensed Operator** class is a non-scam class. A good indicator of this is that the word, "risk", is one of the most probable terms.

Nigerian scam	Mail scam	Licensed operator	Online betting
i	you	investment	online
my	i	investments	casino
you	your	you	betting
your	will	invest	gambling
me	name	your	casinos
am	post	risk	games
all	money	investing	sport
will	now	funds	vegas
we	make	can	las
would	list	returns	odds
not	newsgroups	investors	sportsbook
thanks	only	shares	free
thank	just	their	sports
course	if	fund	internet
one	my	return	betted
work	message	over	best
some	all	australian	book
money	article	more	wagering
good	step	managed-funds	guide
just	more	portfolio	sports-betting
now	made	not	gaming
well	me	property	football
time	letter	investor	line
great	people	cash	your
0.31	0.30	0.27	0.41

Table 1: The 25 top terms from four sample topics induced from the ScamSeek corpus for a 64 topic model.

2.2 Monte-Carlo Simulation

In this research, we want to measure the strength of the correlations between classes and topics. One challenge of this task is that the classes and the topics are in different forms and the topics are non-parametric distributions. We achieve this aim by utilizing one form of the Monte Carlo Simulation method where a number of random pseudo-LDA models are produced. The correlations between the linguists classes and both the real LDA model as well as the pseudo-models are measured. The correlation scores between all the pseudo-models and the linguists classes are sorted and the real model's correlation score is ranked against the pseudo-models. The percentage of pseudo-model scores that the real model score beats is taken to be the significance level of the real correlation.

Let the correlation between the classes and the LDA topics be called the **real** correlation. From the ranking of the real correlation within all the random correlations an approximate p-value is derived. Let r be the number of random correlations that are the same or better than the real correlation and let n be the number of random models. Then[2]:

$$p\text{-value} = r/n$$

(B. V. North and Sham, 2002) report that using Monte-Carlo procedures to calculate empirical p-values has become commonplace in statistical analysis and give three major motivating factors:

1. Many test statistics do not have a standard asymptotic distribution.
2. Even if such a distribution does exist, it may not be reliable in realistic sample sizes.
3. Calculation of the exact sampling distribution through exhaustive enumeration of all possible samples may be too computationally intensive.

Reason #1 definitely applies to the case of trying to find a distribution for possible LDA models. The LDA estimation algorithm is nonparametric itself so there is no reason to think it would produce topic multinomials that fit a parametric distribution. Reason #2 does not apply. Reason #3 is a major factor for using Monte-Carlo techniques in the case of this research. Each randomised topic has $N = 18,000$ terms. To randomise a LDA model each topic has its terms and probabilities shuffled in a pseudorandom matter. There are $N!$ different shuffles for each topic which is for all practical purposes infinite in this case.

[1]The correlation measure used to determine the most correlated class is the distributional intersection (DI) measure which is described later in the methods section.

[2]There is some dispute as to whether r/n or $(r+1)/(n+1)$ is the better p-value estimator. (Ewens, 2003) and (Broman and Caffo, 2003) prove that $(r+1)/(n+1)$ is biased so we use r/n here.

3 Similarities and differences between document classes and LDA topics

The LDA generative corpus model assumes that every corpus document draws its terms from κ topics, where κ is a parameter of the LDA model. One of the products of the LDA model estimation process is a γ-vector for each document which gives the estimated distribution of a document's terms over the topics. Normalizing this vector by dividing by the total number of document terms gives the document topic proportions which is the same information that the LDA model's θ_m RV represents for a given document m.

Unlike topics, the document classes the linguists constructed are meant to be mutually exclusive; a document may belong to one and only one of those classes. Although this is a significant difference between topics and these document classes, in practice the two are not too dissimilar. An analysis of all the normalised γ-vectors shows that, on average, each document devotes around 60% of its terms to a major topic, and allocates between 4-20% of its remaining content to each of four or five minor topics, leaving only small amounts of the topic mass to the rest of the topics. This pattern seems to hold irrespective of the number of topics used to generate the LDA model, as table 2 shows. Since most documents have a single topic with more than a majority of the topic mass, we will assume that topics can approximate the behavior of document classes.

	Topic rank						
Topics	1st	2nd	3rd	4th	5th	6th	7th
8	61.0	21.7	10.0	4.5	1.8	0.7	0.2
16	58.8	20.7	9.8	5.0	2.7	1.4	0.7
32	55.4	19.7	10.0	5.8	3.5	2.1	1.3
64	57.5	17.8	9.2	5.6	3.5	2.2	1.4
128	61.6	16.8	8.2	4.7	2.9	1.8	1.1
256	69.1	14.7	6.6	3.7	2.2	1.3	0.8
mean	60.6	18.6	9.0	4.9	2.8	1.6	0.9

Table 2: The average percentage of the 7 top ranked topics from each document in six different LDA models.

In addition to creating document classes, the linguists also created **motif** classes to embody certain qualities of documents that transcend the document classes. In this way, the motifs are closer to topics than document classes. The linguists identified characteristic phrases for the motif classes just as they did for the document classes. An example of a motif class is one called the **persuasion** class which has indicative phrases that are common to many scams in which a scammer tries to persuade victims to do something. Many of the scam documents exhibit some of these persuasion phrases. Unfortunately, exact phrases cannot be revealed because parts of the ScamSeek project are proprietary.

For the remainder of the paper, the term **classes** will be used to signify both document classes and motif classes.

4 Methods

Two types of methods were employed to estimate a p-value for the correlation between the linguists classes and the LDA topics: categorical and term-based. The categorical method attempts to measure the randomness in the relationship between the topics and the linguists' document classes. The term-based methods measure correlations between word distributions in the LDA topics and the linguists' class characteristic phrases.

LDA models were generated on 1917 documents from the ScamSeek corpus. Eight models were induced with the following numbers of topics: $2, 4, 8, 16, 64, 128, 256$. These models are referred to as the "real" models to differentiate them from the random LDA models introduced below.

4.1 Using the χ^2 test

The χ^2 test(Devore, 1999) can be used to test if two categorical variables are statistically independent. A contingency table is used to show the counts of some entity for every possible pairing of categories, one from each of the two variables. The empirical counts are compared to the counts that would be expected if the two variables were independent.

The χ^2 experiments described in this section only utilise the document classes and not the motif classes.

The raw LDA γ-vectors give a document's term count for each topic therefore topics are categorical in this context. To make a document class into a categorical variable, the γ-vectors for all the documents in the same document class can be summed so that each cell contains the total term count for one topic over all the documents in that class. Then, each cell (i, j) of the χ^2 contingency table will hold the total number of words from document class i that were assigned to topic j.

There is one problem with using the χ^2 test in this setting. Completely correct usage of the χ^2 test requires that each joint event from the contingency table is independent of all the others. However, according to (Blei, 2004, pg. 20), under LDA, the terms of the document are exchangeable, meaning that their order does not matter. This implies the terms are not independent of each other but rather *conditionally independent* with respect to the latent topics. Because of this potential problem, any results must be viewed with some caution.

The χ^2 statistic was calculated using each of the eight LDA models to determine the relationship between the document classes and the topics. These tests all indicated that the relationship was highly significant with a p-value of zero.

To verify this result, control experiments were performed where 10 random test sets were generated by shuffling the documents assigned to each class. The χ^2 test was run on each of the randomised sets. For the random sets, the χ^2 statistic was much lower than the value obtained from the real class assignments. Unexpectedly, the calculated p-value was still zero, indicating that even the randomised tests were highly significant.

We concluded that this method of applying the χ^2 test was not appropriate for the task of rejecting H_0, and that the most likely reason is that the document words are not completely independent.

4.2 Using Monte-Carlo Simulation

Next, we turn to a term-based method of trying to verify the H_a hypothesis, using word distribution correlations between topics and classes rather than a categorical analysis. To test this hypothesis Monte-Carlo simulation was used as described in the Background section 2.2. Futher details are provided in there section.

Again in this method, an approximate p-value is calculated from the ranking of real correlations within a sorted list of pseudo-correlations. The real correlations are between the words of the linguists' class characteristic phrases and real LDA topics while the pseudo-correlations are between those phrase words and a set of randomly generated pseudo-topics.

4.2.1 Forming the random LDA models

To begin with, for each of the eight real models (models with 2, 4, 8, 16, 32, 64, 128, 256 topics), one hundred randomised models were generated. Real LDA models have topics that concentrate most of their probability mass on a relatively small number of terms compared to the total number of terms in the distribution. The method of randomization was chosen so as to maintain the same level of probabilistic "clumpiness" in the random topics. To form a pseudo-random LDA model from a real model, for each real topic, the terms and their probabilities are separated. To form a pseudo-topic, the terms are shuffled and assigned to one of the pre-existing multinomial probabilities from the real model's corresponding topic.

4.2.2 Correlating one class with one LDA topic

Again, we are trying to rank the **best** correlation of a real topic with a class among the correlations of that class with the best correlations among all the pseudo-topics in each randomised LDA model. This section defines some notation needed in discussing these class/topic correlations. This notation assumes a specific model (defined by the number of topics) and a specific correlation measure have been chosen. Different kinds of correlation measures will be explained below.

Below, classes are referred to with the index i. Topics are referred to with the index k. An index of r refers to the one real model while an integer index j refers to one of the 100 random models.

In our notation, C_{irk} , refers to the correlation of the ith class and real model's kth topic and C_{ijk} refers to the correlation of the ith class and jth random model's kth topic.

In order to obtain the p-value for each class, correlation measures are calculated for each pairing of class and topic, both real and random. First C_{irk} is calculated for the one real model. Next, C_{ijk} is calculated for each of the hundred random LDA models. The real topic that shows the best correlation score with class i is, $\widehat{C_{ir}}$. Next, the procedure is performed on each of the 100 random LDA models so a correlation C_{ijk} between the class and each pseudo-topic k in each random model j is calculated. The best correlation for each random model $\widehat{C_{ij}}$ is found. The best correlations for each random model are sorted from least correlated to most correlated. Then the rank of the best real topic correlation is found within the sorted list of random best correlations. Since our criteria for significance is $\alpha = 0.05$ then for a given number of topics, type of correlation measure and class i, if:

$$\widehat{C_{ir}} > \widehat{C_{ij}}$$

for 95 of the 100 random models then we would take this as sufficient evidence that the null hypothesis can be rejected in favor of the alternative hypothesis.

The following subsections first define a method for forming multinomial distributions from class indicative phrase and next specifies three correlation measures defined on two multinomial distributions over the same range of terms.

4.2.3 A distribution from class phrases

The LDA topics are multinomial distributions over 18,000 terms. One way to correlate a class with these topics is to form a multinomial distribution from the class. The phrases that the linguists generated as being characteristic of the class can be used to achieve this goal. All the phrases are treated as though they came from a single document and processed in the same way the corpus documents were processed before the LDA models were built from them. This means joining terms together into multiword expressions (MWE) where appropriate and eliminating stopwords. Next, a histogram is formed with the terms and MWE's as elements. Finally, the count for each element is normalised by the total number of elements, thus yielding a probability distribution.

4.2.4 The vector cosine correlation measure

Now that a distribution, $\mathbb{C}_i$, has been formed for each class i, we can correlate them with each topic distribution, $\mathbb{T}_k$. One way to do this is by treating the two distributions as vectors in term space. The cosine of the angle between these two vectors can be seen as a measure of how similar the two distributions are. If the angle is zero then the two distributions are the same whereas if they are perpendicular they are maximally dissimilar. The cosine of the angle, θ, between $\mathbb{C}_i$ and $\mathbb{T}_k$ can be gotten from the formula:

$$\cos\theta = \frac{\mathbb{C} \cdot \mathbb{T}}{||\mathbb{C}||\,||\mathbb{T}||}$$

This measure will vary in the range $[0, 1]$ where 1 indicates the two distributions are identical.

4.2.5 The hypergeometric distribution correlation measure

The hypergeometric distribution (HD) (Devore, 1999, pg. 122) is often associated in with the probability of drawing lottery numbers that match the winning numbers. In the way the HD is used here, the winning lotto numbers are analogous to the words of the class indicated phrases and the most probable terms in a topic are analogous to the numbers on the lotto ticket. The HD assumes the following:

1. There is a population of size N to be sampled from.
2. Each member of the population can either be a *success* or a *failure*. There are M successes in the population.
3. A sample of size n is drawn in an independent and identically distributed manner.

$N = 18,000$ is the total number of terms in both the class and topic multinomial distributions. For a given class, $\mathbb{C}$, a term is defined as a *success* if it matches one of the terms from the class characteristic phrases, for a total of M possible successes. $\mathbb{C}_M$ denotes the set of those success terms. Now, given the kth topic $\mathbb{T}_k$, $\mathbb{T}_{k,\hat{M}}$ is the set of the M most probable terms in that topic. Let $\mathbb{I}_k$ be the number of elements in the intersection set $\mathbb{C}_M \cap \mathbb{T}_{k,\hat{M}}$. The probability of $\mathbb{I}_k$, for the subset of hypergeometric distributions where $n = M$ is:

$$P(\mathbb{I}_k|N, M) = \frac{\begin{pmatrix} M \\ \mathbb{I}_k \end{pmatrix}\begin{pmatrix} N-M \\ M-\mathbb{I}_k \end{pmatrix}}{\begin{pmatrix} N \\ M \end{pmatrix}}$$

The lower the above probability is the greater the chance of correlation between $\mathbb{C}_i$ and $\mathbb{T}_k$. Since this probability can be extremely small, log probabilities are used to express it. Therefore, the range of this correlation measure is $(-\infty, 0)$.

4.2.6 The distribution intersection correlation measure

Another simpler measure of distribution correlation is the amount of probability mass the two distributions share. The formula for calculating this measure for the distributions of the ith class $\mathbb{C}_i$ and the kth topic $\mathbb{T}_k$ is:

$$DI(\mathbb{C}_i, \mathbb{T}_k) = \sum_{j=1}^{N} \min(\mathbb{C}_i[j], \mathbb{T}_k[j])$$

where DI stands for "distribution intersection", N is the total number of terms in each distribution. This measure also has the range of $[0, 1]$ with 1 meaning the two distributions are the same.

5 Results

5.1 Results for using 100 random models

To reiterate the problem definition, we seek to determine if there is enough evidence to reject the null hypothesis in favor of the alternative hypothesis. Since we set $\alpha = 0.05$, this means that the real model must have a better correlation should score then 95% of the random models, for a given model and type of correlation measure. The final performance results are measured in terms of the percentage of classes where the H_0 could be rejected. In many cases, the real model did better than all 100 of the pseudo-models so results are also provided for the case where we had set our H_0 rejection threshold to $\alpha = 0.01$.

Three different correlation measures were used: vector cosine (VC), distribution intersection (DI), and hypergeometric distribution (HD). Table 3 shows the results for the models of various numbers of topics and for the three correlation measures. The table gives the percentage of classes that have p-values less than 0.05 and 0.01.

For the DI correlation measure, there was enough evidence to reject H_0 at $\alpha = 0.05$ for comfortably over 90% of the classes for all eight LDA models classes and this was nearly true at $\alpha = 0.01$.

The results for the VC correlation measure are less significant where only five out of eight of the models could claim to reject H_0 for more than 90% of the classes for $\alpha = 0.05$. Also, the correlation level fell off for the models with higher numbers of topics (64, 128, 256) for $\alpha = 0.05$ and there was much larger gap between the correlations at $\alpha = 0.05$ and $\alpha = 0.01$ compared to the much smaller gap for the DI results. One problem with the VC measure is that the angle between the problem with $\mathbb{C}_i$ and $\mathbb{T}_k$ vectors is only measuring differences in the terms that have nonzero probabilities. Therefore, this measure is less restrictive, allowing for a greater chance that a random topic may have the right combination of terms so that its correlation with a class will be better than the corresponding real model's best correlation.

	Vector Cosine		Distrib. Intersection		Hypergeometric	
Topics	%<0.01	%<0.05	%<0.01	%<0.05	%<0.01	%<0.05
2	79.6	91.8	89.8	100	83.7	87.8
4	77.6	95.9	91.8	100	85.7	91.8
8	79.6	95.9	91.8	95.9	85.7	91.8
16	77.6	93.9	91.8	95.9	85.7	87.8
32	73.5	91.8	93.9	95.9	85.7	87.8
64	63.3	85.7	89.8	93.9	91.8	93.9
128	61.2	79.6	91.8	98	91.8	95.9
256	69.4	75.5	87.8	93.9	98	98

Table 3: The %'s of classes having $p < 0.01$ and $p < 0.05$ for 3 different correlation measures using 100 random LDA models for the Monte-Carlo simulation .

The HP measure was the worst that $\alpha = 0.05$ but in the middle for $\alpha = 0.01$. one interesting trend is that it does much better then the VC measure for high topic models (128, 256.)

The DI correlation measure shows the generally higher correlation scores which does not necessarily mean it is the best measure for our purpose. Yet, it is a straightforward measure of the correlation between two distributions and it is the most straightforward to calculate.

5.2 Results for using 1000 random models

The evidence that LDA topics may mirror certain parts of linguistic instincts looks fairly convincing from the tests using 100 random LDA models. To add weight to these results more Monte-Carlo simulations were run using 1000 completely different random LDA models. The results are shown in table 4.

Notice that the column reporting the results for the DI correlation measurement and with $\alpha = 0.05$, has the exact same values as those for hundred

	Vector Cosine		Distrib. Intersection		Hypergeometric	
Topics	%<0.01	%<0.05	%<0.01	%<0.05	%<0.01	%<0.05
2	85.7	93.9	93.9	100	85.7	91.8
4	81.6	95.9	91.8	100	85.7	91.8
8	81.6	93.9	93.9	95.9	87.8	91.8
16	79.6	93.9	91.8	95.9	87.8	89.8
32	77.6	91.8	91.8	95.9	87.8	87.8
64	73.5	85.7	91.8	93.9	93.9	93.9

Table 4: The %'s of classes having $p < 0.01$ and $p < 0.05$ for 3 different correlation measures using 1000 random LDA models for the Monte-Carlo simulation .

model simulation in table 4.[3] If average the percentages for the 2,4, 8,16 ,32 and 64 topic models for the hundred and thousand models test for each column from tables 3 and 4 then three of the columns are exactly the same and two have a change of 1% are less. That the change from the hundred model simulation to the thousand model simulation was minimal is a good sign that this technique of measuring the correlation is stable and adds weight to its validity.

6 Conclusion

Real LDA models and the judgments of the linguists in classifying the corpus do appear to be significantly well correlated when compared to random LDA models.

The distribution intersection correlation is used successfully here as a simple yet effective way of measuring the correspondence between the phrases that the linguists came up with to characterise classes and the words of the topics. The hypergeometric distribution and vector cosine correlation measures also showed significant correlation strengths but to a lesser degree than the DI measure.

The results reported on here should add to the confidence of the NLP field that the LDA corpus model, even though it is only an approximate statistical model, can correspond to human judgments as to what the salient features of a document corpus are.

[3]To have the exact same values may seen strange at first but these are percentagesof classes that beat more than 5% of the random models. Some of the classes that did well in the hundred model test did not meet the significance cut off in the thousand model test and vice versa but the end result was the same.

Acknowledgements

The authors would like to thank Dr. Alex Smola and Dr. Sanjay Chawla for their input into this research.

References

D. Curtis B. V. North and P. C. Sham. 2002. Letter to the editor on "simulation-based p values: Response to north et al..". *Am J Hum Genet*, 71:439–440.

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

Karl W. Broman and Brian S. Caffo. 2003. Letter to the editor on "simulation-based p values: Response to north et al..". *Am J Hum Genet*, 72:496.

Wray L. Buntine. 1995. Graphical models for discovering knowledge. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. S. Uthurasamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 59–83. MIT Press.

Jay L. Devore. 1999. *Probability and Statistics for Engineering and the Sciences*. Duxbury.

Warren J. Ewens. 2003. Letter to the editor on "on estimating p-values by monte carlo methods". *Am J Hum Genet*, 72:496–497.

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

M. Steyvers and T. Griffiths. 2005. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum.

Proceedings of the Australasian Language Technology Workshop 2007, pages 21-30

TAT: an author profiling tool with application to Arabic emails

Dominique Estival, Tanja Gaustad,
Son Bao Pham, Will Radford
Appen Pty Ltd
Chatswood NSW 2067, Australia
{destival,tgaustad,sbpham,wradford}@appen.com.au

Ben Hutchinson[*]
Google
Sydney, Australia
benhutch@google.com

Abstract

This paper reports on the application of the Text Attribution Tool (TAT) to profiling the authors of Arabic emails. The TAT system has been developed for the purpose of language-independent author profiling and has now been trained on two email corpora, English and Arabic. We describe the overall TAT system and the Machine Learning experiments resulting in classifiers for the different author traits. Predictions for demographic and psychometric author traits show improvements over the baseline for some of the author traits with both the English and the Arabic data. Arabic presents particular challenges for NLP and this paper describes more specifically the text processing components developed to handle Arabic emails.

1 Introduction

The goal of the TAT project is to develop a language-independent Text Attribution Tool (TAT) which can provide information on authors for a variety of document types and a range of languages. In the first implementation, the TAT has been developed for profiling the authors of email messages in English and Arabic, with other languages to be added in the future. (Estival et al., 2007) describes the Machine Learning experiments and the results obtained for the English email data. In this paper, we will focus on the results obtained for Arabic emails and will describe the aspects of the TAT which are particular to the processing of our Arabic email data.

We first introduce the two tasks of author attribution and author profiling in Section 2. In Section 3, we describe the data set on which our tool was trained and tested. Section 4 contains an overall description of the TAT system, followed by a more specific description of the processing steps required for Arabic emails in Section 5. After presenting the general experimental setup in Section 6, we report on the results achieved for several demographic and psychometric traits in Section 7 and propose our general conclusions in Section 8.

2 Author attribution and author profiling

The ability to recognise the identity or certain characteristics of authors automatically from texts has a number of potential applications. Author identification and author profiling can provide valuable information for marketing intelligence (Glance et al., 2005), while the rapidly growing field of sentiment analysis and classification (Oberlander and Nowson, 2006) is another application where profiling can make a contribution. Also, author profiling forensics may be helpful in narrowing the choice of potential authors when identifying the source of a threat (Corney et al., 2002; Argamon et al., 2005; Abbasi and Chen, 2005a).

Author attribution is the task of deciding for a given text which author, usually from a predefined set of authors, has written it. Historically, author identification has its roots in literature, with studies of the Bible (Friedmann, 1997), Shakespeare (Ledger and Merriam, 1994) or the Federalist Pa-

[*]The work presented in this paper was carried out while this author was working at Appen Pty Ltd.

pers (Mosteller and Wallace, 1964). Recently, author identification has also been applied to more informal texts, such as emails (de Vel, 2000; de Vel et al., 2001; de Vel et al., 2002), newsgroup messages (Zheng et al., 2003; Zheng et al., 2006) or blog entries (Koppel et al., 2005; Oberlander and Nowson, 2006).

Author profiling is the task of predicting one or more traits for the author of a text and the author profile consists of the set of traits predicted for that author. A major difference between author profiling and author attribution is that it is possible to predict author traits even when the training data does not contain any document by the actual author. Another difference is that greater accuracy can be expected for author profiling when the training data is made up of more authors, because the models for each trait are expected to be more robust. The accuracy of author identification, on the other hand, can be expected to decrease if the number of potential authors is increased.

In (Estival et al., 2007), we presented our work on author profiling for English emails and discussed the literature on previous research in that area for English texts. For Arabic texts, very few studies have been published in the area of author attribution; the only work we know of can be found in (Abbasi and Chen, 2005b) and (Abbasi and Chen, 2005a). We are not aware of any work on author profiling for Arabic.

For author attribution in Arabic, Abbasi and Chen (2005a) apply two different machine learners to a dataset of Arabic forum messages by 20 authors (with 20 messsages per author). The feature set comprises lexical, syntactic, structural and content-specific features, including a number of features specifically tailored to Arabic. These relate mostly to inflection (counting roots rather than inflected words), word length (adjusting the range for the Arabic word length distribution to reflect the fact that Arabic words tend to be shorter than English ones) and elongation dashes (excluding them from word length measurements, but tracking their usage). The main conclusion reached is that using an SVM classifier with all the features achieves the best accuracy on their data set, but that the overall performance is lower than for English.

Abbasi and Chen (2005b) use the same approach, and in addition include an in-depth comparison between the feature sets and results for English and Arabic on forum messages.

3 Data

The data collected for this project and used for training the TAT consists of two sets of emails from 1033 English speakers and from 1030 Arabic speakers. The English data set contains emails written by both native and non-native speakers of English: native speakers of US English, native speakers of Australian/NZ English, native speakers of Spanish from the US and native speakers of Egyptian Arabic from Egypt. The Arabic data set contains emails written by native speakers of Egyptian Arabic, as described in more detail below. In the rest of this section, we focus on the data collection and validation processes for Arabic emails.

3.1 Data Collection

For the Arabic email data set, the collection was conducted in Egypt and all the writers were native speakers of Egyptian Arabic. Compared with the English email data set, a special feature is the encoding of the input for Arabic email. The widespread use of the Internet and even more of text messages via mobile phones without the possibility of Arabic script input has led to the use of the Latin alphabet and the development of some transliteration conventions in the Arabic speaking world. Even though Arabic keyboards are now more common, people still sometimes write email using a Latin keyboard. However, there are no strict rules for the conversion of Arabic script into Latin characters and the way of writing emails with a Latin keyboard varies greatly according to dialect or country, and even across individuals.

Table 1 gives an overview of the number of authors, number of emails and number of words for both the Arabic and English data set. We also include the proportion of emails in Arabic script and in Latin script for the Arabic data.

The data collection process for the Arabic data differed slightly from the process described in (Estival et al., 2007) for the collection of the English data, in that the respondents had to come to a central collection location. Nevertheless, the recruitment pro-

Collection	**# authors**	**# emails**	**# words**
Arabic	1,030	8,028	2,153,333
Arabic script		7,267	
Latin script		761	
English	1,033	9,836	3,367,173

Table 1: Overview of the collected English and Arabic email data.

cess also included notification of privacy and the assurance that their identity would be protected. The respondents agreed to fill out a web questionnaire to provide demographic and psychometric information about themselves and to donate at least ten email messages.

Demographic traits cover basic demographic information about the author, namely age, gender, and level of education. The psychometric traits of the Arabic collection are based on a customized version of the short Eysenck Personality Questionnaire Revised (EPQR-S) (Francis et al., 2006), consisting of 51 questions. These questions aim to identify four psychometric traits: extraversion, lie (or social desirability), neuroticism (or emotionality), and psychoticism (or toughmindedness).

After completing the questionnaire, the writers either composed new email messages which they then sent to their recipients and also forwarded to the data collection email address, or directly forwarded previously sent emails, e.g. from their email client "SentBox".

We collected at least 10 emails from each author, for a total number of 2000 words per author in the Arabic data set. Research has shown that the more complex morphology of Arabic (combined with a rich vocabulary) leads to a higher degree of inherent sparseness in Arabic data compared to similar English data. This suggests that larger amounts of data are needed for statistical Natural Language Processing (NLP) applications in Arabic (Goweder and Roeck, 2001). Therefore, while the minimum was set at 1000 words per writer for the English data, it was 2000 words per writer for the Arabic data.

3.2 Data Validation

The email messages were checked manually to filter out erroneous content such as foreign language emails or forwarded chain letters and to ensure consistency and accuracy of the documents in the corpus. As with any data collection of email, plagiarism and copying were issues that required careful checking of all the data received and we developed a plagiarism detecter to reject emails which had already been submitted.

For both the English and the Arabic data collections, a minimum number of 5 lines per email had been set. Because Arabic writers often do not use new lines or new paragraphs, this had to be measured visually on the screen for the Arabic data.

In summary, the final Arabic data set contains a combined total of 8028 email messages, from 1030 writers who met the following criteria: 1) a valid questionnaire was received for each author; 2) there are at least 5 valid email messages for the author; and 3) the total word count for that author's valid email messages is at least 2000 words.

4 System Description

Figure 1 gives a high-level overview of the TAT system. The system includes several data repositories and a number of components for deriving features and for building classifiers. While the current focus is on processing email input in English and in Arabic, the underlying processing architecture is language independent and will be extended to other types of documents and to other languages.

The modular processing architecture is organized around a chain of processing modules. This chain allows flexible experimentation with various processing modules. At the same time, it provides a robust software framework that promotes reuse and supports flexible deployment options by connecting specific modules together for the task at hand.

Each processing module consumes objects from its input, such as documents, and emits objects containing the analysis of the input objects. The analysis of a document is represented in stand-off anno-

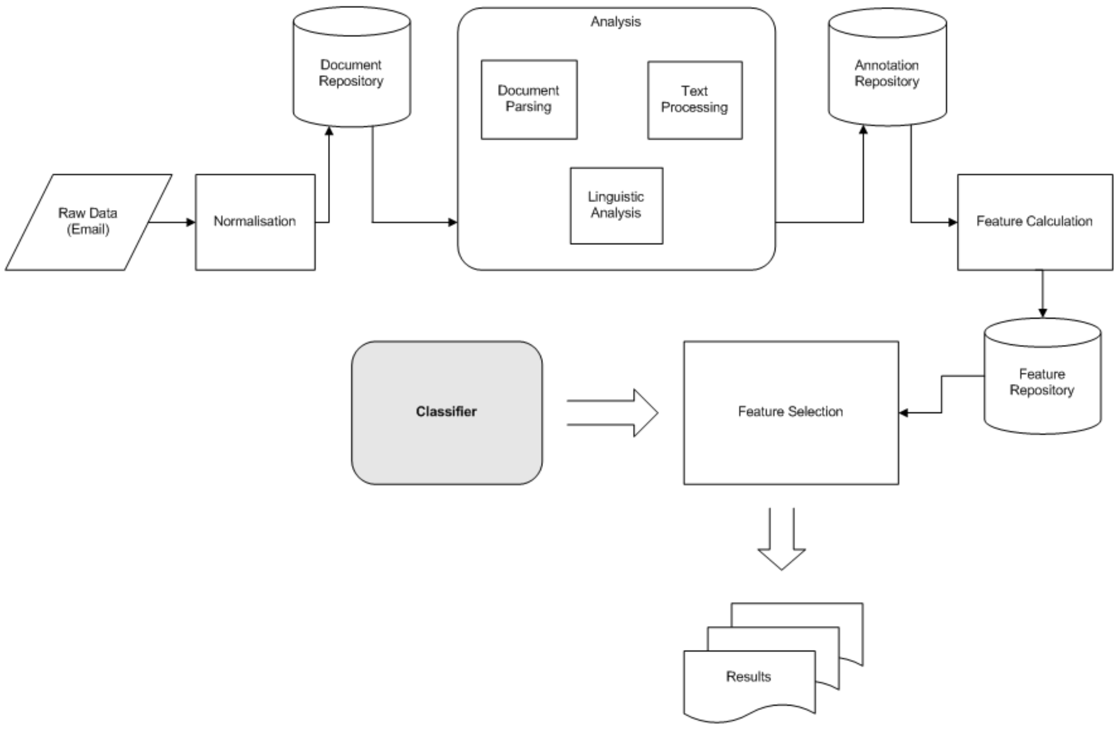


Figure 1: TAT System Diagram

tations and saved in a common structure called the Annotation Repository.

The process is data-driven in the sense that the output of each processing module depends on its input rather than on the way the module is combined with other modules in the chain. This enables processing modules to be reused in different processing chains and in different control environments as long as the input requirements are met.

5 Arabic Text processing

After the initial data collection and validation, several processing steps are needed, including character encoding normalisation, document structure parsing, text processing and linguistic analysis. The results of this processing provides the input to the feature extraction phase. A more extensive description of the processing steps for the English data can be found in (Estival et al., 2007).

Arabic emails present a number of challenges for NLP, including different ways of writing Arabic in Latin script (so-called “franco-arabic”); typical spelling variants in the Egyptian dialect of Arabic and possible spelling normalisation; morphology; English loanwords, spelling errors, and typos.

The Arabic processing modules perform the following functions:

1. Language and character-set identification. The language is tagged as either English or Arabic and the script as either “roman” or “arabic”, using character-based language models.
2. Document structure parsing. This stage distinguishes the text of the email written by the author from other types of elements (adverts, signatures, reply lines, or quoted text).
3. Tokenisation. The input text is split into paragraphs, sentences and words. Words are mainly strings of alphanumeric characters, with a few major exceptions for transliterated Arabic: some two character sequences, e.g. “ 3’ ”, can indicate a single Arabic glyph, namely غ.
4. Character set normalisation. In order to achieve a normalised, unambiguous input for Arabic and Latin scripted texts, words are converted to ASCII only characters using the Buckwalter transliteration scheme (Buckwalter, 2000; Buckwalter, 2002). The Buckwalter scheme

is very commonly used in NLP for Arabic because it represents Arabic orthography strictly on a one-to-one basis (unlike common romanization schemes that add morphological information not actually expressed in the Arabic script).

5. Spelling normalisation. Informal written Arabic often contains non-standard spellings (Buckwalter, 2004; Goweder and Roeck, 2001). We have therefore normalised the spelling based on common spelling variations in Egyptian Arabic. Examples include word final ى becoming ي or word final خ becoming ة. Also, depending on the context, ء can take different chairs or appear by itself on the line. We normalise the hamza chair to أ.
6. Morphological analysis. By comparing the normalised version of a given word with dictionaries of prefixes, suffixes and clitics, linguistic features such as number and person are added, and the remainder of the word is tagged as a stem. The root letters of the stem are then predicted using simple linguistic heuristics, e.g. long vowels are less likely to be root letters.
7. Lexicon taggers. The following word classes are currently tagged: conjunctions, prepositions, pronouns, discourse particles, interrogative pronouns, English loan words, colloquial Egyptian words, and frequent roots.
8. Named Entity Recognition. Named entities which are not language-dependent are tagged. These include numeric dates, numeric times, phone numbers, email addresses and URLs.

6 Experimental setup

Recent years have seen an exponential increase in the use of statistical language processing techniques for a wide range of tasks. In particular, text and document classification problems greatly benefit from statistical approaches. Author profiling can be viewed as a type of document classification task, where the classes correspond to traits of the authors. These traits are arranged along various dimensions and the different options for each dimension are mutually exclusive. For example “male” and “female” are the possibilities for the gender dimension. For each dimension, the email and questionnaire data are used to construct classifiers, using a range of Machine Learning (ML) techniques.

Each document constitutes a single data instance for the purposes of the experiments. For each experiment, ten-fold cross-validation was used, so the results reported in Section 7 are on the entire data set. Once the best combination of ML classifiers, parameters and feature selection has been determined during development, that model is used to classify the test data to evaluate the performance for a given trait.

6.1 Traits and classes

We distinguish three different demographic and four psychometric traits in the experiments presented in this paper, namely age, gender, and level of education for the demographic traits, and extraversion, lie, neuroticism, and psychoticism for the psychometric traits (Eysenck and Eysenck, 1975). This information is extracted from the questionnaires.

For the traits taking numerical values, subjects were split into three classes based on the first and third quartiles. Table 2 summarizes the data distribution for each trait across these classes.

A major difference with regard to the psychometric traits described in (Estival et al., 2007) lies in the fact that we used two different questionnaires for English and for Arabic. The International Personality Item Pool questionnaire (IPIP) (Buchanan et al., 2005), which was used for collecting the English data, yields five psychometric traits: agreeableness, extraversion, neuroticism, conscientiousness, and openness (also referred to as the “Big Five”) (Norman, 1963). However the questionnaire for the Arabic data collection was based on the short form of the revised Eysenck Personality Questionnaire, the EPQR-S first developed for a study in Germany (Francis et al., 2006), and the adaptation of the EPQ for Arabic (Abdel-Khalek and Eysenenck, 1983). The EPQ (Eysenck and Eysenck, 1975) aims to analyze personality along four traits, namely extraversion, neuroticism, psychoticism, and lie. This entails that the Arabic results for the psychometric traits are not directly comparable to the English ones.

Demographics			
Age:	Gender:	Level of education:	
<25 (691)	Male (728)	No tertiary edu. (970)	
25 to 35 (236)	Female (302)	Some tert. edu. (60)	
>35 (103)			
Psychometrics			
Extraversion:	Lie:	Neuroticism:	Psychoticism:
low	low	low	low
medium	medium	medium	medium
high	high	high	high

Table 2: Traits and Classes, with frequencies in parentheses where applicable.

6.2 Features

For each document, a feature vector is calculated. A feature is typically a descriptive statistic calculated from both the raw text and the annotations. For example, a feature might express the relative frequency of two different annotation types (e.g. number of words/number of sentences), or the presence or absence of an annotation type (e.g. Advert).

For the Arabic data, 518 features were calculated. These were divided into several subgroups shown in Table 3. The main purpose of the groupings was to make more informed choices during the feature selection stage and to facilitate experimentation with various combinations of feature groups.

Character-level features cover features such as the frequency of punctuation characters or word length. Arabic-specific character features include information on spelling normalisation and Arabic special characters. Morphological features mainly encode information on number, person and gender markers, such as clitics or suffixes. Lexical features include certain POS tags (e.g. preposition) and whether a word is an English loanword or specific to the Egyptian dialect.

6.3 Classification algorithms and feature selection

Classifiers for predicting author traits from the linguistic features were trained using the WEKA toolkit (Witten and Frank, 2005). During training, classifiers are created by the selection of sets of features for each trait, and classifier parameters are tuned through cross-validation. To evaluate and test the classifiers, new documents are given as input and existing classifiers are selected to predict author traits.

The machine learning algorithms tested include decision trees (J48 (Quinlan, 1993), RandomForest (Breiman, 2001)), lazy learners (IBk (Aha et al., 1991)), rule-based learners (JRip (Cohen, 1995)), Support Vector Machines (SMO (Keerthi et al., 2001)), as well as ensemble/meta-learners (Bagging (Breiman, 1996), AdaBoostM1 (Freund and Schapire, 1996)). These algorithms were used in combination with feature selection methods based on either a feature subset evaluator together with a search method (consistency subset evaluator with a best-first search) or a single attribute evaluator with various numbers of attributes selected (χ^2, GainRatio, and InformationGain) (see chapter 10.8 in (Witten and Frank, 2005) for details).

7 Results and discussion

The results shown here were computed on the Arabic email data set described in Section 3 using the different classifiers and general setup introduced in Section 6. Table 4 shows the results on all seven traits (demographic and psychometric), including the respective baseline associated with each separate classification task. We also state which settings (ML algorithm, feature set, and feature selection) were used to achieve the results reported. Education and gender are both binary classification tasks, whereas age uses three classes. All the psychometric traits are divided into three classes (see Section 6.1 and Table 2 for details on the exact split).

The results show that for six out of the seven traits

Feature group	Description
all	all features for Arabic
arabicNamedEntities	language independent named entities, such as URLs
arabicChar	Arabic character features
arabicMorphological	Arabic morphology features
arabicLexical	Arabic lexicon and word features

Table 3: Feature groups for Arabic.

Trait	ML algorithm	Features	Feature Sel.	Baseline	Results
Age:	Bagging	all except arabicLexical	InfoGain	70.09	72.10
Gender:	SMO	all	None	72.16	81.15
Education:	Bagging	all	InfoGain	93.62	93.66
Extraversion:	SMO	all except arabicMorphological	None	48.27	54.35
Lie:	Bagging	all	InfoGain	40.41	52.30
Neuroticism:	Bagging	all	InfoGain	43.42	54.93
Psychoticism:	Bagging	all	InfoGain	49.39	56.98

Table 4: Results for all demographic and psychometric traits on Arabic email data.

tested for, classification is significantly[1] improved over the baseline. For education, virtually no improvement can be seen which is due to the extremely skewed data (as indicated by the very high baseline of 93.62%). Even though the baselines for the other demographic traits are also quite high, our system still achieves a better classification accuracy for age and gender than the majority baseline.

The better result for gender can in part be explained by the fact that gender is morphologically marked in Arabic. One of the relevant constructions with respect to identifying an author's gender are predicative sentences with first person subjects. For example, in the Arabic equivalent of "I am happy", *happy* is morphologically marked as either feminine or masculine. Since our features include morphological information, our classifier detects gender differences very accurately. A more detailed analysis of the effects of each feature group on the prediction of gender (shown in Table 5) reveals that lexical features are also of great assistance.

Table 6 shows the results for English and Arabic demographic traits that are directly comparable. This seems to confirm previous results showing that predicting author traits for Arabic is likely to be more difficult than for English. One should not forget, however, that the baselines for all Arabic demographic traits are extremely high which means that little data is available for the minority classes.

For the psychometric traits, we achieve similar improvements over the baseline as for English. This is particularly encouraging, as most research on Arabic author attribution has shown results for Arabic to be lower than for English. It seems that predicting a profile rather than an identity might be advantageous for Arabic, or at least a viable back-off option.

From our results, two ML algorithms emerge as best performing for all traits examined, namely SMO and Bagging. Bagging seems to profit from feature selection whereas the Support Vector Machine based SMO algorithm does not show additional improvements when combined with feature selection. This differs slightly from our results with the English data set, where no clear conclusion could be drawn with regard to the usefulness of feature selection for different algorithms.

An analysis of the results shows that the highest accuracy is achieved by including all available features, with the exclusion of a single feature group for age (arabicLexical) and for extraversion (arabicMorphological).

[1]Significance was tested using a χ^2 test with p=0.01.

Feature Group	Best Accuracy	Decrease in Accuracy
all	81.15	0.00
all except arabicNamedEntities	80.79	-0.36
all except arabicMorphological	80.19	-0.96
all except arabicChar	79.99	-1.16
all except arabicLexical	77.44	-3.71

Table 5: Contribution of feature groups to improvements in gender prediction.

Trait	**English email data**		**Arabic email data**	
	Accuracy	Improvement over Baseline	Accuracy	Improvement over Baseline
Age	56.46	+17.03	72.10	+02.01
Gender	69.26	+14.78	81.15	+08.99
Education	79.92	+21.14	93.66	+00.04

Table 6: Comparison of results for demographic traits for English and Arabic.

8 Conclusion and future work

We have presented some results of experiments to automatically predict author traits from email messages. This work is of interest for a number of potential applications, from marketing intelligence to sentiment analysis. The results presented in this paper were conducted on the Arabic subset of the email data we have collected (containing approximately 8028 emails).

The experiments reported here were aimed at discovering how well a range of ML algorithms perform on our data set for three demographic and four psychometric author traits. Our results support the conclusions drawn in (Estival et al., 2007) that the chosen approach works well for author profiling and that using different classifiers in combination with a subset of available features can be beneficial for predicting single traits.

Future work will include the extension of the TAT to other document types and other languages. For Arabic text processing, it might be fruitful to investigate a more sophisticated analysis of words into roots (Darwish, 2002; de Roeck and Al-Fares, 2000).

Acknowledgements

This research was carried out within the framework of a BAA grant (IS-QD-2467). We would like to thank Judith Bishop for her important contribution to the project in successfully managing the data collection.

References

Ahmed Abbasi and Hsinchun Chen. 2005a. Applying authorship analysis to Arabic web content. In Paul B. Kantor et al., editor, *Intelligence and Security Informatics, Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2005)*, pages 183–197. Springer.

Ahmed Abbasi and Hsinchun Chen. 2005b. Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20(5):67–75.

Ahmed Abdel-Khalek and Sybil Eysenenck. 1983. A cross-cultural study of personality: Egypt and England. *Research in Behaviour and Personality*, 3:215–226.

David Aha, Dennis Kibler, and Mark Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Shlomo Argamon, Sushant Dhawle, Mosche Koppel, and James W. Pennebaker. 2005. Lexical predictors of personality type. In *Proceedings of the 2005 Joint Annual Meeting of the Interface and the Classification Society of North America*, St. Louis.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Tom Buchanan, John A. Johnson, and Lewis R. Goldberg. 2005. Implementing a Five-Factor personality inventory for use on the internet. *European Journal of Psychological Assessment*, 21:115–127.

Tim Buckwalter. 2000. Arabic transliteration. http://www.qamus.org/transliteration.htm.

Tim Buckwalter. 2002. Arabic morphological analyzer. Linguistic Data Consortium (LDC2002L49).

Tim Buckwalter. 2004. Issues in Arabic orthography and morphology analysis. In *Proceedings of the COLING 2004 Workshop on computational approaches to Arabic script-based languages*, Geneva.

William Cohen. 1995. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123.

Malcolm Corney, Olivier de Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC 2002)*, pages 282–292, Las Vegas.

Kareem Darwish. 2002. Building a shallow Arabic morphological analyzer in one day. In *Proceedings of ACL 2002 Workshop on computational approaches to Semitic languages*, Philadelphia.

Anne de Roeck and Waleed Al-Fares. 2000. A morphologically sensitive clustering algorithm for identifying Arabic roots. In *Proceedings of ACL 2000*, Toulouse.

Olivier de Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2001. Mining email content for author identification forensics. *SIGMOD Record*, 30(4):55–64.

Olivier de Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2002. E-mail authorship attribution for computer forensics. In Daniel Barbara and Sushil Jajodia, editors, *Data Mining for Security Applications*. Kluwer Academic Publishers.

Olivier de Vel. 2000. Mining e-mail authorship. In *Proceedings of the Workshop on Text Mining, ACM International Conference on Knowledge Discovery and Data Mining (KDD'2000)*, Boston.

Dominique Estival, Tanja Gaustad, Son Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for English emails. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING 2007)*, pages ??–??, Melbourne.

Hans Eysenck and Sybil Eysenck. 1975. *Manual of the Eysenck Personality Questionnaire*. Hooder and Stoughton Educational, London.

Leslie Francis, Christopher Lewis, and Hans-Georg Ziebertz. 2006. The short-form revised Eysenenck personality questionnaire (EPQR-S): A German edition. *Social Behaviour and Personality*, 34(2):197–204.

Yoav Freund and Robert Schapire. 1996. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco.

Richard Friedmann. 1997. *Who wrote the Bible?* Harper, San Francisco.

Natalie Glance, Matthew Hurst, Kamal Nigam, Mathew Siegler, Robert Stockton, and Takashi Tomokiyo. 2005. Deriving marketing intelligence from online discussion. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 419–428, Chicago.

Abduelbaset Goweder and Anne De Roeck. 2001. Assessment of a significant Arabic corpus. In *Proceedings of the ACL/EACL Workshop on Arabic Language Processing: Status and Prospects*, Toulouse.

Sathiya Keerthi, Shirish Shevade, C. Bhattacharyya, and K. Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. In Paul B. Kantor et al., editor, *Intelligence and Security Informatics, Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2005)*, pages 209–217. Springer.

Gerrard Ledger and Thomas Merriam. 1994. Shakespeare, Fletcher, and The Two Noble Kinsmen. *Literary and Linguistic Computing*, 9(3):235–248.

F. Mosteller and D.L. Wallace. 1964. *Inference and Disputed Authorship: The Federalist*. Series in behavioral science: Quantitative methods edition. Addison-Wesley.

Warren T. Norman. 1963. Toward an adequate taxonomy of personality attributes: replicated factor structure in peer nomination personality rating. *Journal of Abnormal and Social Psychology*, 66:574–583.

Jon Oberlander and Scott Nowson. 2006. Whose thumb is it anyway? Classifying author personality from weblog text. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 627–634, Sydney.

Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo.

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

Rong Zheng, Yi Qin, Zan Huang, and Hsinchun Chen. 2003. Authorship analysis in cybercrime investigation. In Hsinchun Chen et al., editor, *Proceedings of the first NSF/NIJ Intelligence and Security Informatics Symposium*, pages 59–73. Springer.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *JASIST*, 57(3):378–393.

Proceedings of the Australasian Language Technology Workshop 2007, pages 31-39

Exploring approaches to discriminating among near-synonyms

Mary Gardiner
Centre for Language Technology
Macquarie University
gardiner@ics.mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
madras@ics.mq.edu.au

Abstract

Near-synonyms are words that mean approximately the same thing, and which tend to be assigned to the same leaf in ontologies such as WordNet. However, they can differ from each other subtly in both meaning and usage—consider the pair of near-synonyms *frugal* and *stingy*—and therefore choosing the appropriate near-synonym for a given context is not a trivial problem.

Initial work by Edmonds (1997) suggested that corpus statistics methods would not be particularly effective, and led to subsequent work adopting methods based on specific lexical resources. In earlier work (Gardiner and Dras, 2007) we discussed the hypothesis that some kind of corpus statistics approach may still be effective in some situations, particularly if the near-synonyms differ in sentiment from each other, and we presented some preliminary confirmation of the truth of this hypothesis. This suggests that problems involving this type of near-synonym may be particularly amenable to corpus statistics methods.

In this paper we investigate whether this result extends to a different corpus statistics method and in addition we analyse the results with respect to a possible confounding factor discussed in the previous work: the skewness of the sets of near synonyms. Our results show that the relationship between success in prediction and the nature of the near-synonyms is method dependent and that skewness is a more significant factor.

1 Introduction

Choosing an appropriate word or phrase from among candidate near-synonyms or paraphrases is a significant language generation problem since even though near-synonyms and paraphrases are close in meaning, they differ in connotation and denotation in ways that may be significant to the desired effect of the generation output: for example, word choice can change a sentence from advice to admonishment. Particular applications that have been cited as having a use for modules which make effective word and phrase choices among closely related options are summarisation and rewriting (Barzilay and Lee, 2003). Inkpen and Hirst (2006) extended the generation system HALogen (Langkilde and Knight, 1998; Langkilde, 2000) to include such a module.

We discuss a particular aspect of choice between closely related words and phrases: choice between words when there is any difference in meaning or attitude. Typical examples are *frugal* and *stingy*; *slender* and *skinny*; and *error* and *blunder*.

In this paper, as in Gardiner and Dras (2007), we explore whether corpus statistics methods have promise in discriminating between near-synonyms with attitude differences, particularly compared to near-synonyms that do not differ in attitude. In our work, we used the work of (Edmonds, 1997), the first to attempt to distinguish among near-synonyms, adopting a corpus statistics approach. Based on that work, we found that there was a significant difference in attitudinal versus non-attitudinal near-synonyms. However, the Edmonds algorithm produced on the whole poor results, only a little above the given baseline, if at all. According to (Inkpen, 2007), the poor results were due to the way the algorithm handled data sparseness; she consequently presented an alternative algorithm with much better results. We also found that attitudinal versus non-attitudinal near-synonyms differed significantly in their baselines as a consequence of skewness of synset distribution, complicating analysis.

In this paper then we develop an algorithm based on that of Inkpen, and use a far larger data set and a methodology suited to large data sets, to see whether this alternative method will support our previous findings. In addition we analyse results with regard to a measure of synset skewness. In Section 2 we outline the near-synonym task description; in Section 3 we present our method based on Inkpen; in Section 5 we present out method based on Inkpen, and our experimental method using it; in Section 4 we evaluate its effectiveness in comparison with Inkpen's own method; in Section 5 we test our hypothesis, present our results and discuss them; and in Section 6 we conclude.

2 Task Description

Our experiment tests a system's ability to fill a gap in a sentence from a given set of near-synonyms. This problem was first described by Edmonds (1997). Edmonds describes an experiment that he designed to test whether or not co-occurrence statistics are sufficient to predict which word in a set of near-synonyms fills a *lexical gap*. He gives this example of asking the system to choose which of *error*, *mistake* or *oversight* fits into the gap in this sentence:

(1) However, such a move also of cutting deeply into U.S. economic growth, which is why some economists think it would be a big _______.

Performance on the task is measured by comparing system performance against real word choices: that is, sentences such as example 1 are drawn from real text, a word is removed, and the system is asked to choose between that word and all of its near-synonyms as candidates to fill the gap.

3 An approximation to Inkpen's solution to the near-synonym choice problem

We know of two descriptions of algorithms used to choose between near-synonyms based upon context: that described by Edmonds (1997) and that described by Inkpen (2007).

In our previous work we used Edmonds' method for discriminating between near-synonyms as a basis for comparing whether near-synonyms that differ in attitude in predictability from near-synonyms that do not. The more recent work by Inkpen is a more robust and reliable approach to the same problem, and therefore in this paper we develop a methodology based closely on that of Inkpen, using a different style of training corpus, in order to test whether the differences between the performance of near-synonyms that differ in sentiment and those that do not persists on the better performing method.

Edmonds' and Inkpen's approaches to near-synonym prediction have the same underlying hypothesis: that the choice between near-synonyms can be predicted to an extent from the words immediately surrounding the gap. Returning to example 1, their approaches use words around the gap, eg *big*, to predict which of *error*, *mistake* or *oversight* would be used. They do this using some measure of how often *big*, and other words surrounding the gap, is used in contexts where each of *error*, *mistake* and *oversight* are used. Edmonds uses every word in the sentence containing the gap, whereas Inkpen uses a generally smaller window of words surrounding the gap.

In this section we briefly describe Edmonds' approach to discriminating between near-synonyms in Section 3.1 and describe Inkpen's approach in more detail in Section 3.2. We then describe our adaptation of Inkpen's approach in Section 3.3.

3.1 Edmonds' approach

In Edmonds' approach to the word choice problem, the suitability of any candidate word c for a sentence S can be approximated as a score(c, S) of suitability, and where score(c,S) is a sum of the association between the candidate c and every other word w in the sentence.

(2) $$\text{score}(c, S) = \sum_{w \in S} \text{sig}(c, w)$$

In Edmonds' original method, which we used in Gardiner and Dras (2007), sig(c, w) is computed using either the t-score of c and w or a second degree association: a combination of the t-scores of c with a word w_0 and the same word w_0 with w. Edmonds' t-scores were computed using co-occurrence counts in the 1989 Wall Street Journal, and the performance did not improve greatly over a baseline of choosing the most frequent word in the synset to fill all gaps.

3.2 Inkpen's approach

In Inkpen's method, the suitability of candidate c for a given gap is approximated slightly differently: the entire sentence is not used to measure the suitability of the word. Instead, a certain sized window of k words either side of the gap is used. For example, if $k = 3$, the word missing from the sentence in example 3 is predicted using only the six words shown in example 4.

(3) Visitors to Istanbul often sense a second, _______ layer beneath the city's tangible beauty.

(4) sense a second, _______ layer beneath the

Given a text fragment f consisting of $2k$ words, k words either side of a gap g $(w_1, w_2, \ldots, w_k, g, w_{k+1}, \ldots, w_{2k})$, the suitability $s(c, g)$ of any given candidate word c to fill the gap g is given by:

$$(5) s(c,g) = \sum_{j=1}^{k} \text{PMI}(c, w_j) + \sum_{j=k+1}^{2k} \text{PMI}(w_j, c)$$

PMI(x, y) is the pointwise mutual information score of two words x and y, and is given by (Church and Hanks, 1991):

$$(6) \qquad \text{PMI}(x,y) = \log_2 \frac{C(x,y) \cdot N}{C(x) \cdot C(y)}$$

$C(x)$, $C(y)$ and $C(x, y)$ are estimated using token counts in a corpus: $C(x, y)$ is the number of times that x and y are found together, $C(x)$ is the total number of occurrences of x in the corpus and $C(y)$ the total number of occurrences of y in the corpus. N is the total number of words in the text.

Inkpen estimated $C(x)$, $C(y)$ and $C(x, y)$ by issuing queries to the Waterloo MultiText System (Clarke and Terra, 2003). She defined $C(x, y)$ the number of times where x is followed by y within a certain *query frame* of length q within a corpus, so that, for example, if $q = 3$, example 7 would count as a co-occurrence of *fresh* and *mango*, but example 8 would not:

(7) He likes *fresh* cold *mango*.

(8) I like *fresh* fruits in general, particularly *mango*.

She also experimented with document counts where $C(x)$ is the number of documents that x is found in and $C(x, y)$ is the number of documents in which both x and y are found, called PMI-IR (Turney, 2001); but found that this method did not perform as well, although the difference was not statistically significant.

Inkpen's method outperformed both the baseline and Edmonds' method by 22 and 10 percentage points respectively.

3.3 Our variation of Inkpen's approach

Our variation on Inkpen's approach is designed to estimate PMI(x, y), the pointwise mutual information of words x and y, using the Web 1T 5-gram corpus Version 1 (Brants and Franz, 2006).

Web 1T contains n-gram frequency counts, up to and including 5-grams, as they occur in a trillion words of World Wide Web text. There is no context information beyond the n-gram boundaries. Examples of a 3-gram and a 5-gram and their respective counts from Web 1T are shown in examples 9 and 10:

(9)
```
means official and      41
```

(10)
```
Valley National Park 1948 Art
51
```

These n-gram counts allow us to estimate $C(x, y)$ for a given window width k by summing the Web 1T counts of k-grams in which words x and y occur and x is followed by y.

Counts are computed using a an especially developed version of the Web 1T processing software "Get 1T"[1] originally described in Hawker (2007) and detailed in Hawker et. al (2007). The Get 1T software allows n-gram queries of the form in the following examples, where <⋆> is a wildcard which

[1]Available at `http://get1t.sf.net/`

will match any token in that place in the n-gram. In order to find the number of n-grams with *fresh* and *mango* we need to construct three queries:

(11) `<*> fresh mango`

(12) `fresh <*> mango`

(13) `fresh mango <*>`

However, in order to find *fresh* and *mango* within 4 grams we need multiple wildcards as in example 14, and added the embedded query hashing functionality described in Hawker et. al (2007).

(14) `fresh <*> <*> mango`

Queries are matched case-insensitively, but no stemming takes place, and there is no deeper analysis (such as part of speech matching).

This gives us the following methodology for a given lexical gap g and a window of k words either side of the gap:

1. for every candidate near-synonym c:
    (a) for every word w_i in the set of words proceeding the gap, $w_1, \ldots, w_k$, calculate PMI(w_i, c) as in equation 6, given counts for $C(w_i)$, $C(c)$ and $C(w_i, c)$ from Web 1T[2]
    (b) for every word w_j in the set of words following the gap, $w_{k+1}, \ldots, w_{2k}$, calculate PMI(c, w_j) as in equation 6, given counts for $C(c)$, $C(w_j)$ and $C(c, w_j)$ from Web 1T
    (c) compute the suitability score $s(c, g)$ of candidate c as given by equation 5
2. select the candidate near-synonym with the highest suitability score for the gap where a single such candidate exists
3. where there is no single candidate with a highest suitability score, select the most frequent candidate for the gap (that is, fall back to the baseline described in Section 3.4)[3]

[2]The result of equation 6 is undefined when any of $C(x) = 0$, $C(y) = 0$ or $C(x, y) = 0$ hold, that is, x or y or at least one n-gram containing x and y cannot be found in the Web 1T counts. For the purpose of computing $s(c, g)$, we define PMI$(x, y) = 0$ when $C(x) = 0$, $C(y) = 0$ or $C(x, y) = 0$, so that it has no influence on the score $s(c, g)$ given by equation 5.

[3]Typically, in this case, all candidates have scored 0.

Since Web 1T contains 5-gram counts, we can use query frame sizes from $q = 1$ (words x and y must be adjacent, that is, occur in the 2-gram counts) to $q = 4$.

3.4 Baseline method

The baseline method that our method is compared to uses the most frequent word from a given synset as the chosen candidate for any gap requiring a member of that synset. Frequency is measured using frequency counts of the combined part of speech and word token in the 1989 Wall Street Journal.

4 Effectiveness of the approximation to Inkpen's method

In this section we compare our approximation of Inkpen's method described in Section 3.3 with her method described in Section 3.2. This will allow us to determine whether our approximation is effective enough to allow us to compare attitudinal and non-attitudinal near-synonyms.

4.1 Test sets

In order to compare the two methods, we use five sets of near-synonyms, also used as test sets by both Edmonds and Inkpen:

- the adjectives *difficult*, *hard* and *tough*;
- the nouns *error*, *mistake* and *oversight*;
- the nouns *job*, *task* and *duty*;
- the nouns *responsibility*, *burden*, *obligation* and *commitment*; and
- the nouns *material*, *stuff* and *substance*.

Inkpen compared her method to Edmonds' using these five sets and two more, both sets of verbs, which we have not tested on, as our attitudinal and non-attitudinal data does not included annotated verbs. We are therefore interested in the predictive power of our method compared to Inkpen's and Edmond's on adjectives and nouns.

4.2 Test contexts

We performed this experiment, as Edmonds and Inkpen did, using the 1987 Wall Street Journal as a source of test sentences.[4] Where ever one of the words in a test set is found, it is removed from the context in which it occurs to generate a gap for the algorithm to fill.

So, for example, when sentence 15 is found in the test data, the word *error* is removed from it and the system is asked to predict which of *error*, *mistake* or *oversight* fills the gap at 16:

(15) . . .his adversarys' characterization of that minor sideshow as somehow a colossal *error* on the order of a World War. . ..

(16) a colossal _______on the

4.3 Parameter settings

Recall from Section 3.2 these two parameters used by Inkpen: k and q.

Parameter k is the size of the 'window' of context on either side of a lexical gap in *the test set*: the k words on either side of a gap are used to predict which of the candidate words best fills the gap.

Parameter q is the query size used when querying *the corpus* to find out how often words x and y occur together in order to compute the value of $C(x, y)$. In order to be counted as occurring together, x and y must occur within a window of length at most q.

Inkpen found, using Edmonds' near-synonym set *difficult* and *hard* as a development set, that results are best for a small window ($k \in \{1, 2\}$) but that the query frame had to be somewhat longer to get the best results. Her results were reported using $k = 2$ and $q = 5$, chosen via tuning on the development set.

We have retained the setting $k = 2$ and explored results where $q = 2$ and $q = 4$: due to Web 1T containing 5-grams but no higher order n-grams we cannot measure the frequency of two words occurring together with any more than three intervening words, so $q = 4$ is the highest value q can have.

4.4 Results and Discussion

Table 1 shows the performance of Edmonds' method and Inkpen's method as given in Inkpen (2007)[5] and our modified method on each of the test sets described in Section 4.1. Note that Inkpen reports different baseline results from us—we have not been able to reproduce her baselines. This may be due to choosing different part of speech tags: we simply used `JJ` for adjectives and `NN` for nouns.

Inkpen's improvements for the test synsets given in Section 4.1 were between +3.2% and 30.6%. Our performance is roughly comparable, with improvements as high as 31.2%. Further, we tend to improve especially largely over the baseline where Inkpen also does so: on the two sets *error* etc and *responsibility* etc..

The major anomaly when compared to Inkpen's performance is the set *job*, *task* and *duty*, where our method performs very badly compared to both Edmonds' and Inkpen's methods and the baseline (which perform similarly). We also perform under both methods on *material*, *stuff* and *substance*, although not as dramatically.

Overall, the fact that we tend to improve over Edmonds where Inkpen also does so suggests that our algorithm based on Inkpen's takes advantage of the same aspects as hers to gain improvements over Edmonds, and thus that the method is a good candidate for use in our main experiment.

5 Comparing attitudinal and non-attitudinal synsets

Having determined in Section 4 that our modified version of Inkpen's method performs as a passable approximation of hers, and particularly that where her method improved dramatically over the baseline and Edmonds' method that ours improves likewise, we then tested our central hypothesis: that attitudinal synsets respond better to statistical prediction techniques than non-attitudinal synsets.

5.1 Test set

In order to test our hypothesis, we use synsets divided into near-synonym sets that differ in attitudinal and sets that do not.

This test set is drawn from our set of annotated *attitudinal* and *non-attitudinal* near-synonyms described in Gardiner and Dras (2007). These are WordNet2.0 (Fellbaum, 1998) synsets whose members occur particularly frequently in the 1989 Wall

[4]All references to the Wall Street Journal data used in this paper refer to Charniak et. al (2000).

[5]Inkpen actually gives two methods, one using PMI estimates from document counts, one using PMI estimates using word counts. Here we are discussing her word count method and use those values in our table.

Set	Inkpen's baseline value %	Edmonds' increase over baseline %	Inkpen's increase over baseline ($q = 5$) %	No. test sentences we found	Our baseline value %	Our increase over baseline %	
						$q = 2$	$q = 4$
difficult etc.	41·7	+6·2	+17·4	5959	44·3	+15·3	+12·3
error etc.	30·9	+18·9	+30·6	1026	46·8	+25·5	+20·4
job etc.	70·2	−1·3	+3·2	4020	74·2	−14·4	−23·0
responsibility etc.	38·0	+7·3	+28·0	1119	36·7	+31·2	+24·9
material etc.	59·5	+5·1	+12·7	934	57·8	+5·5	−1·1

Table 1: Performance of Inkpen's test sentences on Edmond's method, Inkpen's method and our method ($k = 2$)

Street Journal. The synsets were annotated as *attitudinal* and *non-attitudinal* by the authors of this paper. Synsets were chosen where both annotators are certain of their label, and where both annotators have the same label. This results in 60 synsets in total: 8 where the annotators agreed that there was definitely an attitude difference between words in the synset, and 52 where the annotators agreed that there were definitely not attitude differences between the words in the synset.

An example of a synset agreed to have attitudinal differences was:

(17) *bad*, *insecure*, *risky*, *high-risk*, *speculative*

An example of synsets agreed to not have attitudinal differences was:

(18) *sphere*, *domain*, *area*, *orbit*, *field*, *arena*

The synsets are not used in their entirety, due to the differences in the number of words in each synset (compare {*violence*, *force*} with two members to {*arduous*, *backbreaking*, *grueling*, *gruelling*, *hard*, *heavy*, *laborious*, *punishing*, *toilsome*} with nine, for example). Instead, a certain number n of words are selected from each synset (where $n \in \{3, 4\}$) based on the frequency count in the 1989 Wall Street Journal corpus. For example *hard*, *heavy*, *gruelling* and *punishing* are the four most frequent words in the {*arduous*, *backbreaking*, *grueling*, *gruelling*, *hard*, *heavy*, *laborious*, *punishing*, *toilsome*} synset, so when $n = 4$ those four words would be selected. When the synset's length is less than or equal to n, for example when $n = 4$ but the synset is {*violence*, *force*}, the entire synset is used.

These test sets are referred to as *top3* (synsets reduced to 3 or less members) and *top4* (synsets reduced to 4 or less members).

5.2 Test contexts

Exactly as in Section 4.2, our lexical gaps and their surrounding contexts are drawn from sentences in the 1987 Wall Street Journal containing one of the words in the test synsets.

5.3 Parameter settings

As described in Sections 3.2 and 4.3, there are two parameters that can be varied regarding the context around a lexical gap (k), and the nearness of two words x and y in the corpus in order for them to be considered to occur together (q).

As per Inkpen's results on her development set, and as in Section 4 we use the setting $k = 2$ and vary q such that $q = 2$ on some test runs and $q = 4$ on others. We cannot test with Inkpen's suggested $q = 5$, as that would require 6-grams.

5.4 Results and Discussion

The overall performance of our method on our sets of attitudinal and non-attitudinal near-synonyms is shown in Table 2.

We did four test runs in total, two each on sets *top3* and *top4* varying q between $q = 2$ and $q = 4$. The baseline result does not depend on q and therefore is the same for both tests of *top3* and of *top4*.

	Contexts containing a test word		Baseline correctness (%)		q	Our method's correctness (%)	
Synsets	Att.	Non-att.	Att.	Non-att.		Att.	Non-att.
top3	45953	353155	59·52	69·71	2	59·51	69·95
					4	56·93	69·96
top4	48515	357290	56·37	68·91	2	52·26	67·60
					4	50·82	67·59

Table 2: Performance of the baseline and our method on all test sentences ($k = 2$)

Improvement over baseline	Number of synsets		
	Att.	Non-att.	Total
$\geq$ +20%	0	16	16
$\geq$ +10% and $<$ +20%	1	7	8
$\geq$ +5% and $<$ +1%	2	2	4
$>$ -5% and $<$ -5%	2	10	12
$\leq$ -5% and $>$ -10%	0	6	6
$\leq$ -10% and $>$ -20%	1	3	4
$\leq$ -20%	1	8	9

Table 3: Distribution of improvements on baseline for *top3*, $k = 2$, $q = 2$

As in our previous paper (Gardiner and Dras, 2007), the baselines behave noticeably differently for attitudinal and non-attitudinal synsets. Calculating the z-statistic as is standard for comparing two proportions (Moore and McCabe, 2003) we find that the difference between the pair of attitudinal and non-attitudinal results for each test are all statistically significant ($p < 0.01$). Thus, again, it is difficult from the data in Table 2 alone to determine whether the better performance of non-attitudinal synsets is due to the higher baseline performance for those same synsets.

There are two major aspects of this result requiring further investigation. The first is that our method performs very similarly to the baseline according to these aggregate numbers, which wasn't anticipated based on the results in Section 4, which showed that on a limited set of synsets our method performed well above the baseline, although not as well as Inkpen's original method.

Secondly, inspection of individual synsets and their performance reveals that this aggregate is not representative of the performance as a whole: it is simply an average of approximately equal numbers of good and bad predictions by our method. Table 3 shows that for one test run (*top3*, $k = 2$, $q = 2$) there were a number of synsets on which our method performed very well with an improvement of more than 20 percentage points over the baseline but also a substantial number where it performed very badly, losing more than 20 percentage points from the baseline.

In our previous work we expressed a suspicion that the success of Edmonds' prediction method might be being influenced by the evenness of distribution of frequencies within a synset. That is, if a synset contains a very dominant member (which will cause the baseline to perform well) then the Edmonds method may perform worse against the baseline than it would for a synset in which the word choices were distributed fairly evenly among the members of the set.

Given the results of the test runs shown in Table 2, and the wide distribution of prediction successes shown in Table 3, we decided to test this hypothesis that the distribution of words in the synsets influence the performance of prediction methods that use context. This is described in Section 5.4.1.

5.4.1 Entropy analysis

In this section, we describe an analysis of the results in Section 5.4 in terms of whether the balance of frequencies among words in the synset contribute to the quality of our prediction result.

In order to measure a correlation between the balance of frequencies of words and the prediction result, we need a measure of 'balance'. In this case we have chosen information entropy (Shannon, 1948), the measure of bits of information required to convey a particular result. The entropy of a synset's frequencies here is measured using the proportion

Test set	q	$Category$	$Entropy$
top3	2	$-0{\cdot}11$	$0{\cdot}41*$
top3	4	$-0{\cdot}10$	$0{\cdot}36*$
top4	2	$-0{\cdot}17*$	$0{\cdot}38*$
top4	4	$-0{\cdot}15*$	$0{\cdot}34*$

Table 4: Regression co-efficients between independent variables synset category and synset entropy, and dependent variable prediction improvement over baseline (statistically significant results $p < 0.05$ marked *)

of total uses of the synset that each particular word represents. A synset in which frequencies are reasonably evenly distributed has high information entropy and a synset in which one or more words are very frequent as a proportion of use of that synset as a whole have low entropy.

We then carried out multiple regression analysis using the category of the synset (attitudinal or not attitudinal, coded as 1 and 0 for this analysis) and the entropy of the synset's members' frequencies as our two independent variables; this allows us to separate out the two effects of synset skewness and attitudinality. Regression co-efficients are shown in Table 4.

Table 4 shows that in general, performance is negatively correlated with both category but positively with entropy, although the correlation with category is not always significant. The positive relationship with entropy confirms our suspicion in Gardiner and Dras (2007) that statistical techniques perform better when the synset does not have a highly dominant member. The negative correlation with category implies that the reverse of our main hypothesis holds: that our statistical method works better for predicting the use of non-attitudinal near-synonyms.

There are two questions that arise from the result that our Inkpen-based method gives a different result from the Edmonds-based one.

First, is our approximation to Inkpen's method inherently faulty or can it be improved in some way? We know from Section 4 that it tends to perform well where her method performs well. An obvious second test is to compare our results to another test described in Inkpen (2007) which used a larger set of near-synonyms and tested the predictive power using the British National Corpus as a source of test contexts. This test will test our system's performance in genres quite different from news-wire text, and allow us to make a further comparison with Inkpen's method.

Second, why do we perform significantly better for near-synonyms without attitude difference? One possible explanation that we intend to explore is that attitude differences are predicted by attitude differences exhibited in a very large context; perhaps an entire document or section thereof. Sentiment analysis techniques may be able to be used to detect the attitude bearing parts of a document and these may serve as more useful features for predicting attitudinal word choice than surrounding words.

6 Conclusion and Future Work

In this paper we have developed a modification to Inkpen's method of making a near-synonym choice that on a set of her test data performs reasonably promisingly; however, when tested on a larger set of near-synonyms on average it does not perform very differently to the baseline. We have also shown that, contrary to our hypothesis that near-synonyms with attitude differences would perform better using statistical methods, on this method the near-synonyms without attitude differences are predicted better when there's a difference in predictive power.

Ultimately, we plan to develop a system that will acquire and predict usage of attitudinal near-synonyms, drawing on statistical methods and methods from sentiment analysis. In order to achieve this we will need a comprehensive understanding of why this method's performance was not adequate for the task.

Acknowledgements

Thank you to: Diana Inkpen for sending us a copy of Inkpen (2007) while it was under review; and Tobias Hawker for providing a copy of his Web 1T processing software, Get 1T, before its public release.

This work has been supported by the Australian Research Council under Discovery Project DP0558852.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003: Main Proceedings*, pages 16–23.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13`.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, , and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2000T43`.

Kenneth Church and Patrick Hanks. 1991. Word association norms and mutual information, lexicography. *Computational Linguistics*, 16(1):22–29.

Charles L. A. Clarke and Egidio L. Terra. 2003. Passage retrieval vs. document retrieval for factoid question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 427–428, Toronto, Canada.

Philip Edmonds. 1997. Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 507–509, July.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, May.

Mary Gardiner and Mark Dras. 2007. Corpus statistics approaches to discriminating among near-synonyms. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING 2007)*, pages 31–39, Melbourne, Australia, September.

Tobias Hawker, Mary Gardiner, and Andrew Bennetts. 2007. Practical queries of a massive n-gram database. In *Proceedings of the Australasian Language Technology Workshop 2007 (ALTW 2007)*, Melbourne, Australia. To appear.

Tobias Hawker. 2007. USYD: WSD and lexical substitution using the Web 1T corpus. In *Proceedings of SemEval-2007: the 4th International Workshop on Semantic Evaluations*, Prague, Czech Republic.

Diana Inkpen and Graeme Hirst. 2006. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32(2):223–262, June.

Diana Inkpen. 2007. A statistical model of near-synonym choice. *ACM Transactions of Speech and Language Processing*, 4(1):1–17, January.

Irene Langkilde and Kevin Knight. 1998. The practical value of N-grams in generation. In *Proceedings of the 9th International Natural Language Generation Workshop*, pages 248–255, Niagra-on-the-Lake, Canada.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics and the 6th Conference on Applied Natural Language Processing (NAACL-ANLP 2000)*, pages 170–177, Seattle, USA.

David S. Moore and George P. McCabe. 2003. *Introduction to the Practice of Statistics*. W. H. Freeman and Company, 4 edition.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October.

Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML 2001)*, pages 491–502, Freiburg and Germany.

Practical Queries of a Massive n-gram Database

Tobias Hawker
School of
Information Technologies
University of Sydney
NSW 2006, Australia
toby@it.usyd.edu.au

Mary Gardiner
Centre for Language Technology
Macquarie University
gardiner@ics.mq.edu.au

Andrew Bennetts
Canonical Ltd.
andrew@puzzling.org

Abstract

Large quantities of data are an increasingly essential resource for many Natural Language Processing techniques. The Web 1T corpus, a massive resource containing n-gram frequencies produced from one trillion words drawn from the World Wide Web, is a relatively new corpus whose size will increase performance on many data-hungry applications. In addition, a fixed resource of this kind reduces reliance on using web results as experimental data, increasing replicability of researchers' results.

However, effectively utilising a resource of this size presents significant challenges. We discuss the challenges of using a data source of this magnitude, and describe strategies for overcoming these, including efficient extraction of queries including wildcards, and specialised data compression. We present a software suite, "Get 1T", implementing these techniques, released as free software for use by the natural language research community, and others.

1 Introduction

The size and quality of data used for statistical Natural Language Processing can have a major impact on the results a system achieves (Banko and Brill, 2001). The World Wide Web gives researchers access to an unprecedented quantity of machine-readable natural language text. However, even for techniques that can take advantage of unannotated text, there is much more web data available than can normally be used, because this same size overwhelms the processing resources available to most researchers. One way of overcoming this practical problem is to use the processing resources of commercial search engines, by using the number of hits they report for frequency estimation — see for example Grefenstette (1999), Turney (2001), Keller and Lapata (2003) and Nakov and Hearst (2005). There are however several drawbacks to this approach, recently highlighted by Kilgarriff (2007), particularly replicability of experiments.

The recent release of the Web 1T corpus (Brants and Franz, 2006)[1] presents an opportunity to access web-scale data for n-gram frequency estimation without the drawbacks of using a search engine. The corpus provides frequency counts for n-grams up to five tokens long, drawn from approximately 1 trillion words of web data. Promising results have already been achieved using this resource for Word Sense Disambiguation and Lexical Substitution (Hawker, 2007; Yuret, 2007) and for Noun-Phrase Bracketing (Vadas and Curran, 2007).

However, the scale of even this distilled collection of web data presents significant processing challenges. Naïve methods for extracting the desired information from the corpus, such as linear search or keyword indexing, are hopelessly impractical, and even algorithms with good asymptotic performance such as binary search are limited in their applicability. Approaches that attempt to overcome the scaling problems by using indices for the set of discrete to-

[1] http://www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt

kens in the corpus are also rendered intractable by the size of the vocabulary. In this paper we present solutions to these difficulties of scale, strategies for practical extraction of the items of interest from this mountain of data.

1.1 Wild Cards

For many NLP applications, it is useful to be able to discover not only the frequency of explicitly specified patterns, but also the frequency of patterns where any token is permissible in certain locations. These 'wildcard' queries allow, for example, the determination of Point-wise Mutual Information in Hawker (2007).

As an example, consider the trigram (from the corpus): *feet seem light*. If we are interested in how likely the word *seem* is to occur in this context, we must not only find frequencies for *feet seem light* but also entries for *feet so light* and *feet the light*. In the notation we use for our queries, we can capture this idea of finding the aggregate frequency for all trigrams with *feet* in the first position and *light* in the third with the query `feet <*> light`. Note that to find the aggregate frequency for *all* matching trigrams, we must employ a search strategy which will indicate a match for *any* suitable string.

In the remainder of this paper, we briefly describe the details of the Web 1T corpus and consider and then rule out possible approaches to using it that are not practical at this scale. We then introduce two practical approaches that allow for the extraction of only the information of interest including one that permits the use of wildcards. Our software tool, "Get 1T", released to the community as free software, has implemented these approaches.

2 Web 1T

The Web 1T corpus (Brants and Franz, 2006) is a collection of n-grams and their frequency of occurrence as found in 1,024,908,267,229 tokens (approximately 1 trillion) comprising 95,119,665,584 sentences of text from publicly accessible web pages. The corpus aims to cover unique pages in English. It is filtered to exclude tokens such as those with problematic encoding, token length, large quantities of accented characters, and unprintable/control characters.

n	distinct n-grams
1	13,588,391
2	314,843,401
3	977,069,902
4	1,313,818,354
5	1,176,470,663

Table 1: Number of distinct patterns

The n-grams counted cover patterns of tokens from unigrams to 5-grams. Tokens with case differences are treated as distinct. The collection is filtered by a cut-off frequency of 200 occurrences for unigrams, and 40 occurrences for bigrams to 5-grams. There are n-grams in the corpus that contain a token with fewer than 200 unigram instances — those tokens within these n-grams are represented with a special token (`<UNK>`). Sentence boundaries are also included as tokens, denoted by `<S>` and `</S>`, thus for example a 5-gram might contain 4 words and a sentence boundary.

The number of unique patterns arising from this collection is itself very large, as shown in Table 1.

With the patterns and counts stored in text format, one per line, and then compressed, the frequency collection occupies around 25GB of disk space, and is distributed by the LDC on 6 DVDs.

2.1 Comparison with Search Engine Hits

Recently, Kilgarriff (2007) has raised issues surrounding the use of search engine counts for determining relative frequencies of various instances of natural language. Owing to the commercially sensitive and competitive nature of search engine strategies, the process behind the counts reported by search engines is not generally known, and the data on the web changes over time; thus the counts reported are potentially unreliable. Changes may occur in the algorithm, query syntax, or even using the same search engine for identical queries on consecutive days — Kilgarriff (2007) found the results from 6 in 30 queries differed by at least a factor of two on consecutive days, and speculates that this may be due to queries being serviced by different computers at different parts of the update cycle. This results in experiments that may be impossible to replicate, even by one researcher over time, let alone by others.

Aside from replicability concerns, This may account for some findings that search results are less reliable than corpus counts in, for example, context-sensitive spelling correction (Liu and Curran, 2006).

By contrast, the approach for constructing the Web 1T corpus has been disclosed (Brants and Franz, 2006), and is intentionally aimed at yielding accurate n-gram models, rather than estimating the popularity of query keywords for a web search designed to locate pages. Since Web 1T is distributed to researchers, there are no limits on the number of queries that can be performed in a given time frame aside from any imposed by resource limitations of the researcher. The values for different queries are directly comparable, as they cover the same corpus. Importantly, experiments are entirely repeatable, as the results of queries using a specific release of the corpus do not change.

The n-gram database approach is not without a few drawbacks however. Snippets giving broader context than the maximum 5-token n-gram size are not available. The original source of a document, useful for such things as topic determination, is lost. Parsing such short fragments is, in general, not likely to be reliable. POS tagging is possible on the longer n-gram sequences, though is likely to be somewhat less reliable than typical POS tagging performance on full sentences.

Another serious drawback is that the resources required for performing queries are not provided on massively redundant hardware at essentially zero cost, as is the case for commercial search engines. This requires researchers to query the data on their own hardware using software that is able to handle this quantity of data efficiently.

This paper aims to detail approaches that permit practical use of the Web 1T corpus. We have released free software tools implementing these approaches to the community[2] under the GNU GPL.

3 Infeasible Approaches to Processing

In the following discussion, the question of accessing the data will be framed in terms of *queries* — particular n-gram patterns or abstractions thereof, such as `feet so light` or `feet <*> light`, whose count is desired from the corpus, much as would be submitted to a search engine API.

[2]Available to the public at `http://get1t.sf.net/`

In working with a corpus at this scale, it is undesirable access the disk to retrieve data that is not useful for the application at hand, and very problematic to consider the same data multiple times. Any strategy for querying the corpus must thus attempt to avoid reading unneeded data, and where it must, make as few passes over the data as possible.

3.1 Direct Queries

One approach to extracting queries of interest might be to take advantage of the fact that the n-gram patterns are not listed arbitrarily, but are sorted into a lexicographically ascending order. This enables the use of a binary search to find a query without having to consider most entries in the database. As binary search complexity increases only with the logarithm of the number of database entries, it seems reasonable that this method might scale better to large quantities of data than any linear approach.

The Web 1T corpus data is split into a single directory for each n-gram length (1–5 tokens). The patterns are sorted in lexicographically ascending order, and split into files of at most 10,000,000 entries, individually compressed. An index file specifies the first n-gram in each file, which can also be used as a guide to the endpoint of the previous file.

Binary search for a small number of queries is tractable without the prohibitive requirement of storing all n-grams in RAM. The index files fit easily in RAM however, and finding which n-gram file to search depends on the number of unique patterns for that length (see Table 1). For 4-grams (the worst case), this leads to an average of about $\log_2 132$, or 7 iterations per query.

There are 10^7 n-grams in each compressed file, which leads to an average of $\log_2 10^7$, or about 23, tests of location per exact query. Many parts of the file must be retrieved to find each element. Each of these accesses requires computational effort (which in many cases will require a disk seek) and as the files do not have fixed record size, there is an additional burden of finding line-breaks before the actual n-gram record can be determined for comparison against the desired query. The string comparisons required are also computationally expensive.

For even a moderate number of queries (more than a few hundred), reads from all over the disk to retrieve each query are very inefficient. It is impractical to store the entire corpus in RAM, at least for commodity resources typically available. The corpus is of the order of 80GB uncompressed, which is well beyond RAM resources typically available to researchers at the time of writing.

Furthermore, case-insensitive queries are very problematic, as case differences may occur at any point in the string. In ASCII, the two cases of a letter, upper and lower, are lexicographically distant (values differ by 32), and thus otherwise identical queries may be separated by many entries or be in entirely different files. One potential solution is to re-generate the corpus for case-insensitive use. This is possible, and need only be performed once, but increases the storage burden, and while sorting data of this size is possible using existing techniques (for example, mergesort) it would not be a small task for typical resources by any means.

However, the most difficult drawback to overcome with binary searching is the desirability of performing queries with wildcards — queries whose strings are not specified exactly, but may match any tokens at some locations. Unless the wild component is near the end of the pattern (which permits the use of binary searches to select a subspace of the original, which can then be exhaustively searched) binary searches are completely impractical.

3.2 Indexing

Another possible approach to extracting queries from the data might be the use of a mechanism that takes advantage of the fact that the n-gram patterns of length 2–5 are comprised of tokens from the finite set of unigrams. Unfortunately, this approach too is very difficult for data on the scale of the Web 1T data. There are more than 13 million unigrams in the corpus, and assigning a unique binary value to each one would thus require 24 bits per token. Attempting to index the corpus using indices of 24 bits and only a single byte at each indexed location would require in excess of 100 TB for bigrams, with almost all locations empty.

We measured the entropy per token in the unigram frequencies, and determined there were 10.7 bits per token for case-sensitive measurement, and 10.3 bits per token for the case-insensitive case. However, while entropy coding allows the *average* number of bits per token to be brought closer to these ideal representations, some n-grams with combinations of infrequent tokens would have representations that far exceed this average size.

4 Practical Approaches to Processing

This section describes methods that render querying the Web 1T corpus practical by relying on the insight that the corpus should be read from disk as few times as possible — preferably once. Thus, our strategies require that pre-processing occur before the frequencies for the desired queries are extracted from the corpus. One of two pre-processing steps is required. A researcher might use our tool to pre-process the corpus, permitting quick, on-the-fly queries, at the price of only having approximate counts, and of false positive counts for some queries not present in the corpus. Or he or she might be able to specify queries in advance, in case we can compute exact counts in a single pass through the corpus.

4.1 Pre-processing the Corpus

This strategy makes queries of the corpus practical by compressing the massive quantity of data to a manageable level. This is achieved in two ways: by reducing the resolution of the frequency information and then by replacing the n-gram strings themselves with implicit information based on the location in the compressed data. Depending on the task, many statistical approaches can handle some noise in the data; although fidelity may be somewhat reduced, the impact is not all negative — it may also smooth.

We thus construct a single monolithic binary file for each length of n-gram. Instead of storing the exact string of the n-gram alongside its count, as in the original data, we stored the quantised set of frequencies at *locations* determined by a hash value of the n-grams concerned.

For many applications, the absolute resolution of the frequency information used is not as important as the magnitude. In particular, in many statistical approaches the absolute difference between two quantities will not be as informative as the ratio: the difference between a pattern occurring 500,000 and 500,050 times is much less significant that the difference between patterns occurring 50 and 100 times. It is thus desirable to have the resolution of the frequency approximately constant with respect to magnitude, thus preserving the dynamic range of all frequencies. A suitable transformation is to store a value corresponding to the logarithm of the frequency. If this value is then quantised, a great deal of compression is possible. For implementation on practical hardware, an integer number of bytes is preferable. Using a single byte for each frequency allows 255 discrete magnitude levels to be encoded.

The frequencies of patterns in the data have a very large range, from 40 to 95,119,665,584. The quantisation is performed with a logarithm base determined by the maximum count. A zero byte represents an unseen n-gram (frequency of zero) and the remaining 255 quantisation levels are selected such that their span is uniform in the logarithmic space. The use of this logarithmic scale means that error introduced by the quantisation is roughly proportional to the magnitude of the frequency being quantised. Using zero bytes for unseen n-grams simply requires the initialisation of all locations to zero before beginning the compression process.

To permit optimal use of the relatively small quantised representation, the quantisation ranges were determined separately for each value of n, ensuring the maximum count for that number of tokens was transformed to the maximum integer value. As the quantised values are converted back to approximate counts in linear space when being reported, this scaling ensures acceptable precision for each quantisation scaling while still yielding comparable frequencies, even between different n-gram types.

When translating quantised values back to approximate frequencies, such as when forming features from the frequencies, all instances of a given quantised value are interpreted as the geometric mean of the boundaries of the range of integers that fall within it. This is also the arithmetic mean of the logarithms concerned.

There is also a choice to be made when a collision is found during compression — when an n-gram hashes to a value that is already non-zero in the compressed file. At this point there are several options: the existing value can be combined with the new value, such as being added together; a choice can be made between the two values, such as the smaller or larger, or the existing or new value; or a compromise value, such as the geometric mean of the two values, can be stored. Which mechanism is suitable depends on the downstream task being performed. The implementation allows this choice to be made by the user at the time the data is compressed.

For a given file size, there is a trade-off between the resolution of the frequency information and the number of unique keys, which in turn influences the probability of collisions. For example, counts can be made finer resolution by using two bytes per file location. This is at the expense of the hash function, which will thus have one fewer bits of key space for data files of the same size.

4.1.1 Retrieval

To perform a query, the desired search string is hashed, the quantised value retrieved at the location indicated by the hash, and transformed back into a linear count. Practical hash sizes are around 30–32 bit hashes, which yield indices of 1–4GB. The size of the hash is constrained by the fact that it must fit in RAM for building the hash to be tractable. Uniform hashing necessarily involves a pseudo-random sequence of locations. This implies that if RAM were insufficient for the chosen hash size, disk seeks would be frequent, and impact very strongly on performance. It would be possible to build parts of larger hash files in RAM, using several passes over the data, but this requires 2^n passes for each additional n bits of hash, which would also quickly becomes prohibitive.

4.1.2 Properties

This strategy cannot yield a false negative result for a query — a zero count when the queried pattern does in fact occur. However, false positives are possible due to hash collisions and can become problematic. The collisions are due to there being an infinite number of possible patterns that would map to each location, yet the hash value being finite.

This approach is thus not suited to wildcards at all. Uniform hashing functions with finite hash values are one-way, and deliberately yield different hash values for inputs that differ only slightly. Thus the set of possible matches must be generated as inputs to the hashing function, and so each hypothetical match must be generated and treated as a query. The number of possible matches for any under-specified sequence is enormous, and even if only a very small fraction of these queries results in false positives, the sheer number of queries required quickly renders the combined effect insurmountable.

4.1.3 Implementation

This software compresses each n-gram pattern length into a single file whose size in bytes is a power of two. The files may be read into RAM and queried in a random-access fashion, or counts can be retrieved directly from the disk.

The uniform distribution property of the hash function, desirable in minimising the number of hash collisions becomes problematic if accessing the compressed data directly from disk. If the n-gram frequencies are retrieved in the order they are used by the system, the hashing transformation renders the requests into a sequence of pseudo-random disk accesses, leading the disk heads to skip back and forth repeatedly. One possible solution is to map the file into memory for access. However, for a finite set of queries this too is sub-optimal, as many blocks of data must be read from disk that will not be used.

If however, the queries are collated, hashed, and sorted based upon their hash value, the retrieval software never needs to re-read or seek backwards. Even more advantageously, blocks not containing patterns of interest do not have to be read at all. For even a large number of queries this process is very rapid.

Python bindings have also been created for accessing files created using this approach. Disk access times are generally greater than the CPU work for retrieval, and the hashing itself is performed in a native C extension, so the performance penalty involved using an interpreted language is minor.

4.2 Pre-processing Queries

The approach described in this section does not include any approximations or false-positive counts, and allows queries that will match any token in specified positions. The implementation of this approach has been used to perform successful experiments in Word Sense Disambiguation and Lexical Substitution (Hawker, 2007). The price for these desirable properties is that all queries must be collated before the corpus information is extracted; on-the-fly queries are not possible.

The key idea in our approach is the reversal of the target of the search — from the database to the queries themselves. To adapt a well-known metaphor, we can view each query as a needle of a particular size and shape, and the Web 1T corpus as an enormous haystack, possibly containing a piece of hay that resembles the size and shape of the needle (i.e. an entry matching the query). In this metaphor, taking each desired needle individually and trying to find matches in the enormous haystack is clearly an insurmountable task for any large-scale haystack. Our method involves collecting all needles in which we are interested before any search is performed, and then considering each strand in the haystack in turn, finding whether it matches any of the needles. This is still an intensive task, but it is certainly tractable, as each piece of hay need be considered only once.

4.2.1 Search Strategy

Queries are provided to the program as input when the search is launched. Queries are stored in a nested hashtable, where the key to the hash table at any given level i is the i-th word in the query. For example, the query `frozen hell buckets` will be indexed under *frozen* at the first level, *hell* at the second with other queries beginning with `frozen` and under *buckets* at the third level with other queries beginning with `frozen hell`. The final mapping is to the counter representing the number of matches for this query. Each level contains a mapping to the next token. The algorithm for constructing these structures is detailed in Figure 1.

During query construction, wildcard placeholders are treated identically to other tokens. They are represented in our implementation by the token <*>, which is not present in the unigram counts for the Web 1T corpus. The counter is used in hashtables at the final token to store the frequencies found when patterns do match.

After all queries have been processed into these structures, each pattern of the appropriate length in the corpus is checked to determine if any matches occur. The recursive procedure used to check for matches is given in Figure 2. Initial arguments are $h = h_0$ and a depth = 1. The set of tokens and the frequency of those tokens is consistent across all invocations of the Search procedure for a given Web 1T entry, and is thus omitted from the arguments shown for brevity. The two recursive invocations are

```
Let $h_1$ be initial hashtable;
foreach query do
    Let $h = h_1$;
    foreach $i \in 1 \ldots n$ do
        token$_i$ = query[$i$];
        if h[token$_i$] does not exist then
            if $i \neq n$ then
                set h[token$_i$] = new hashtable;
            else
                set h[token$_i$] = counter
                initialised to 0;
            end
        end
        $h = h$[token$_i$]
    end
end
```

Figure 1: Query Structure Construction

```
Procedure Search(h, depth) begin
    token$_i$ = query[depth];
    if h[tokens$_i$] exists then
        if depth $\neq n$ then
            Search(h[tokens$_i$], depth + 1)
        else
            counter = h[tokens$_i$];
            counter $\leftarrow$ counter + frequency;
        end
    end
    if h[WILDCARD] exists then
        if depth $\neq n$ then
            Search(h[WILDCARD], depth + 1)
        else
            counter = h[WILDCARD];
            counter $\leftarrow$ counter + frequency;
        end
    end
end
```

Figure 2: **Procedure** Search(`h, depth`)

to match the current token with both any query specifying that particular token at that location, and any query specifying a wildcard in that position.

4.2.2 Algorithmic Complexity

The time taken to add queries is maximised when each token in every query is novel, and thus is not present in the existing structure. In this case the time for each query is proportional to the number of tokens. This means that adding q queries of length n is $O(nq)$.

Considering our search strategy for a single corpus entry, the largest number of hashtable lookups are required when queries exist for all combinations of both wildcards and exactly matching tokens. This worst-case instance requires two hashtable lookups for the first token, four lookups for the second and so on: a total of $2^{n+1} - 2$ lookups for patterns of length n. In the asymptotic limit, the algorithmic complexity for each corpus entry is thus $O(2^n)$. If we let m represent the number of entries of length n in the corpus, the independent processing of each entry yields an overall complexity of $O(2^n m)$.

In practice however, there are several factors that combine to keep search performance fast. Firstly, n has a maximum value of 5, leading to only 62 hashtable lookups in the worst case. The actual performance of the system depends on the characteristics of the data and the queries. For the exponential behaviour to apply to large numbers of entries in the database, each of these distinct entries would have to be matched by a corresponding exact query, accompanied by a large number of wildcard permutations. As there are far fewer queries than database entries, the worst-case performance can not apply to more than a small fraction of the database entries. In the best case, very few entries match queries at all — comparison for most patterns can then be terminated at the first token, with a complexity of $O(m)$.

Even in the worst case for hashtable lookups, the algorithmic complexity does not contain a term for the number of queries. While searches over more queries and an abundance of wildcards will impact the running time to some extent, even with many millions of queries the overall performance has a practical upper bound. As Keller and Lapata (2003) observe, a single linguistic query may expand to many corpus queries to account for inflectional and other variation, and thus continued efficiency as the number of queries is increased is thus most desirable. This is not the case for binary search or any of the other methods discussed previously.

Memory requirements are related only to the number of, and similarities among the queries specified. The memory required for increasing the number of queries will be no worse than linearly proportional to the size of the query set. This is apparent when the case where each term in each query is unique is considered: each term will be stored exactly once in the data structure. Any overlaps in the leftmost tokens of queries will reduce the memory burden with respect to this worst-case requirement.

4.2.3 Implementation

This software runs once through the n-grams for each pattern length, and extracts counts for all specified queries. The input format for the queries matches that of the corpus exactly — one query per line — aside from the lack of the final tab character and numeric count. The queries are processed into the nested hashtable structures in RAM before any access is made to the corpus data. After construction of these structures, all Web 1T corpus files containing entries for patterns of the appropriate pattern length (each containing 10^7 counts) are processed in turn. Each is decompressed into RAM and once there each entry is checked against the query structures as described previously.

Following processing of all relevant corpus files, the results are written to text files. Case-sensitivity is configurable at run-time via command-line options. Counts are stored in RAM rather than written immediately, as for case-insensitivity and queries involving wildcard counts, the frequency reported may be the sum of many individual counts in the corpus.

It is also possible to find not just the total number of n-grams for a given wildcard, but also enumerate each match; this may also be configured via the command line at run time, but for 2 and 3-gram queries it is not recommended, as it generally results in a prohibitively large number of hits.

5 Performance

Tools which employ both strategies described earlier have been implemented, and have been made available as free software. For performance reasons, C was used as the implementation language.

The speed of the approach using the pre-processed (compressed) corpus depends straightforwardly on the number of queries, and is generally constrained by the time to read data from disk.

The pre-processed query approach was implemented with adequate performance on commodity hardware as a design goal. As a result, the runtime speed is quick, and scales well with large numbers of queries. Experiments have been performed that search for over 1,000,000 5-grams with acceptable performance. For typical queries, the sparseness of hits among 5-grams, even in a resource spanning as much variety as the Web 1T corpus ensures that the worst-case performance is infrequent.

Processing all 5-grams for 10^6 queries on a computer with a 2.66GHz Xeon CPU took around 1 hour and 1.5GB of RAM. Simple parallelisation is easily achieved by processing patterns of different lengths on separate computers or CPU cores (assuming of course that sufficient RAM and CPUs are available). This approach can reduce the time taken for the entire run to the time for that of the longest-running pattern length. The memory footprint of the program is at worst linearly related to the number of queries, and as these tests show, is manageable for a large number of queries.

6 Conclusion

It is possible to query a resource the size of the Web 1T corpus using commodity hardware and effective hashing-based strategies. Software has been created that makes the use of this resource practical, and has been successfully used to harness the information available for NLP tasks including Word Sense Disambiguation, Lexical Substitution and Noun-Phrase Bracketing. The methods employed need only to make a single pass of the corpus data. In one approach, the corpus is transformed to a more amenable structure; in the other, the queries are indexed and searched, rather than the corpus.

The tools we have implemented, using the the techniques described in this paper, facilitate the use of the massive scale of data now available for more disparate and data-hungry NLP tasks.

7 Acknowledgements

Many thanks to James Curran and Jon Patrick for their assistance and advice. We also thank Charles B. Falconer, author of the free hashlib library for C, which we use for the pre-processed queries. Part of this work has been supported by the Australian Research Council under Discovery Project DP0558852.

References

Michelle Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics (ACL-01)*, pages 26–33. Toulouse, France.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1. Technical report, Google Research.

Gregory Grefenstette. 1999. The WWW as a resource for example-based MT tasks. In *ASLIB Translating and the Computer Conference*. London.

Tobias Hawker. 2007. USYD: WSD and lexical substitution using the Web 1T corpus. In *Proceedings of the Fourth International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SemEval-07)*, pages 446–453. Prague, Czech Republic.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Adam Kilgarriff. 2007. Googleology is bad science. *Computational Linguistics*, 33(1):147–151.

Vinci Liu and James R. Curran. 2006. Web text corpus for natural language processing. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 233–240.

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 17–24. Ann Arbor, Michigan.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001), Freiburg, Germany*, pages 491–502.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics (ACL)*, pages 240–247. Prague, Czech Republic.

Deniz Yuret. 2007. KU: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SemEval-07)*, pages 207–214. Prague, Czech Republic.

Extending Sense Collocations in Interpreting Noun Compounds

Su Nam Kim,♠♡ **Meladel Mistica**♠ and **Timothy Baldwin**♠♡
♠ CSSE
and
♡ NICTA VRL
University of Melbourne
VIC 3010, Australia
{snkim,mmistica,tim}@csse.unimelb.edu.au

Abstract

This paper investigates the task of noun compound interpretation, building on the sense collocation approach proposed by Moldovan et al. (2004). Our primary task is to evaluate the impact of similar words on the sense collocation method, and decrease the sensitivity of the classifiers by expanding the range of sense collocations via different semantic relations. Our method combines hypernyms, hyponyms and sister words of the component nouns, based on WORDNET. The data used in our experiments was taken from the nominal pair interpretation task of SEMEVAL-2007 (4th International Workshop on Semantic Evaluation 2007). In our evaluation, we test 7-way and 2-way class data, and show that the inclusion of hypernyms improves the performance of the sense collocation method, while the inclusion of hyponym and sister word information leads to a deterioration in performance.

1 Introduction

This paper investigates the automatic interpretation of noun compounds (i.e. NCs), such as *paper submission* and *computer science department*. NC interpretation is a well-known problem that aims to predict the **semantic relation** (i.e. SR) between a head noun and its modifying nominal(s) (i.e. modifiers). SRs, simply put, encapsulate how the head noun and the other nominals in a noun compound are related. As English noun phrases are right-headed, the head noun occurs after all modifying nouns. For example, *brick house* is interpreted as a *house* that is modified by the word *brick*, which exhibits a PRODUCT-PRODUCER relationship between the two nouns in the compound. In contrast, the modifier and head in *house brick* exhibits a PART-WHOLE relationship, which is interpreted as a brick from a house, rather than the former interpretation of a house made of bricks. The set of SRs that we are concerned with in this paper is defined in Section 5.1.

Research on NCs can be categorised into four main groups: defining SRs, disambiguating the syntax of NCs, disambiguating the semantics of NCs, and interpreting NCs via SRs. Each task is detailed in Section 2.1. Interpreting NCs has received much attention of late, and the problem has been addressed in areas of machine translation (MT), information extraction (IE), and applications such as question-answering (QA). NCs pose a considerable challenge to computational linguistics due to the following issues (Lapata, 2002): (1) NCs are extremely productive; (2) the semantic relationship between the head noun and its modifier(s) is implicit; and (3) the interpretation of an NC can vary due to contextual and pragmatic factors. Due to these challenges, current NC interpretation methods are too error-prone to be employed directly in NLP applications without human intervention or preprocessing.

In this paper, we investigate the task of NC interpretation based on sense collocation. It has been shown that NCs with semantically similar components share the same SR (Kim and Baldwin, 2007); this is encapsulated by the phrase **sense collocation** in Moldovan et al. (2004). For example, *ap-*

ple pie has the same interpretation as *banana cake* of PRODUCT-PRODUCER. This can be predicted by the fact that the modifiers of both NCs (*apple* and *banana*, respectively) are semantically similar (they are both fruit), and the head nouns of both NCs (*pie* and *cake*, respectively) are a type of baked edible concoction. That the two NCs are based on the same combination of semantic classes is a strong predictor of the fact that they have the same SR.

One obvious problem when proceduralising semantic collocation in an interpretation method is data sparseness, i.e. we can't expect to find instances for all combinations of semantic classes, particularly if we have a rich inventory of semantic classes such as WORDNET. One approach to ameliorating the data spareness is bootstrapping, in the manner of Kim and Baldwin (2007), where new data is induced by substituting the components of the NCs with semantically similar terms. Our approach in this paper is to add related terms as features into a classifier. The related terms we add are the NC components' hypernyms, hyponyms, and sister words, based on the hypothesis that these related words can contribute to the disambiguation of SRs.

The remainder of this paper is structured as follows. In Section 2 we introduce previous research on NC interpretation and then talk specifically about the research directly relevant to this work. Section 3 and Section 4 describe our motivation and method, respectively. We describe the data used in our experiments in Section 5 and present the results of our experiments in Sections 6 and 7. Finally, we conclude our work in Section 8.

2 Related Work

This section presents a short description of the computational tasks relating to NCs, and then reviews research directly impinging on this research.

2.1 Background

The major tasks related to NCs involve syntactic and semantic disambiguation.

The first step in semantic disambiguation is the task of defining what relations exist in NCs. This has gained much attention in recent decades, as well as controversy, (Downing, 1977; Levi, 1979; Finin, 1980). In the study conducted by Levi (1979), it was claimed that there were 9 distinct SRs, which could be discretely defined and interpreted within NCs, while Finin (1980) claimed an unlimited number of SRs. The problems surrounding this task involve the issue of granularity versus coverage, which to date remains widely debated.

Syntactic disambiguation (called bracketing) is required when NCs are composed of more than 2 components, such as in the case of *computer science department*, introducing the need for phrasal disambiguation (Lauer, 1995; Nakov, 2005). Lauer (1995) proposed probabilistic models (based on dependency and adjacency analyses of the data). Later Nakov (2005) built upon this by adding linguistic features into these probabilistic models.

Methods employed in word sense disambiguation (WSD) have also been used to enhance NC interpretation; the noun components that comprise the NCs are disambiguated using these WSD techniques (Sparck Jones, 1983; Kim and Baldwin, 2007). Kim and Baldwin (2007) carried out experiments on automatically modeling WSD and attested the usefulness of conducting word sense analysis of an NC in determining its SR.

2.2 Previous Approaches to NC Interpretation

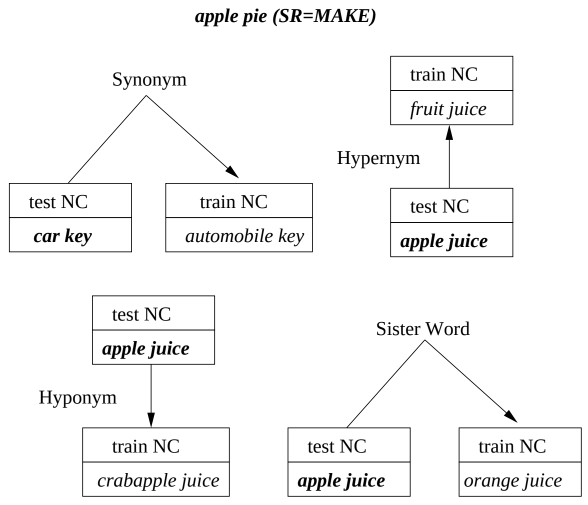
A majority of research undertaken in interpreting NCs has been based on two statistical methods: SEMANTIC SIMILARITY (Barker and Szpakowicz, 1998; Rosario, 2001; Moldovan et al., 2004; Kim and Baldwin, 2005; Nastase, 2006; Girju, 2007; Kim and Baldwin, 2007) and SEMANTIC INTERPRETABILITY (Vanderwende, 1994; Lapata, 2002; Kim and Baldwin, 2006; Nakov, 2006). Our work, based on an extension of the sense collocation approach, corresponds to the semantic similarity method.

A significant contribution to this area is by Moldovan et al. (2004), who used the sense collocation (i.e. pair of word senses) as their primary feature in disambiguating NCs. Many subsequent studies have been based on this sense collocation method, with the addition of other performance-improving features. For example, Girju (2007) added contextual information (e.g. the grammatical role and POS) and cross-lingual information from 5 European languages as features to her model. In contrast, Kim and Baldwin (2007) utilise sense collocations in a different way: instead of adding additional features in their model, they increase the size of their training data by substituting components of existing training instances to generate additional training instances (which is assumed to have the same SR as the original). For an SR to be preserved, the newly-generated NC must be semantically similar and hence maintain the same sense collocation as the original NC on which it was based. A number of researchers (Rosario (2001), Kim and Baldwin (2005), Nastase (2006), inter alia) have attempted to interpret NCs via implicit sense collocations. In particular, they have come up with various methods for avoiding direct WSD. Rosario (2001) retrieved the sense pairs in the context of a hierarchical class set for the biomedical domain. Kim and Baldwin (2005) used a word-level similarity measure to express the sense collocation of NCs. Nastase (2006) listed the hypernyms of components as sense features.

3 Motivation

As mentioned above, Moldovan et al. (2004) showed that the sense collocation of NCs is a key feature when interpreting NCs. Further research in this area has shown that not only synonymous NCs share the same SR, but NCs whose components are replaced with more loosely related words also commonly have the same SR as the original NCs (Kim and Baldwin, 2007). For example, *car factory*, *vehicle factory* and *truck factory*—corresponding to synonym, hypernym and sister word substitutions, respectively, over *automobile factory*—share the same SR of PRODUCT-PRODUCER as the source NC.

Figure 3 shows an example of semantic neighbours for the two NCs *car key* and *apple pie*. *Car key* can be interpreted as PRODUCT-PRODUCER by referring to the training NC *automobile key*, since they have the same sense collocation. With *apple juice*, the sense collocation method tries to locate matching sense collocations in the training data, and finds that *fruit juice* matches closely, with the modifier being a hypernym of *apple*. From this, we can hope to correctly interpret *apple juice* as having the SR PRODUCT-PRODUCER. In order to achieve this, we require some means of comparing nouns taxonomically, both vertically to capture hypernyms and hyponyms, and horizontally to capture sister words.



As intimated above, our motivation in conducting this research is to be able to include hypernym, hyponym and sister word information without using direct substitution over the training instances, but still preserving the essence of the sense collocation approach. The disadvantage of the method employed by Kim and Baldwin (2007) of recursively bootstrapping off a seed set of NCs via different lexical relations, is that noise will inevitably infect the training data, skewing the classifier performance. The original method described in Moldovan et al. (2004) only relies on observed sense collocations. The components of the NCs are represented as specific synsets in WORDNET, and the model does not capture related words. Hence, in this paper, we aim to develop a model that can take advantage of relatedness between WORDNET synsets via hypernyms, hyponyms and sister words, without the risk of losing semantic granularity or nintroducing noisy training data. Note that in Kim and Baldwin (2007), we used synonyms, hypernyms and sister words. As synonyms have an identical sense collocation within WORDNET (i.e. pairing of synsets) to the original NC, they are ignored in this research. Instead, we add hyponyms as a means of broadening the range of sense collocation.

4 Method

First, we describe the principal idea of the sense collocation approach to NC interpretation and the probability model proposed in Moldovan et al. (2004). Then we present our method using hypernyms, hyponyms and sister words in order to extend the sense collocation method.

4.1 Sense Collocation

The basic idea behind sense collocation in Moldovan et al. (2004) is based on the 'pair-of-word-senses' from the component nouns in NCs. They also introduced a probability model called **semantic scattering**, as detailed in Equations 1 and 2 below. In essence, the probability $P(r|f_i f_j)$ (simplified to $P(r|f_{ij})$) of a modifier and head noun with word sense f_i and f_j, respectively, occurring with SR r is calculated based on simple maximum likelihood estimation:

$$P(r|f_{ij}) = \frac{n(r, f_{ij})}{n(f_{ij})} \quad (1)$$

The preferred SR r^* for the given sense combination is that which maximizes the probability:

$$\begin{aligned} r^* &= \text{argmax}_{r \in R} P(r|f_{ij}) \\ &= \text{argmax}_{r \in R} P(f_{ij}|r) P(r) \quad (2) \end{aligned}$$

4.2 Adding Similar Words

We extend the approach of Moldovan et al. (2004) by adding similar words as features focusing on hypernyms, hyponyms and sister words of the modifier and head noun.

We accumulate the features for semantic relations based on different taxonomic relation types, from which we construct a feature vector to build a classifier over. The features of each taxonomic relation type are listed below. The first is features used in the original sense collocation method. The second, third and fourth are our experimental features, based on hypernyms, hyponyms and sister words respectively.

1. $< WS_{mod}, WS_{head} >$
2. $< WS_{mod}, H^i_{mod}, WS_{head}, H^i_{head} >$
3. $< WS_{mod}, O_{mod}, WS_{head}, O_{head} >$
4. $< WS_{mod}, S_{mod}, WS_{head}, S_{head} >$

where mod is the modifier, $head$ is the head noun, WS_{mod} is the WORDNET synset of the modifier, WS_{head} is the WORDNET synset of the head, H^i is an ith-degree ancestor (with direct hypernyms corresponding to H^1), O is a hyponym and S is a sister word. We include up to the 7th-degree ancestor (i.e. H^7), in line with the findings of Nastase (2006). Note that while a given synset has a unique hypernym in WORDNET (assuming no cycles, or the ability to remove cycles by precompiling a tree structure), it can have arbitrarily many hyponyms and sister words. Here, we take the cross product of the different hyponym and sister word candidates for a given synset.

We build our final classifier with TIMBL V6.0, a memory-based learner (Daelemans et al., 2004).

5 Data

Below, we outline the data used in our experiments.

5.1 Semantic Relation

The SR between a head and its modifier(s) in a NC tells us how to (default) interpret the NC. For example *door knob* corresponds to the PART-WHOLE relation, which means we can interpret *knob* as being part of a *door*. We sidestep the considerable challenge of developing an optimal set of semantic relation categories by using the set of SRs and data from the SEMEVAL-2007 nominal pair interpretation task (Girju et al., 2007). The SRs defined for the task are: CAUSE-EFFECT (CE), CONTENT-CONTAINER (CC), INSTRUMENT-AGENCY (IA), ORIGIN-ENTITY (OE), PART-WHOLE (PW), PRODUCT-PRODUCER (PP) and THEME-TOOL (TT). Table 1 provides a definition of each SR along with example NCs.

5.2 Data Collection

From the SEMEVAL-2007 annotated data (Girju et al., 2007), we collect two sets of data: a 2-class dataset and a 7-class dataset. The 2-class dataset is taken from the original SEMEVAL-2007 task, and comprises a set of positive and negative instances for each of the 7 SRs. The 7-class dataset is derived from this, by combining all positive NCs across the 7 SRs, in line with the methodology of Kim and Baldwin (to appear). The taxonomic relations are derived from WORDNET3.0. In each of the two sets, we use each of hypernyms, hyponyms and sister words. Table 2 shows the number of hyponyms and sister words in each dataset.

Semantic relation	Definition	Examples
Cause-Effect (**CE**)	N_1 is the cause of N_2	*virus flu, hormone growth, inhalation death*
Instrument-Agency (**IA**)	N_1 is the instrument of N_2, N_2 uses N_1	*laser printer, ax murderer, sump pump drainage*
Product-Producer (**PP**)	N_1 is a product of N_2, N_2 produces N_1	*honey bee, music clock, supercomputer business*
Origin-Entity (**OE**)	N_1 is the origin of N_2	*bacon grease, desert storm, peanut butter*
Theme-Tool (**TT**)	N_2 is intended for N_1	*reorganization process, copyright law, work force*
Part-Whole (**PW**)	N_1 is part of N_2	*table leg, daisy flower, tree forest*
Content-Container (**CC**)	N_1 is store or carried inside N_2	*apple basket, wine bottle, plane carge*

Table 1: The set of 7 semantic relations, where N_1 is the head noun and N_2 is a modifier

class	Hyponym		Sister word	
	mod	*head*	*mod*	*head*
7-classes	4866	4708	7167	7456
2-classes (CE)	1272	774	3220	2043
2-classes (IA)	955	1804	1726	3722
2-classes (PP)	1526	1688	3058	3009
2-classes (OE)	2394	1730	3861	2907
2-classes (TT)	1383	812	2767	1698
2-classes (PW)	1403	1770	2900	4117
2-classes (CC)	1598	820	2620	1909

Table 2: Total number of hyponym- and sister word-based NCs

6 7-way classification experiment

We ran our first experiment over the 7-class dataset. The baseline was computed using a *Zero-R* (i.e. majority class) classifier.[1] The performance of the original method proposed in Moldovan et al. (2004) is considered as a benchmark for our experiments. Table 3 shows the performance of the original sense collocation method and that of the extended sense collocation model proposed in this paper.

Table 3 shows that our method, combined with hypernyms, outperforms the original sense collocation method, with the highest accuracy of .588 achieved with 5th-degree ancestors of the head noun and modifier. This confirms that hypernyms are valuable in extending the range of sense collocation for NC interpretation.

In stark contrast to the results for hypernyms, the results for hyponyms and sister words significantly reduced the accuracy. The reason for this anomaly is that hypernyms are able to generalize the sense collocation without losing key discriminative features (i.e. the hypernyms always, by definition, subsume the original semantic information), while hyponyms and sister words add many sense collocations for which we have no direct evidence (i.e. we indiscriminately specialise the semantics without any motivation). Hence, hyponyms and sister words drastically blur the sense collocation.

The reason that the accuracy of the hypernym method drops in beyond a certain level is that the semantic collocations start to blend in together, and lose their power of discrimination.

7 2-way classification experiment

In our second experiment, we ran the systems over the original data from SEMEVAL-2007, in the form of a binary classifier for each of the 7 SRs. The performance of each of the 2-way classification tasks is shown in Table 4.

As we can see in Table 4, the basic pattern of the results is the same as for the 7-way classification task in Table 3. Adding hypernyms enhances performance, peaking for 4th-degree ancestors in this case at .679. As with the 7-way classification task, hyponyms and sister words degraded performance, for the same reasons as before.

Looking at the performance of each SR, we found that some SRs are easier to interpret than others. Notably, PRODUCT-PRODUCER and THEME-TOOL were high performers, while CAUSE-EFFECT was considerably harder to classify. These trends coincide with the system results for the SEMEVAL-2007 task. Girju et al. (2007) analyze this effect in terms of the intrinsic semantic complexity of the different SRs, and also the relative size of the training data for each SR. These effects are also observable in the breakdown of precision and recall of each SR in Figure 1.

As we used the data from the SEMEVAL-2007 task, we are able to directly compare the perfor-

[1]The majority class was PRODUCT-PRODUCER.

	B	M+	H^1	H^2	H^3	H^4	H^5	H^6	H^7	O	S
Accuracy	.217	.496	.544	.552	.573	.562	**.588**	.568	.557	.197	.142

Table 3: Results for the 7-way classification task: B = baseline, M+ = Moldovan et al. (2004) method, H^i = ith-order Hypernym, O = Hyponym and S = Sister word; the best performing system is indicated in **boldface**

	B	M+	H^1	H^2	H^3	H^4	H^5	H^6	H^7	O	S
CE	.547	.547	533	.573	.600	.606	.586	.607	**.630**	.467	.453
IA	.507	.581	.595	.608	.649	**.671**	.653	.629	.645	.500	.500
PP	.655	.667	.679	.691	.679	**.737**	.700	.690	.687	.655	.655
OE	.558	.636	.623	.610	.662	.645	.662	.625	**.712**	.558	.558
TT	.636	.697	.727	.712	.742	**.766**	.732	.717	.650	.515	.394
PW	.634	.620	.690	.690	.629	.657	.585	**.731**	.630	.633	.634
CC	.514	.676	**.703**	.689	.689	.676	.667	.647	.698	.446	.514
All	.579	.632	.649	.653	.662	**.679**	.654	.661	.667	.541	.534

Table 4: Results for each of the 2-way classification tasks: B = baseline, M+ = Moldovan et al. (2004) method, H^i = ith-order Hypernym, O = Hyponym and S = Sister word; the best performing system is indicated in **boldface**

mance of our method with the official results from the competing systems. Table 5 shows the three baselines provided by the SEMEVAL-2007 organisers (see Girju et al. (2007)). Here, *All True* is computed by guessing "true" for all relations, maximizing recall; *probability* is computed by randomly assigning "true" (or "false") with a probability matching the distribution of the labels in the training data for the given relation, and is intended to balance precision and recall; and *majority* is computed by assigning the majority class (either "true" or "false") from the training data for the given relation.

We also present the best-performing system and the average performance within group B from the SEMEVAL-2007 task (the grouping of systems which don't use gold-standard sense tags, and which also don't make use of the "query" used to source the examples).

As shown in Table 5, the performance of our method using hypernyms outperformed all three baselines. The performance using hyponyms and sister words only exceeded the All True and Probability baselines. The interesting point here is that although the method is meant for general-purpose interpretation not for the binary decision task, our proposed method with hypernyms achieves better results than the baselines and is competitive with the

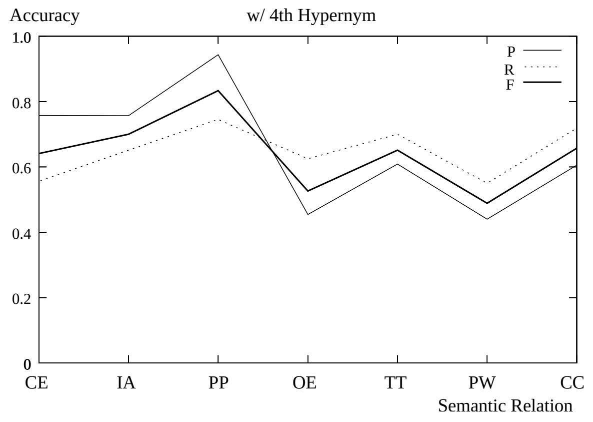


Figure 1: TPR for each of the binary tasks with 4th-degree hypernyms

other systems in the original competition (average accuracy of group B = .656 vs. our best = .679). Therefore, we conclude that sense collocation integrated with hypernyms has the potential to extend the basic sense collocation method and improve performance for the NC interpretation task.

8 Conclusion

In this paper, we have investigated the impact of using different taxonomic relations to expand a sense collocation method of NC interpretation. That is, we

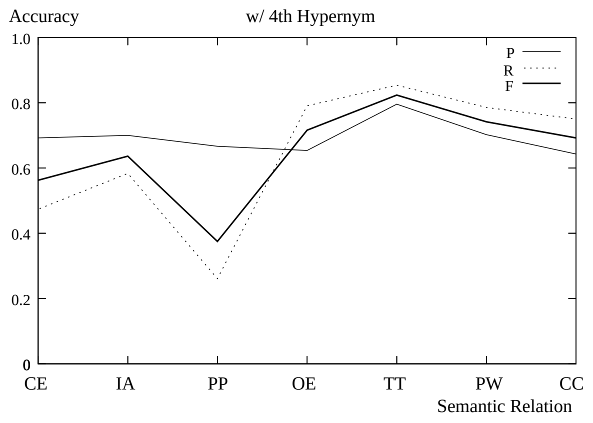


Figure 2: TNR for each of the binary tasks with 4th-degree hypernyms

Method	P	R	F	A
All True	.485	1.00	.648	.485
Probability	.485	.485	.485	.517
Majority	.813	.429	.308	.570
Best	.797	.698	.724	**.763**
Average	.650	.637	.631	.656

Table 5: Results of 2-way classification (P=precision, R=recall, F=F-score, A=accuracy)

experimented with the integration of similar terms into a sense collocation model. We added up to the 7th-degree hypernyms, direct hyponyms and direct sister words terms as features to the classifier. We ran experiments over 7-way and 2-way classification tasks using data from SEMEVAL-2007, and found that the inclusion of hypernym information significantly improved accuracy, while hyponyms and sister words degraded performance by arbitrarily over-specialising the sense information.

While intuitively all of hypernyms, hyponyms and sister words would appear to provide rich features for a sense collocation method, further research is needed to develop ways of successfully incorporating hyponyms and sister words into the NC interpretation task.

References

Ken Barker and Stan Szpakowicz. 1998. Semi-automatic recognition of noun modifier relationships. In *Proc. of the 17th International Conference on Computational Linguistics*, pp. 96–102, Montreal, Canada.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. *TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide*. ILK Technical Report 04-02.

Pamela Downing. 1977. On the Creation and Use of English Compound Nouns. *Language*, 53(4):810–842.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Timothy W. Finin. 1980. *The Semantic Interpretation of Compound Nominals*. Ph.D. thesis, University of Illinois.

Roxana Girju. 2007. Improving the interpretation of noun phrases with cross-linguistic information. In *Proc. of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 568–575, Prague, Czech Republic.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turneyand Deniz Yuret. 2007. SemEval-2007 task 04: classification of semantic relations between nominals. In *Proc. of Semantic Evaluation Workshop*, Prague, Czech Republic.

Su Nam Kim and Timothy Baldwin. 2005. Automatic interpretation of noun compounds using WordNet similarity. In *Proc. of the 2nd International Joint Conference On Natural Language Processing*, pp. 945–956, JeJu, Korea.

Su Nam Kim and Timothy Baldwin. 2006. Interpreting semantic relations in noun compounds via verb semantics. In *Proc. of the 44th Annual Meeting of the Association for Computational Linguistics and 21st International Conference on Computational Linguistics (COLING/ACL-2006)*. pp. 491–498, Sydney, Australia.

Su Nam Kim and Timothy Baldwin. 2007. Disambiguating noun compounds. In *Proc. of the 22nd AAAI conference on Artificial Intelligence (AAAI-07)*, 2007, pp.901-906, British Columbia, Canada.

Su Nam Kim and Timothy Baldwin. 2007. Interpreting noun compounds using bootstrapping and sense collocation. In *Proc. of the Pacific Association for Computational Linguistics (PACLING)*, Melbourne, Australia.

Su Nam Kim and Timothy Baldwin. to appear. Benchmarking noun compound interpretation. In *Proc. of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-08)*, Hyderabad, India.

Maria Lapata. 2002. The disambiguation of nominalizations. *Computational Linguistics*, 28(3):357–388.

Mark Lauer. Designing Statistical Language Learners: Experiments on Noun Compounds, Ph.D thesis. 1995, Macquarie University.

Judith Levi. 1979. The syntax and semantics of complex nominals. In *The Syntax and Semantics of Complex Nominals*. New York:Academic Press.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *Proc. of the HLT-NAACL 2004 Workshop on Computational Lexical Semantics*, pp. 60–67, Boston, USA.

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proc. of the 9th Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 17-24, Ann Arbor, Michigan, USA.

Preslav Nakov and Marti Hearst. 2006. Using verbs to characterize noun-noun relations. In *Proc. of the 12th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA)*, Bularia.

Vivi Nastase and Jelber Sayyad-Shiarabad and Marina Sokolova and Stan Szpakowicz. 2006. Learning noun-modifier semantic relations with corpus-based and WordNet-based features. In *Proc. of the 21st Annual Conference on Artificial Intelligence (AAAI-06)*, pp. 781-787, Boston, MA, USA.

Barbara Rosario and Hearst Marti. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *In Proc. of EMNLP*, 82–90

Karen Sparck Jones. 1983. Compound noun interpretation problems. *Computer Speech Processing*, Prentice-Hall, Englewood Cliffes, NJ, USA.

Diarmuid Seaghdha and Ann Copestake. 2007. Co-occurrence contexts for noun compound interpretation. In *Proc. of the ACL-2007 Workshop on A Broader Perspective on Multiword Expressions*, Prague, Czech Republic.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *Proc. of the 15th Conference on Computational linguistics*, pp. 782–788.

Proceedings of the Australasian Language Technology Workshop 2007, pages 57-65

Named Entity Recognition in Question Answering of Speech Data

Diego Mollá **Menno van Zaanen** **Steve Cassidy**
Division of ICS
Department of Computing
Macquarie University
North Ryde
Australia
{diego,menno,cassidy}@ics.mq.edu.au

Abstract

Question answering on speech transcripts (QAst) is a pilot track of the CLEF competition. In this paper we present our contribution to QAst, which is centred on a study of Named Entity (NE) recognition on speech transcripts, and how it impacts on the accuracy of the final question answering system. We have ported AFNER, the NE recogniser of the AnswerFinder question-answering project, to the set of answer types expected in the QAst track. AFNER uses a combination of regular expressions, lists of names (gazetteers) and machine learning to find NeWS in the data. The machine learning component was trained on a development set of the AMI corpus. In the process we identified various problems with scalability of the system and the existence of errors of the extracted annotation, which lead to relatively poor performance in general. Performance was yet comparable with state of the art, and the system was second (out of three participants) in one of the QAst subtasks.

1 Introduction

AnswerFinder is a question answering system that focuses on shallow semantic representations of questions and text (Mollá and van Zaanen, 2006; van Zaanen et al., 2007). The underlying idea of AnswerFinder is that the use of semantic representations reduces the impact of paraphrases (different wordings of the same information). The system uses symbolic algorithms to find exact answers to questions in large document collections.

The design and implementation of the AnswerFinder system has been driven by requirements that the system should be easy to configure, extend, and, therefore, port to new domains. To measure the success of the implementation of AnswerFinder in these respects, we decided to participate in the CLEF 2007 pilot task of question answering on speech transcripts (QAst). The task in this competition is different from that for which AnswerFinder was originally designed and provides a good test of portability to new domains.

The current CLEF pilot track QAst presents an interesting and challenging new application of question answering. The objective in QAst is to answer questions based on transcripts of meetings and lectures. Both automatic and manual transcripts are provided; the automatic transcripts being the result of applying a speech recogniser to the audio recordings. The data for the task is taken from corpora collected by the AMI (Augmented Multiparty Interaction) project (Carletta et al., 2005) and from the CHIL (Computers in the Human Interaction Loop) project (Waibel et al., 2004). While both corpora are extensively annotated, only speaker turn annotation is provided in the input data for this task.

In our contribution we focus on adapting AFNER, our Named Entity Recogniser (NER), for speech transcripts and its application for Question Answering. Named Entity (NE) recognition is the task of finding instances of specific types of entities in free text. This module is typically one of the most important sources of possible answers available to QA systems and therefore an improvement on its accuracy should result on an improvement of the accuracy of the complete QA system.

The AFNER system, like the AnswerFinder system, was designed with flexibility in mind. Since the properties of the NE recognition task in this competition are in several respects quite different to those of the task AFNER was originally designed for (as discussed in section 3.3), the QAst competition also allows us to measure the success of our AFNER implementation according to the configurability and extensibility criteria.

2 Question Answering on Speech Transcripts

The task of Text-Based Question Answering (QA) has been very active during the last decade, mostly thanks to the Question Answering track of the Text REtrieval Conference (TREC) (Voorhees, 1999). The kinds of questions being asked range from fact-based questions (also known as factoid questions) to questions whose answer is a list of facts, or definitions. The methods and techniques used have converged to a prototypical, pipeline-based architecture like the one we will describe here, and only recently the task has been diversified to more complex tasks such as TREC's QA task of complex interactive question answering (Dang and Lin, 2007) or the Document Understanding Conference (DUC)'s track of query-driven summarisation (Dang, 2006).

Whereas the TREC competitions concentrate on searching in English texts, CLEF (Cross-Language Evaluation Forum) focuses on non-English, cross-lingual and multi-lingual search. Within this forum several competitions are organised. The QAst track deals with question answering on speech data.

Prior to the QAst pilot track of CLEF there has been very little work on the area of question answering of speech data. Much of the work has focused on the task of recognising named entities by applying machine learning using features that leverage the very special kinds of information of speech data, particularly the lack of punctuation and capitalisation information. The work by Surdeanu et al. (2005) is an example of such an approach. Another line of work tries to recover the lost capitalisation information by using machine learning methods trained on regular text and tested on text where all capitalisation information has been removed. This is the approach followed, for example, by Li et al. (2003). Note, however, that Li et al. did not work on speech data as we are trying to do here but on regular text where case information has been removed. As we discuss below, speech data have many other factors that need to be taken into consideration.

Two data sets were provided by CLEF for development of systems participating in the evaluation. These were transcripts of lectures taken from the CHIL (Waibel et al., 2004) project and meetings from the AMI (Carletta et al., 2005) project. We made use of the AMI data because we had access to the original annotations which included named entities. This data consists of transcripts of 35 meetings each with up to four speakers. These contained around 254,000 words of dialogue. Due to disk space constraints we only made use of 15 meetings containing around 160,000 words in the development of our system.

2.1 AnswerFinder

The AnswerFinder question answering system is essentially a framework consisting of several phases that work in a sequential manner. For each of the phases, a specific algorithm has to be selected to create a particular instantiation of the framework. The aim of each of the phases is to reduce the amount of data the system has to handle from then on. This allows later phases to perform computationally more expensive operations on the remaining data.

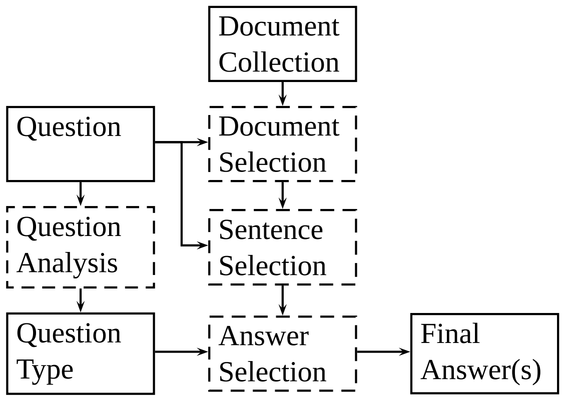


Figure 1: AnswerFinder system overview

Figure 1 provides an overview of the AnswerFinder framework. The first phase is a document retrieval phase that selects documents relevant to the question. AnswerFinder was developed to work on large document collections and this phase can make a significant reduction in the amount of text that must be handled in subsequent steps.

Next is the sentence selection phase which selects a subset of sentences from the relevant documents selected in the previous phase. During sentence selection, all sentences that are still left (e.g. all sentences in the selected documents in the first step) are scored against the question using a relevance metric. The most relevant sentences according to this metric are kept for further processing. This phase can be applied to the remaining sentences several times using different metrics, each time reducing the number of sentences further.

After sentence selection, the remaining sentences are passed to the answer selection phase. The answer selection phase aims at selecting the best of the possible answers to return to the user. In the experiments described here, the list of possible answers is generated by applying a NER to the remaining sentences.[1]

Next, the question is analysed, providing information about the kind of answer that is required. From the possible answers, those that match the type of answer required by the question are selected and scored. Finally, the best answer is returned to the user. Best answer in this context is considered to be the answer that has both the highest score and an answer type that matches the question, or simply the answer with the highest score if none of the possible answers fit the expected answer type.

2.2 Applying AnswerFinder to Speech Transcripts

Question answering on speech transcripts introduces specific challenges compared to text-based QA due to the nature of the genre and the process of transcription. AnswerFinder has been initially developed to work on news articles which are typically well-written pieces of text. The casual, multi-party spoken language used in this evaluation is very different. For example,

- There are frequent false starts and sentences that are interrupted in the discourse.
- There are filling words that usually do not appear in free text (and in particular news text), such as “er”, “uh”, etc. In our experiments, this is particularly problematic when these words appear inside a named entity, e.g. “Rufford, um, Sanatorium, that’s right”.
- The grammatical structure of the transcription does not conform to that of free text. Consequently most tools, such as parsers and chunkers, which would normally be used in specific AnswerFinder phases, produce very poor results.
- If the transcript is an automatic transcript (produced by a speech recogniser) there are errors of transcription and missing information, most notably punctuation characters and capitalised characters. This information is used in many phases of AnswerFinder when answering questions on news data.
- During training, a corpus annotated with named entities is used. The density of named entities in free speech is much smaller than in usual corpora (containing news text).

Many of the above features make it difficult to do traditional linguistic processing such as parsing and semantic interpretation. For this reason, many of the instantiations of the phases we have implemented, which typically use complex linguistic processing (as described in van Zaanen et al. (2007)) would not perform well. We consequently decided not to use some of AnswerFinder’s more linguistically-intensive modules. Instead we focused on attempting to increase the accuracy of the task of recognition of named entities. Thus, the question answering method used for QAst is entirely based on the task of finding and selecting the right entities.

In particular, the instantiation of the AnswerFinder framework that generated the QAst 2007 results consists of the following algorithms for the phases:

[1] In general, some sentence selection methods have the ability to generate possible answers that can also be selected during the answer selection phase. However, these algorithms are not used in these experiments as will be discussed in section 2.2.

- The document selection component returns the full list of documents provided for the complete list of questions. The total number of documents provided by the organisers of QAst is fairly small and therefore the other components of AnswerFinder are able to handle all documents. Essentially no documents are preselected in this instantiation. We do not attempt to rank the documents in any way.

- As a pre-processing step, the named entity recogniser is run over all the documents. This allows for more efficient handling of the set of questions, as named entity recognition only has to occur once.

- The sentence selection component is based on the word overlap between the question and the document sentences. This metric counts the number of words that can be found in both question and sentence after removing stop words. A simple sentence splitter method is used, which relies on the existence of punctuation marks when available, or on speech turns. Only sentences that contain NEs of the required type are considered.

- Each of the named entities found in the selected sentences are scored. The score of a NE is the sum of the number of occurrences of that NE with a particular type.

- The question classification component is based on a decision list of hand-constructed patterns of regular expressions. Each regular expression determines a question type and consequently a set of NE types.

- The answer extraction component selects five NEs that are of the expected answer type and have the highest NE scores. QAst allows for the system to return up to five answers. If four or fewer NEs of the correct type are found, then a NIL answer (meaning no answer) is returned as an option after presenting all found NEs. If no NEs of the expected type are found at all, the returned answer is NIL.

3 AFNER

Within the AnswerFinder project, we recently incorporated a purpose-built NER, called AFNER (Mollá et al., 2006). This NER has been specifically designed for the task of QA. AFNER differs from other NERs in that it aims to increase recall of recognition of entities, at the expense of a possible loss of precision (Mollá et al., 2006; van Zaanen and Mollá, 2007). Crucially, it allows the allocation of multiple tags to the same string, thus handling the case of ambiguous entities or difficult entities by not committing to a single tag. The rationale is that we do not want to remove the right answer at this stage. Instead we let the final answer extraction and scoring mechanism make the final decision about what is a good answer.

AFNER is ultimately based on machine learning. We use a maximum entropy classifier, and the implementation of this classifier is adapted from Franz Josef Och's *YASMET*[2]. Obviously, the selection of the features used in the classifier is very important.

3.1 Features

The features used by AFNER combine three kinds of information: regular expressions, gazetteers, and properties internal and external to the token. These features are described in more detail elsewhere (Mollá et al., 2006; van Zaanen and Mollá, 2007) and we will only briefly present them here.

The regular expressions used in AFNER are manually created and are useful for identifying strings that match patterns that are characteristic to entity types such as dates, times, percentages, and monetary expressions. These types of named entities are relatively standardised and therefore easy to find with high precision. However, the range of entities that can be discovered using regular expressions is limited. Matching a particular regular expression is a key feature used in identifying entities of these particular types.

Gazetteers are useful for finding commonly referenced entities such as names. AFNER uses three lists (locations, person names, and organisations), with a total of about 55,000 entries. The occurrence of tokens in one of the gazetteers is incorporated in the machine learning component. This allows for,

[2]`http://www.fjoch.com/YASMET.html`

Regular Expressions	Specific patterns for dates, times, etc
FoundInList	The token is a member of a gazetteer
InitCaps	The first letter is a capital letter
AllCaps	The entire word is capitalised
MixedCaps	The word contains upper case and lower case letters
IsSentEnd	The token is an end of sentence character
InitCapPeriod	Starts with capital letter and ends with period
OneCap	The word is a single capitalised letter
ContainDigit	The word contains a digit
NumberString	The word is a number word ('one', 'thousand', etc.)
PrepPreceded	The word is preceded by a preposition (in a window of 4 tokens)
PrevClass	The class assigned to the previous token
ProbClass	The probability assigned to a particular class in the previous token
AlwaysCapped	The token is capitalised every time it appears

Table 1: A selection of features used in AFNER

for example, context information in the final decision of the tag assignment for that particular token.

Finally, there are three types of features that relate to specific aspects of the separate tokens. The first type focuses on internal evidence and highlights token properties including capitalisation, alpha/numeric information, etc. Some specific features are listed in Table 1.

The second type of features focuses on external evidence that relates a token to tokens in surrounding text. Features that indicate which class has been assigned to the previous tokens and all of its class probabilities are also part of this type of feature.

The last type of features focuses on global evidence related to all occurrences of the same token. These features are mainly inspired on features described by Chieu and Ng (2002). Currently AFNER only checks whether a token is always capitalised in a passage of text.

3.2 General Method

The features described in the previous section are used in a maximum entropy classifier which for each token and for each category computes the probability of the token belonging to the category. Categories in this case are the named entity types prepended with 'B' and 'I' (indicating whether the token is at the beginning or inside a NE respectively), and a general 'OUT' category for tokens not in any entity. So for n named entities, $n * 2 + 1$ categories are used.

The classifier returns a list of tags for each token ordered based on probability. We select only those tags that have a probability of more than half of the probability of the next tag in the list. This initial threshold already removes tags that have a low probability. However, we also only allow a certain maximum number of tags to pass through. Preliminary experiments revealed that often the top two or three tag probabilities have similar values, but that tags lower down the list still pass the initial threshold, while they are not correct. By setting a threshold that limits the maximum number of tags to be returned we also filter those out. The results presented in this paper are generated by setting the second threshold to allow two tags per token. Initial experiments showed that this increases recall considerably. Allowing more tags increases recall only slightly while decreasing precision considerably.

Once tokens are assigned tags, they are combined to produce the final list of multi-word NEs as described elsewhere (Mollá et al., 2006; van Zaanen and Mollá, 2007). The result is an assignment of named entities to the sequence of tags where the named entities may be nested. This way we aim at high recall by allowing multiple interpretations of problematic strings that could be ambiguous.

Class	Type	# in BBN	# in AMI
ENAMEX	Language	9	0
	Location	2,468	16
	Organization	4,421	27
	Person	2,149	196
	System	0	448
	Color	0	283
	Shape	0	147
	Material	0	267
TIMEX	Date	3,006	9
	Time	96	147
NUMEX	Measure	2,568	293
	Cardinal	0	646

Table 2: Named Entities used for QAst. The numbers of entities listed in the two last columns refer to the actual training set (a subset of BBN and AMI).

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

Table 3: Entity types used in the original version of AFNER

3.3 Adaptation of AFNER to QAst

AFNER has been developed to work on news data, and as such, we had to modify parts of the system to allow it to be used in the QAst task. The first adaptation of AFNER is the selection of NE types. Originally AFNER focused on a limited set of entities similar to those defined in the Message Understanding Conferences (Sundheim, 1995), and listed in Table 3.

For QAst we used a set of entity types that closely resembles the kinds of answers expected, as described by the QAst 2007 specification. The types used by the modified AFNER are listed in Table 2.

The regular expressions that are used in AFNER to find MUC-type named entities were extended to cover the new types of entities. This process did not require much additional work, other than adding a few common names of shapes and colours. The lists of names that was part of the initial AFNER was left untouched.

The general machine learning mechanism was left unmodified, and the set of features was also left untouched. The only difference was the choice of training corpus. We mapped the annotated entities of the BBN corpus that we had used previously, and added a fragment of the development set of the AMI corpus.

However, due to problems of scalability during training (the intermediate files produced were very large due to the increased number of classifier categories) we were not able to use all the files. For these experiments we used 26 documents from the AMI corpus and 16 from the BBN corpus. Table 2 shows the total number of entities annotated in the BBN and the AMI parts of the training set. The entity types of each kind of corpus complement each other, though some of the entity types had few instances in the corpora, most notably, the type Language only occurred nine times.

We decided to use the BBN corpus to complement the annotations of AMI because some entity types that were very scarce in AMI were very common in BBN. Also, the entity types annotated in AMI are not the sort of types that would typically be annotated as named entities. For example, the entity type “Person” would have instances like *industrial designer*. Furthermore, the quality of some of the annotations of the AMI corpus was poor. In at least two of the 26 meetings the contents of named entities seemed to be random strings. After submitting the results, we found a bug in our corpus processing script which resulted in some named entities having extra words included in them.

4 Results

We participated in all the QAst tasks, which are described below:

CHIL$_M$ Manual transcripts from the CHIL corpus of lectures;

CHIL$_A$ Automated transcripts from the CHIL corpus;

AMI$_M$ Manual transcripts from the AMI corpus of meetings; and

AMI$_A$ Automated transcripts from the AMI corpus.

We provided two runs per task. We were interested on determine the impact of the machine learning component of AFNER. Given the reduced number of training documents and the existence of errors in some of them we expected that the machine learning component would not be useful. Thus, the first run used the full AFNER system, whereas the second run (named “noML”) used a version of AFNER that had the machine learning component disabled (essentially only using the regular expressions and the gazetteers). The results are shown in Table 4.

The results returned by CLEF indicate, as expected, comparatively poor performance with respect to the other participants. We are pleased to notice, however, that the results of task CHIL$_M$ are second best (from a group of three participants). Task CHIL$_M$ is the task that used the AMI transcripts and it was the task that we used to develop and fine-tune the system. The other tasks simply used the same settings. We are particularly pleased to learn that the results of task CHIL$_M$ are higher than the results we obtained during development time. This is possibly due to the nature of our tuning experiments, since we automatically applied the answer patterns to the answers found, and it could have been the case that correct answers which happened not to match the patterns were automatically marked as incorrect in our experiments. The evaluations carried by CLEF used human judges so they would be able to detect correct answers that had an unusual format.

The results indicate that none of the differences in results between the full and the noML runs are statistically significant under the paired t-test. This confirms our suspicion that the machine learning component of AFNER was not helping the question answering process at all. The likely reason for this is, as described above, the small size of the training data and the existence of noise in the NE annotations of the AMI corpus.

Our method to handle NIL questions is simple yet relatively effective to the point that correct NIL answers were an important part of the correct answers. Task AMI$_A$ in particular, which has 15 NIL questions, results in a halved MRR (from 14.10% down to 7.05% in our noML run) when all NIL questions are removed. It is encouraging to observe that, even after removing all NIL questions, task CHIL$_M$ has relatively good results (from 26.39% down to 22.38% in our noML run). The results of the non-NIL questions are shown in Table 5.

5 Conclusions and Further Work

In our contribution to the QAst competition we reused as much as we could of AnswerFinder, our question answering system, and AFNER, our Named Entity recogniser. Due to the nature of the speech corpus we needed to simplify the processing done by AnswerFinder and made it rely more heavily on the entities found by AFNER. The whole experiment showed successfully that both AnswerFinder and AFNER are flexible and can be adapted easily to new tasks.

The small training corpus and the presence of annotation errors in the AMI corpus made the machine learning component of AFNER ineffective. An immediate line of further research is to investigate the cause of the errors, and correct them. Other lines of research are:

- Revise the machine learning component of AFNER, possibly replace it with another more scalable method, so that larger training corpora can be used. Currently we are investigating more efficient ways of storing the intermediate data.

Run	**Questions**	**Correct Answers**	**MRR**	**Accuracy**
full-CHIL$_M$	98	17.35%	9.98%	6.12%
noML-CHIL$_M$	98	16.33%	9.44%	5.10%
full-CHIL$_A$	98	14.29%	7.16%	3.06%
noML-CHIL$_A$	98	12.24%	5.88%	2.04%
full-AMI$_M$	96	35.42%	24.51%	16.67%
noML-AMI$_M$	96	33.33%	26.39%	20.83%
full-AMI$_A$	93	19.35%	11.24%	6.45%
noML-AMI$_A$	93	22.58%	14.10%	8.60%

Table 4: Results of the CLEF runs

Run	**Questions**	**Correct Answers**	**MRR**	**Accuracy**
full-CHIL$_M$	88	12.50%	8.56%	6.82%
noML-CHIL$_M$	88	11.36%	7.95%	5.68%
full-CHIL$_A$	87	5.75%	4.06%	3.45%
noML-CHIL$_A$	87	3.45%	2.87%	2.30%
full-AMI$_M$	86	29.07%	22.33%	18.60%
noML-AMI$_M$	86	25.58%	22.38%	19.77%
full-AMI$_A$	79	6.33%	3.90%	2.53%
noML-AMI$_A$	78	8.97%	7.05%	5.13%

Table 5: Results of non-NIL questions

- Review the features used for identifying the entities. Most of the current features rely on information about capitalisation, presence of digits, or punctuation marks but none of those are available on speech transcripts. In practice, using features that always provide the same values means that the machine learning component does not add much to the non-machine learning information, as shown in the experiment. More useful features will increase the use of the machine learning component.

- Use additional corpora. There are a few corpora of speech transcriptions available with annotations of named entities that we could use. Among the options is the corpus of speech transcripts within the SQUAD project with the UK Data Archive at the University of Edinburgh.

To conclude, question answering on speech transcripts is a challenging task that deserves greater attention by the research community. The CLEF QAst track is a step toward facilitating research on this area. Our participation in QAst is a step from our side to contribute to this exciting research area.

References

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, Iain A. McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. 2005. The ami meetings corpus. In L. P. J. J. Noldus, F. Grieco, L. W. S. Loijens, and Patrick H. Zimmerman, editors, *Proceedings of the Measuring Behavior 2005 symposium on "Annotating and measuring Meeting Behavior"*. AMI-108.

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

Hoa Dang and Jimmy Lin. 2007. Different structures for evaluating answers to complex questions: Pyramids won't topple, and neither will human assessors. In *Proceedings ACL*.

Hoa Tran Dang. 2006. Duc 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55, Sydney. Association for Computational Linguistics.

Wei Li, Rohini Srihari, Cheng Niu, and Xiaoge Li. 2003. Question answering on a case insensitive corpus. In

Proc. ACL 2003 Workshop on Multilingual Summarization and Question Answering, pages 84–93.

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

Diego Mollá, Menno van Zaanen, and Luiz A.S. Pizzato. 2006. Named entity recognition for question answering. In *Proceedings ALTW 2006*, page 8 pages.

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

Mihai Surdeanu, Jordi Turmo, and Eli Comelles. 2005. Named entity recognition from spontaneous open-domain speech. In *Proceedings Interspeech-05*, Lisbon.

Menno van Zaanen and Diego Mollá. 2007. A named entity recogniser for question answering. In *Proceedings PACLING 2007*.

Menno van Zaanen, Diego Mollá, and Luiz Pizzato. 2007. Answerfinder at trec 2006. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings TREC 2006*, page 8 pages.

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

A. Waibel, H. Steusloff, and R. Stiefelhagen. 2004. Chil - computers in the human interaction loop. In *5th International Workshop on Image Analysis for Multimedia Interactive Services*, Lisbon, Portugal.

Proceedings of the Australasian Language Technology Workshop 2007, pages 66-74

Experiments in Mutual Exclusion Bootstrapping

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

Abstract

Mutual Exclusion Bootstrapping (MEB) was designed to overcome the problem of *semantic drift* suffered by iterative bootstrapping, where the meaning of extracted terms quickly drifts from the original seed terms (Curran et al., 2007). MEB works by extracting mutually exclusive classes in parallel which constrain each other.

In this paper we explore the strengths and limitations of MEB by applying it to two novel lexical-semantic extraction tasks: extracting bigram named entities and WordNet lexical file classes (Fellbaum, 1998) from the Google Web 1T 5-grams.

1 Introduction

Extracting lexical semantic resources from text with minimal supervision is critical to overcoming the knowledge bottleneck in Natural Language Processing (NLP) tasks ranging from Word Sense Disambiguation to Question Answering.

Template-based extraction is attractive because it is reasonably efficient, works on small and large datasets, and requires minimal linguistic preprocessing, making it fairly language independent. Hearst (1992) proposed template-based extraction for identifying hyponyms using templates like X, Y, and/or other Z where X and Y are hyponyms of Z.

Riloff and Shepherd (1997) proposed *iterative bootstrapping* where frequent neighbours to terms from a given semantic class are extracted in multiple bootstrap iterations. Roark and Charniak (1998) improved its accuracy by optimising the bootstrapping parameters. In *mutual bootstrapping* (Riloff and Jones, 1999) the terms, and the contexts they occur in, are extracted. Similar approaches have been used in Information Extraction (IE) for identifying company headquarters (Agichtein et al., 2000) and acronym expansions (Sundaresan and Yi, 2000).

In *Mutual Exclusion Bootstrapping* (MEB), we assume the semantic classes partition terms into disjoint sets, that is, the classes are *mutually exclusive* (Curran et al., 2007). Each class is extracted in parallel using separate bootstrapping loops that each race to collect terms and contexts. Although this assumption is clearly false, it significantly reduces the extraction errors of existing approaches.

This paper presents two applications of MEB that allow some insight into MEB's strengths and limitations. First, we extend MEB to extracting *bigram* BBN named entity types (Weischedel and Brunstein, 2005). We discover that both unigram and bigram MEB are very sensitive to the context window surrounding the extracted terms. Surprisingly, MEB is insensitive to the order the semantic classes are presented and the noise in the terms themselves.

Second, we extract common nouns using 25 semantic classes defined by the WordNet lexical files (Fellbaum, 1998). We use a closed vocabulary of WordNet unigram nouns, so the evaluation can be performed automatically against WordNet. We find that MEB performs well on classes with narrow definitions and thus more coherent contexts, such as animal, but performs poorly on classes like cognition. We also find that increasing the number of seed terms increases the accuracy significantly.

2 Mutual Exclusion Bootstrapping

Mutual bootstrapping (Riloff and Jones, 1999) has the advantage that it can identify new templates or *contexts*, which in turn identify new terms, significantly increasing recall. Unfortunately, erroneously adding a term with a different predominant sense or a context that weakly constrains the terms, quickly leads to *semantic drift*, where erroneous terms or contexts infect the semantic class.

Mutual Exclusion Bootstrapping (MEB) attempts to minimise semantic drift in both the terms and contexts (Curran et al., 2007). It does this by extracting all of the semantic classes in parallel, using an independent bootstrapping loop for each class, with the constraint that a term or context *must only be used by one* class. We assume that each term has only one sense and that each context only extracts terms with one sense, that is, the semantic classes are *mutually exclusive* with respect to terms and contexts.

This assumption is far from realistic, but it is very effective at reducing the degree of semantic drift. For many terms, especially the bigram named entities, there is a clearly dominant semantic class. However, for some pairs of semantic classes, e.g. nationalities and languages, there is a significant lexical overlap and so they are far from mutually exclusive.

The MEB algorithm is shown in Algorithm 1. In each iteration, contexts and then terms are added to each semantic class. If more than one class attempts to extract a context or term in the current iteration then it is eliminated, leading to mutual exclusion between the semantic classes. The terms and contexts are ranked in the same way as Riloff and Jones (1999), our only addition in MEB is the parallel mutual exclusion constraint.

Mutual exclusion is very strict and many terms and contexts are discarded. This is not a major issue when precision is paramount and we are using a large dataset, e.g. Web 1T, but it can be problematic on smaller datasets. It is a significant problem when the semantic classes are far from mutually exclusive because many viable contexts are rejected when the terms they extract are polysemous, even though the contexts themselves reliably select one sense.

MEB is potentially sensitive to the order the contexts and terms are added to semantic classes, since once they are added to a class they cannot be added elsewhere (by the mutual exclusion assumption). In this sense, the individual bootstrapping loops compete in parallel to reach a term or context first, and claim it for themselves. Polysemous terms may be added to just one semantic class if it is not identified by contexts from multiple semantic classes simultaneously, and this also applies for contexts. There is no guarantee that the predominant sense of a term will be reached first, although if it is significantly more frequent, it is likely to be reached first since it will appear in more contexts.

```
in  : Seed word lists $S_k$ $\forall$ categories $k$
in  : Raw contexts $\mathscr{C}$ and terms $\mathscr{T}$
in  : # terms $N_T$ and contexts $N_C$ per iteration
out: Term $T_k$ and context $C_k$ lists $\forall$ categories $k$
$T_k \longleftarrow S_k \forall$ categories $k$;
foreach iteration do
    foreach $c \in \mathscr{C}$ do
        count # times $c$ occurs with each $t \in T_k$;
        discard $c$ if occurs with multiple classes;
    foreach class $k$ do
        sort set of $c$ by above occurrence counts;
        add top $N_C$ contexts to $C_k$;
    foreach $t \in \mathscr{T}$ do
        count # times $t$ occurs with each $c \in C_k$;
        discard $t$ if occurs with multiple classes;
    foreach class $k$ do
        sort set of $t$ by above occurrence counts;
        add top $N_T$ terms to $T_k$;
```

Algorithm 1: Mutual Exclusion Bootstrapping

3 Using the Google Web 1T n-grams

Riloff and Jones (1999) used contexts extracted from POS tagged and chunked text by AutoSlog-TS (Riloff, 1996). Our goal was to keep MEB language independent to maintain this key advantage of template-based approaches. We also wanted to demonstrate that MEB scales efficiently to extremely large datasets and vocabularies.

Google has collected the Web 1T corpus (Brants and Franz, 2006), which consists of unigram to 5-gram counts calculated over 1 trillion words of web text collected during January 2006. The text was tokenised using Penn Treebank tokenisation, except that words are usually split on hyphens; and dates, email addresses, and URLs are kept as single tokens. Sentence boundaries were detected using statistical techniques. The individual words in the n-grams occurred ≥200 times, otherwise they were replaced with <UNK>. Each n-gram appears ≥40 times. There is 25GB of compressed data.

We use the 3-, 4-, or 5-grams from Web 1T as our raw data, depending on the experiment. The middle token (for unigrams) or tokens (for bigrams) form the *term* and the one or two tokens on either side form the *context*. This context definition is quite language independent (except for languages without word segmentation). Unfortunately, we can only extract terms consisting of one or two words, and the contexts are noisier than those extracted from parsed text, cf. Curran (2004).

For the bigram experiments we follow the process described in Curran et al. (2007). We removed n-grams with non-titlecase middle token(s) because we only extract proper noun named entity types, and we removed all contexts containing numbers. For the WordNet experiments we only included n-grams where the middle token(s) were a term in WordNet. In every experiment, we eliminate contexts that only appear with one term, and thus terms that only appear in one context, since they cannot be reached.

The size of the resulting dataset varied depending on the experiment from 176MB (for the bigrams heavily filtered using the t-test) to 1.2GB (for the bigrams with a window of one word either side and the WordNet experiments). All of the data must be loaded into memory and for the largest experiments this requires 1.6GB of RAM using our space-optimised C++ implementation.

4 Named Entity Classes

In our first set of experiments we continue our previous work on proper-noun named entities. We based our semantic classes on the 29 entity types used to annotated the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005). We ignored entity types that did not primarily include proper nouns, for example the DESCRIPTION types, CHEMICALS and QUANTITIES.

For the unigram experiments we reused our previous classification where we ignored entity types that were almost exclusively multi-word terms, for example WORKS OF ART and LAWS. We also split the PERSON class into MALE and FEMALE first names

LABEL	UNI	BI	DESCRIPTION
NAME		●	Person: name *'Katie Holmes' 'Adam Smith'*
FEM	●		Person: female first name *Mary Patricia Linda Elizabeth*
MALE	●		Person: male first name *James John Robert Michael William*
LAST	●		Person: last name *Smith Johnson Williams Jones Brown*
TITLE	●	●	Honorific title *President Dr Lord Miss Major*
NORP	●	●	Nationality, Religion, Political (adj) *Republican Christian 'South African'*
FAC	●	●	Facility: names of man-made structures *Broadway Legoland 'Golden Gate'*
ORG	●	●	Organisation: e.g. companies, gov. *Intel Microsoft 'American Express'*
GPE	●	●	Geo-political entity *Canada China London 'Los Angeles'*
LOC	●	●	Locations other than GPEs *Africa Asia Pacific Earth 'Middle East'*
DATE	●	●	Reference to a date or period *January May Friday 'Easter Day'*
LANG	●	●	Any named language *English Arabic Hebrew 'Scots Gaelic'*
EVENT		●	Battles, sporting, and other named events *'World War' 'Hurricane Katrina'*
LAW		●	Document that has been made into a law *'Reform Act' 'First Amendment'*

Table 1: The semantic classes used for the proper noun unigram (Column 2) and bigram (Column 3) experiments. Bigram examples are shown in quotes.

and LAST names to investigate more fine-grained distinctions for this class.

For the bigram experiments, we kept a single class NAME for person name, and reintroduced the LAW and EVENT classes. Most classes are common to both the unigram and bigram experiments. As in our previous experiments, some classes were easier to evaluate manually because we were only extracting unigrams, whilst others were more difficult. Similar difficulties exist in the bigram classes as well. The complete list of semantic classes used in the named entity experiments are summarised in Table 1.

5 Named Entity Evaluation

Our evaluation followed the manual inspection process used in our previous experiments. To make this more efficient, we stored a cache of previous evaluator decisions for each class, so that once a decision had been made for a particular term in a particular class it would be made automatically in future instances. This dramatically reduces the effort required for manual evaluation.

Although the seed lists were mutually exclusive, for the purposes of evaluation, ambiguous words such as *French* were counted as correct if they appeared in either valid category (NORP or LANG).

If a single word was an clearly part of a multi-word term we counted it as correct (e.g. *Coast* as a LOC) with the exception of the mixed unigram-bigram experiments. If the word was not strongly indicative of a semantic class (e.g. *The*) it was not counted as correct. Mis-spellings of words (e.g. *Januray*) were also counted as correct. The extracted terms that were unrecognised by the human evaluator were checked using Wikipedia and Google.

We calculate accuracy at n – the percentage of correct terms in the top n ranked terms, following previous bootstrapping work. This is averaged over the semantic classes (Av(n)). We manually evaluated all semantic classes down to $n = 50$, which adequately discriminates between most configurations. We vary the number of seeds (nS), and terms (nT) and context (nC) added in each iteration.

6 Named Entity Experiments

Our initial expectation was that bigram named entities would be an easier task than unigram named entities because they had fewer senses and so better satisfied the mutual exclusion assumption. Also, we expected them to be easier to evaluate since they were less ambiguous. However, the results did not match our intuition and so we experimented with unigrams and bigrams to determine the cause.

6.1 Context Geometry

A major disadvantage for the bigram and longer n-gram experiments is that the size of the context must be reduced to accommodate the term itself within a fixed sized n-gram (e.g. the Web 1T 5-grams). Even if longer n-grams were collected for bootstrapping, there would still be the problem of sparser counts (even from one trillion words).

We started by repeating our original unigram named entity experiments but this time we reduced the context window to one token on the left and/or right, as shown in Table 2. Table 3 shows the impact of context geometry on unigram accuracy. Our previous best unigram results are UNI5GMS with 78%. Removing a token from the right context (UNI4LEFT) makes almost no difference to the results, but removing a token from the left context (UNI4RIGHT) makes an enormous difference (a loss of almost 30%). The effect of removing both (UNI3GMS) is slightly worse again.

We should also note that the UNI3GMS and UNI4GMS experiments use the Web 1T 3- and 4-gram data, so the counts are larger and more reliable, and the chance of shared contexts is greater. This suggests that the impact of removing the left context is even greater than these results indicate.

We also considered the minimum number of contexts a term had to appear in to be included. Our previous experiments required two contexts – otherwise a term cannot be discovered. Increasing this cutoff to 10 made no significant difference.

The impact of context geometry on bigram accuracy is shown in Table 4. The penalty for removing some left context was not as great for bigrams, dropping from 68% with BI5GMS to 48% with BI4GMS. The remaining unigram experiments use UNI5GMS and the bigram experiments use BI5LEFT.

NAME	TEMPLATE
UNI5GMS	w_1 w_2 **X** w_3 w_4
UNI4LEFT	w_1 w_2 **X** w_3
UNI4RIGHT	w_1 **X** w_2 w_3
UNI3GMS	w_1 **X** w_2
BI5LEFT	w_1 w_2 **X X** w_3
BI5RIGHT	w_1 **X X** w_2 w_3
BI4GMS	w_1 **X X** w_2

Table 2: Unigram and bigram Web 1T templates.

	nS	nT	nC	Av(10)	Av(50)
UNI5GMS	5	5	10	90	**78**
UNI4LEFT	5	5	10	86	74
UNI4RIGHT	5	5	10	76	49
UNI3GMS	5	5	10	74	48

Table 3: Effect of context geometry on unigrams.

	nS	nT	nC	Av(10)	Av(50)
BI5LEFT	5	5	10	92	**68**
BI5RIGHT	5	5	10	83	51
BI4GMS	5	5	10	77	48

Table 4: Effects of context geometry on bigrams.

STOP	mean Av(50)	σ Av(50)
NO	69	2.9
YES	**75**	3.0

Table 5: Effects of category order on unigrams.

6.2 Class Ordering and Stop Classes

One criticism of the MEB algorithm is that it may be highly dependant on the order in which the classes are considered. Because of the mutual exclusion (i.e. once a word has been assigned to a particular class, it can't be assigned to any other class) the class order clearly has the potential to impact the results.

To test this we have run a set of ten unigram experiments with the classes arranged in random permutations. The results are shown in Table 5. The standard deviation of the ten sets is 3.0, around a mean of 75. This shows that although it has some impact, MEB is reasonably robust to changes in the category order. The standard deviation from these experiments can be used as an indication of the scatter across the MEB experiments in general.

We also compared the accuracy of using, and not using, the stop classes, which are used to constrain specific semantic drift problems (Curran et al., 2007). When the stop classes were used, they always appeared first in the same order, before the randomly permuted semantic classes. The difference in the means in Table 5 between the set with and without stop classes shows that using stop classes does improve the accuracy of MEB.

6.3 Number of Contexts

Our experiments with unigrams in Curran et al. (2007) showed that the best results were obtained by adding 10 contexts per iteration of the bootstrapping process. We have repeated this experiment for bigrams, with the results shown in Table 6.

These experiments show that that best results are obtained for numbers of contexts between 5 and 20. There is a significant drop-off in accuracy ($\sim 10-20\%$) for values of nC less than 5 or greater than 20. This demonstrates a preference for having more evidence for new terms being reliable than for simply adding more terms in each iteration. It also shows that keeping the number of terms added per iteration (nT) and the number of contexts (nC) added per iteration reasonably well balanced is the best strategy. This makes sense if we consider the extreme cases: adding 5 terms and only 1 new context per iteration would mean that it was difficult for the system to expand into new space; adding 5 terms and 100 new contexts per iteration would mean that many of the contexts may not be representative of the contexts that those 5 terms appear in.

nS	nT	nC	Av(10)	Av(50)
5	5	1	73	52
5	5	2	76	50
5	5	5	92	71
5	5	10	92	**73**
5	5	20	93	70
5	5	50	87	58
5	5	100	84	59

Table 6: Effects of changing the number of contexts added per iteration.

6.4 Filtering Using Collocations

One issue that arises when extracting bigrams (or longer n-grams) is the possibility that random combinations of tokens may be selected by chance in the MEB process. To investigate this we have carried out a series of experiments on data that was pre-filtered using collocation statistics.

We filtered the Web 1T data so that we only kept bigrams that were significant collocations based on their frequency in the Web 1T corpus. We chose the t-test as our measure of significance as it is simple to calculate and we do not have any low frequency values (< 5) for which the t-test is known to perform badly. Our calculation follows Manning and Schütze (1999, pg. 165). If $f(w)$ and $f(w_1,w_2)$ are the unigram and bigram frequencies from Web 1T, then the t-test is:

$$t = \frac{p(w_1,w_2) - p(w_1)p(w_2)}{\sqrt{\frac{p(w_1,w_2)}{N-1}}} \quad (1)$$

where N is the number of tokens. Using the Maximum Likelihood Estimates (MLE) we have:

$$p(w_1,w_2) = \frac{f(w_1,w_2)}{N-1} \quad (2)$$

$$p(w) = \frac{f(w)}{N} \quad (3)$$

The results for cutoffs at different significance levels are shown in Table 7. These experiments

t	nS	nT	nC	Av(10)	Av(50)
50	5	5	10	86	69
100	5	5	10	87	**70**
250	5	5	10	85	68
500	5	5	10	81	58

Table 7: Effects of using only significant collocations. A value of 100 in column 1 means that only bigrams with a significance of $t \geq 100$ were used.

show that the filtering had no statistically significant result on the accuracy of MEB. In a sense, this is not surprising, as the MEB process of ranking new terms on the number of contexts they occur in is already performing a form of significance testing.

However, filtering on collocations does have the advantage of significantly reducing the size of the vocabulary without a significant loss of accuracy at the Av(50) level. For example the number of unique bigram terms in the BI5LEFT experiments in previous sections is 1 858 097, compared to 482 053 for the $t \geq 100$ filtered subset ($\sim$ 25%) and 87 537 for the $t \geq 250$ filtered subset ($\sim$ 5%). This is particularly important when dealing with massive corpora.

6.5 Multi-word Expressions

Of greater interest than extracting unigrams or bigrams alone is the application of MEB to the general case of extracting n-grams of any length. Since the maximum length of the term and context in the Web 1T corpus is five tokens, and given the decline in accuracy that comes with reducing the length of the context (see Tables 3 and 4) it would be impractical to extract terms with more than two tokens.

Hence our final experiment with proper noun named entity extraction combines the unigram and bigram data together. This serves as an initial test of extracting multi-word expressions as it is not specific to only unigrams or only bigrams. The data consists of that used for UNI4LEFT and BI5LEFT, so that the context surrounding the unigram or bigram has the same length and geometry.

The categories we used for this experiment are those in Table 1 that are marked as suitable for both unigrams and bigrams. The results for this experiment are shown in Table 8. These are comparable to our best results for bigram extraction.

nS	nT	nC	Av(10)	Av(50)
5	5	10	84	69

Table 8: Results for extracting bi- and unigrams

	UNI5GMS	WordNet
terms	263 613	29 157
contexts	10 449 412	18 832 474
terms-contexts	42 039 483	88 178 856

Table 9: Comparison of the datasets used in the UNI5GMS and WordNet experiments. The number of unique terms, unique contexts and unique term-context combinations is shown.

7 WordNet Common Nouns

In our second set of experiments we investigate the application of MEB to common nouns. For these experiments we used the noun classes from WordNet, as described in the next section. We expected the performance on this task to be worse than for proper nouns for a number of reasons. Firstly, common nouns have a larger number of senses, on average, compared to proper nouns. This breaks the mutual exclusion assumption that is central to MEB's success. Secondly, common nouns are likely to occur in a wider range of contexts than many proper nouns. Thirdly, the common noun categories are more general and less well defined than for proper nouns, and abstract nouns are also likely to be harder to categorise than concrete nouns.

One factor that favours common noun extraction is that the WordNet classes are designed to have reasonably complete coverage of the semantic space. This is not the case in the BBN named entity categories, which is one of the reasons why we introduced stop classes (Curran et al., 2007).

Table 9 compares the size of the initial dataset for the UNI5GMS experiments (Section 6) and the WordNet common noun experiments. Even though we have $\sim$ 10 times fewer unique terms in the WordNet dataset, the number of unique term-context combinations is double that in the UNI5GMS dataset. The total size of the dataset used for the common noun experiments is 1.2GB.

7.1 WordNet Categories

For common nouns we used 25 noun categories from WordNet 3.0. These come from the broad seman-

CATEGORY	# WORDS	# UNI	# BI
act	6650	4917	1512
animal	7509	5010	2227
artifact	11587	7176	4163
attribute	3039	2646	322
body	2016	930	898
cognition	2964	2118	724
communication	5607	3788	1557
event	1074	844	216
feeling	428	396	31
food	2573	1347	1131
group	2624	1218	1012
location	3209	2272	788
motive	42	31	9
object	1545	1000	455
person	11087	9426	1516
phenomenon	641	332	285
plant	8030	4200	3382
possession	1061	492	514
process	770	594	162
quantity	1275	806	287
relation	437	266	150
shape	341	252	78
state	3544	2403	991
substance	2983	1869	1060
time	1028	574	375

Table 10: Noun categories in WordNet and the number of words, unigrams and bigrams in each.

tic classes employed by lexicographers in the initial phase of inserting words into the WORDNET hierarchy, called *lexicographer files* (*lex files*). For the noun hierarchy, there are 25 lex files and a file containing the top level nodes in the hierarchy called Tops. Lex files form a set of coarse-grained sense distinctions within WORDNET. These categories and the number of WordNet words in each category are shown in Table 7.1.

7.2 WordNet Evaluation

These experiments only involved unigrams seen in WordNet and hence we could evaluate directly against WordNet as a complete gold standard. We extracted the unigrams from all of the noun categories in WordNet. We then filtered the Web 1T corpus to extract only contexts where a WordNet unigram was the central token. The rest of the filtering, evaluation and scoring details follow the principles described in Section 5.

Each proposed term was marked as correct if it appeared in that WordNet semantic category. The advantage of a closed system is the ease of evaluating the results. However, an obvious disadvantage is that the system cannot be marked correct for valid unigrams it discovers in a category, that are not listed under that category in WordNet. A full manual evaluation may produce better results.

nS	nT	nC	Av(10)	Av(50)
5	5	10	29	22
10	5	10	51	43
20	5	10	67	52
100	5	10	**73**	**59**

Table 11: Effects of number of seed words.

8 WordNet Experiments

Creating seed lists using the Web 1T frequencies, as we had done in previous experiments, was complicated by skew towards web-related senses. For example, thumbnail was the 5th most frequent word in the body category and site was the 2nd most frequent word in the location category. In the number of seeds experiments we chose the seeds based on their frequency alone, but in the remaining experiments we manually created seed lists.

8.1 Number of Seed Words

We use the *n* most frequent words that were unique to each category as seeds, regardless of whether they have obvious web-related senses. The results for increasing the number of seed words are shown in Table 11. Note that the seed words are not included in the accuracy calculation. The limited number of terms in some categories (in particular, motive) causes a decrease in accuracy when more seeds are used because many of the correct proposed synonyms are now seed words.

There is a substantial increase in accuracy as the number of seeds is increased. This shows that even though the choice of seeds is far from optimal, and is strongly affected by interference, the results are still reasonable as long as a large number of seed words is used.

CATEGORY	Av(10)	Av(50)
animal	100	92
communication	100	94
food	100	96
location	100	98
cognition	0	12
feeling	60	24
object	40	20
relation	60	22
Mean	**62**	**44**

Table 12: Results for a selection of high and low performing common noun categories. The mean was calculated across all the semantic classes. The other parameters were (nS, nT, nC) = (5, 5, 10).

8.2 Comparison of Semantic Classes

To compare performance across semantic classes, we manually selected 5 seed words from the 20 most frequent words in each category (as measured in the Web 1T corpus). This allowed us to excluded words which we knew to have web-related senses that would dominate on the Web 1T data.

The accuracy obtained was 44%, which is substantially lower than for the named entity unigram experiments (maximum 78%). However, the variation in performance across the categories was extremely high, as demonstrated in Table 12. Some categories, such as cognition are extremely difficult.

This demonstrates that MEB is very good at extracting certain kinds of lexical semantic knowledge – primarily for categories that are very well defined, with frequent terms that appear in fairly constrained or idiomatic contexts, for example animals and food. For these categories, MEB performed just as well on common nouns as it did on many of the proper noun named entity categories.

Conclusions

We have presented two novel applications of Mutual Exclusion Bootstrapping (MEB): extracting bigram named entities and common nouns from WordNet. We confirmed that MEB is sensitive to the geometry of the context window surrounding the extracted terms. As expected, a larger context leads to higher accuracy, but interestingly, this is almost entirely due to extra context on the left of the target term. Overall, this makes bigram and longer n-gram extraction more difficult on fixed-sized window data, such as the Web 1T corpus.

Surprisingly, we discovered that MEB is relatively insensitive to the order the semantic classes are presented and to noise in the possible terms themselves.

We applied MEB to common nouns using 25 semantic classes defined by the WordNet lexical files. We performed automatic evaluation using a closed vocabulary and found that MEB performed well on classes with narrower definitions such as animal, but poorly on classes such as cognition. This is partly due to the concrete categories having more coherent contexts. We found that increasing the number of seed terms improved the accuracy, even with poor quality seed terms.

We now plan to experiment with loosening the mutual exclusion assumption to allow for some overlap between categories. There are many possibilities for improving the performance of MEB on common nouns – here we have presented only a preliminary analysis of the WordNet results. We also plan to experiment with text other than the Web 1T corpus so that we can test whether allowing wider contexts will further improve performance.

The experiments we have presented in this paper have demonstrated that MEB is an efficient and accurate method of extracting semantic classes over both unigram and bigram named entities. We have also demonstrated its potential for extracting semantic classes from WordNet for common nouns.

Acknowledgements

Both authors were funded on this work under ARC Discovery grants DP0453131 and DP0665973. We would like to thank the anonymous reviewers for their useful feedback.

References

Eugene Agichtein, Eleazar Eskin, and Luis Gravano. 2000. Combining strategies for extracting relations from text collections. Technical Report CUCS-006-00, Department of Computer Science, Columbia University, New York, March.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Technical Report LDC2006T13, Linguistic Data Consortium.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pages 172–180, Melbourne, Australia, 19–21 September.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Cristiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. The MIT Press, Cambridge, MA USA.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th international conference on Computational Linguistics*, pages 539–545, Nantes, France, 23–28 July.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479, Orlando, FL USA, 18–22 July.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124, Providence, 1–2 August.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistic for semi-automatic semantic lexicon construction. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th annual meeting of the Association for Computational Linguistics*, pages 1110–1116, Montréal, Québec, Canada, 10–14 August.

Neel Sundaresan and Jeonghee Yi. 2000. Mining the web for relations. In *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands, 15–19 May.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical Report LDC2005T33, Linguistic Data Consortium.

Proceedings of the Australasian Language Technology Workshop 2007, pages 75-82

Charting Democracy Across Parsers

Scott Nowson[1] **and Robert Dale**
Centre for Language Technology
Macquarie University
Sydney, Australia
{snowson|rdale}@ics.mq.edu.au

Abstract

Different parsers trained on the same corpus deliver different results, both in terms of overall performance and in terms of the individual analyses they provide. In particular, for any given sentence, one parser may provide a correct analysis, while another will produce an incorrect analysis; but when faced with a different sentence, the first parser may be in error while the second is correct. In this paper, we leverage this observation by exploring how the results of a number of different parsers may be combined to provide a better performance than any single parser. The method involves constructing a chart that contains edges contributed by a collection of parsers, with a simple voting mechanism to choose the most preferred constituents; this provides a significant improvement in performance over any individual parser. More sophisticated voting mechanisms are also discussed.

1 Introduction

Parsers make mistakes. This is perhaps most apparent when a parser trained on a given corpus is applied to data from a domain or genre different to that of the training corpus. One can, of course, retrain the parser on new data that is more representative of the texts to be handled; but annotation is an expensive process, and the literature does not provide a great deal of guidance as to how much annotation is required in order to obtain an acceptable result (but see Reichart and Rappoport (2007a) for some recent interesting results in this area).

Unfortunately, parsers make mistakes even on the corpora on which they are trained. Before we begin to consider how we might adapt a parser to a new domain, we are therefore interested in how we might improve the performance of existing parsers on the corpora used to derive their models.

We make the observation that different parsers have different 'error profiles', by which we mean that different parsers do not necessarily make the same mistakes. Consider the following verb phrase taken from our test corpus:

> *. . . lock in profits by buying futures when futures prices fall*

Figure 1 shows the analyses provided for this verb phrase by three different parsers, as an illustration of the kinds of disagreements that are common. In the first analysis, *in* is misclassified as a preposition, while in the second and third analyses it is correctly analysed as a particle. However, the second parse contains a misparse of the embedded VP *buying futures*, while this is correctly analysed in the first and third parses.

This leads us to the hypothesis that, if we were able to select for each parser those parts of individual parses that are more likely to be correct, then the overall result would be an improvement upon the analysis of any individual parser. We explore this hypothesis in this paper, by providing a framework within which the analyses of different parsers can be combined, and the overall best parse selected.

[1] Scott Nowson is now at Appen Pty Ltd.

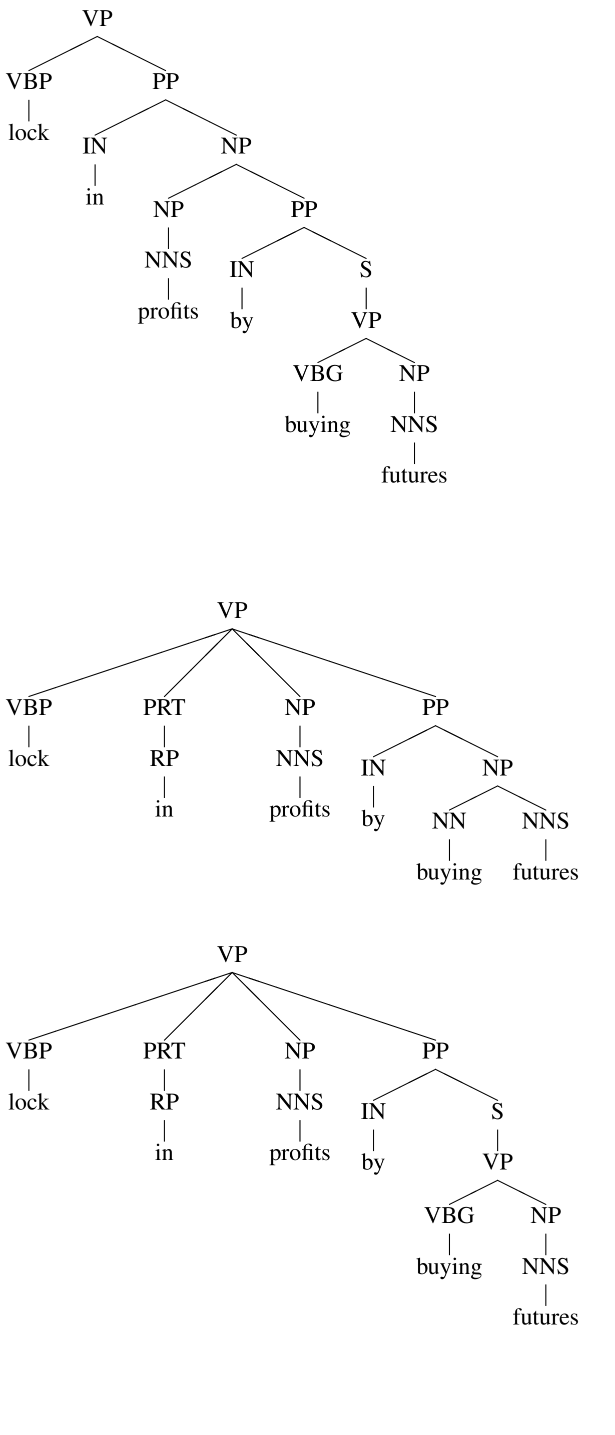


Figure 1: Three analyses by different parsers

The idea of combining the results of different parsers is not in itself new, so in Section 2 we briefly survey related work in this area. In Section 3 we describe our approach, which takes advantage of the central ideas in chart parsing to provide a way of combining parse results, and we describe the parsers used in our experiments. Section 4 describes the results achieved by our method, demonstrating a significant improvement upon the performance achieved by any individual parser. Section 5 discusses how the simple voting mechanism presented here can be made more elaborate, with the prospect of even better improvements in performance, and Section 6 concludes.

2 Background: Combining Parsers

The combination of the results of several different components that carry out the same task—sometimes referred to as the ensemble-based approach—has been employed and shown to be successful in a number of fields such as part-of-speech tagging (Halteren et al., 1998), word sense disambiguation (Pederson, 2000) and question answering (Chu-Carroll et al., 2003).

There are a number of approaches that have been employed for parser combination. Henderson and Brill (1999) describe experiments that fall within two general approaches they label *parse hybridization* and *parse switching*. The most basic form of hybridization is constituent voting, whereby constituents in a parse are included if they can be found in the majority of contributing parses. A second approach is to use a naïve Bayes classifier in order to learn how much each parser should be trusted.

The alternative to this approach is to deal only with complete parses. Henderson and Brill again experimented with two approaches: *similarity switching*, whereby the parse chosen is the one which scores highest when judged for similarity to the remaining parses in the set; and a second naïve Bayes approach to selecting the parse with the highest probability of being the best. All four of Henderson and Brill's approaches produced better results than any of the contributing parsers achieved: their best result was a 30% reduction of precision error rate, a 6% reduction of recall error rate and an absolute F-score increase of 1.58%. These ideas have been extended to take into account more context in adaptation to dependency based parsers with similarly successful results (Zeman and Žabokrtský, 2005).

Henderson and Brill (2000) followed their earlier combination approach with one based on creating an ensemble of complementary parsers. Each parser was based upon the same underlying algorithm, but trained on different data. By using bagging and boosting approaches, their ensemble outperformed all single parsers, with a 0.6% absolute improvement in F-score. In a similar vein, Reichart and Rappoport (2007b) generated a number of parsing models by training one parser on slightly different training corpora. The resulting outputs were compared in order to judge the parse quality: the greater the number of models in agreement, the higher the quality.

Clegg and Shepherd (2005) explored a number of alternative approaches to ensemble parsing when deploying trained parsers in a new domain. Using basic constituent voting based on Brill and Henderson's (1999) method, they report similar improvements, mostly to precision but also to recall. They achieved equally promising results from their variants of parse switching. The first of these was *fall-back cascades* in which parsers are stacked in order of decreasing levels of sophistication. When the more complex model fails, the next parser attempts to parse. The bottom parser may be less accurate, but will be the least likely to fail. Their second whole-parse approach they simply termed *parse selection*, though it is similar to Henderson and Brill's similarity switching. Clegg and Shepherd varied this by trying different similarity metrics, such as constituent overlap or lineage similarity.

Sagae and Lavie (2006) apply a notion of reparsing to a two stage parser combination chart-based approach. Once all single parses are complete, the first stage is to store all possible constituents in a chart with a label, start and end positions, and a weighting. Identical constituents from different parses are merged by adding their weights. The second stage of the process is to run a bottom-up parsing algorithm, but rather than use a weighted grammar, the parser is guided by the weighted set of constituents. They experimented with different approaches to setting the initial weights of each non-terminal label. By combining five parsers they were able to achieve a error reduction of 44% for precision and 14% for recall, and an absolute F-score increase of 1.1% (though it is worth noting each of these are *best* improvements, made across a different run made with different settings).

3 Our Approach

3.1 The Basic Idea

Our approach is based on the central idea in chart parsing (Earley 1970; Kay 1980): for any ambiguous string, the constituents derived from multiple parses can be maintained in one data structure, so that subsequent parses can reuse previously derived partial analyses. The insight leveraged here is that the same idea can be applied to multiple parsers: just as the chart can contain multiple analyses for a string as delivered by one parser, it can just as easily contain multiple analyses delivered by several parsers, thus providing a single unified view of all the different analyses, and allowing us to easily determine where parsers agree and where they disagree.

This is a very simple idea, but one which enables the development of a variety of approaches to choosing which edges should be used in building a preferred parse; and, as we demonstrate below, even the simplest methods provide good results.

Our approach to combination is built upon a basic voting strategy, methodologically similar to Henderson and Brill (1999) and Sagae and Lavie (2006), with implementational similarity to the latter. In its purest form, voting is purely democratic: all nominated constituents are considered equally suitable candidates to fill a position in the parse, and the candidate with the most votes—i.e., the candidate proposed by a majority of the parsers—is the winner. The algorithm is simple:

1. Each sentence is parsed by multiple parsers.
2. The resulting Penn Treebank parse strings are converted into a chart representation.
3. Starting with the root node, voting takes place as to what the children of that node should be.
4. Step 3 is then repeated for each successful child node.
5. When the tree is fully populated by terminal nodes, the final chart is returned as a Penn Treebank parse representation for evaluation.

We now describe these stages in more detail.

3.2 Parsing

The first stage in our process is to parse each sentence with the individual contributing parsers. In the experiments reported here, we use three parsers: the Stanford lexicalised parser (Klein and Manning, 2003); Collins generative parsing model number 2 (Collins, 1999) as re-implemented by Bikel (2004); and the OpenNLP parser (Baldridge et al., 2003). These were chosen for two reasons:

- all three parsers output parses in the standard Penn Treebank notation, making conversion to our chart representation the same process for all; and
- all three are provided with Java API functionality making incorporation into one system more straightforward.

This latter advantage also reduces computation times by enabling just a single parser initialisation step before parsing all sentences.

3.3 Chart Representation

Once each sentence has been parsed, the resulting Penn Treebank parse strings are converted into charts. In the standard approach, a chart is a collection of vertices that span sequences of one or more words of the input, with pairs of vertices that are connected by grammatically labeled edges; where a sequence of words is amenable to more than one analysis, each analysis is represented by a separate edge. Subsequent decisions, or some other choice mechanism, then determine which of the multiple analyses should be chosen.

In the implementation used here, vertices are defined in terms of character positions, while edges are defined by the start and end positions and given a grammatical label. As a step towards efficient implementation, each edge contributed by a given parser also indicates the constituent edges—the grammatical children—contained within the span of that edge. By example, consider the sentence *the cat sat*, which is analysed as follows:

(S (NP (DT the) (NN cat)) (VP (VBD sat)))

and is spanned thus:

```
 t h e   c a t   s a t
0 1 2 3 4 5 6 7 8 9 10 11
```

This analysis would be represented as a chart consisting of edges:

```
(0, 12, S, {(0,7,NP), (8,11,VP)}}
(0, 7, NP, {(0,3,DT), (4,7,NN)})
(0, 3, DT, {(0,3,the)})
(4, 7, NN, {(4,7,cat)})
...
```

3.4 Chart Voting

The fundamental difference between our approach and those of Henderson and Brill (1999) and Sagae and Lavie (2006) described earlier is in the strategy used when selecting constituents. Previous approaches have considered constituents in isolation: Sagae and Lavie's charts contain all possible constituents, each assigned a weight based on their presence across individual parsers, and these are merely used to inform a second stage, bottom-up reparsing. By comparison, our system could be described as a single-stage, top-down process which operates across the prior parses. Similar to Sagae and Lavie, we employ simple voting to determine the choice of constituents, but we consider only the nominated children of each already-decided constituent. Since each such set of children corresponds to a valid parse, we can ensure there will not be any crossing brackets, and that the resulting parse will be grammatically sound.

Each grammatical constituent is defined by an edge within the chart. For each edge that covers the same span of words[1] across the individual parser outputs, the set of potential analyses can be retrieved. Each such solution provides a potential constituent analysis with which to continue the parse, and for whom votes can be tallied. So, in a democratic manner, any child nominated by all contributing parsers is unanimously voted into the parse. Similarly, any constituents that obtain a majority vote also succeed. In the case of a tie, we resort to arbitrarily choosing between the potential solutions. The only restriction is that children are chosen so that the entire span is accounted for and a complete tree is created.

[1]Matching edges are defined by the tuple `<start pos, end pos, label>`.

Of course pure democracy, at least in the case of parser combination, is quite naïve. It treats all candidates as equal and does not take past performance of parsers into consideration; nor does it take into account the possibility that some parsers may perform better in specific situations. Clearly a sensible step forward here is to move towards a more meritocratic approach, as discussed in Section 5 below.

3.5 An Example

As introduced earlier, Figure 1 illustrates how three different parsers can construct parses that differ or are similar in different ways. In this section, we walk through the combination of these parses to provide an example of how our approach works. To save space and to aid clarity, we represent the span of any node simply by listing the words contained in that span.

In our example, we begin part-way through the parse, where the the current node of interest is the VP which spans from *lock* to *futures*. The analyses of this node are retrieved from the charts delivered by the three parsers, and votes are calculated across the children:

VBP (‘lock’)	3 votes
PRT (‘in’)	2 votes
NP (‘profits’)	2 votes
PP (‘by . . . futures’)	2 votes
PP (’in . . . futures’)	1 vote

Three votes represents a unanimous decision, while two is a majority; so, the decomposition of the VP node that is common to the second and third analyses is chosen.

Note that the PRT daughter of the VP node receives two votes in total, and subsequently so in turn does it’s daughter, the RP. However, though the NP node also received just two votes, the NNS node at the next level of analysis receives three votes. This is because in the case of one of these analyses the NP node is buried deeper in the tree. Similarly, though the PP *in buying futures* was voted twice for its position in the tree, it can be found in all three parses at some level.

However, we have a disagreement as to the decomposition of the PP:

IN (‘by’)	3 votes
S (‘buying futures’)	2 votes
NP (‘buying futures’)	1 vote

Parser	P	R	F
Stanford	87.0	85.7	86.4
OpenNLP	88.1	87.7	87.9
Collins	72.9	88.9	80.1
Combined	90.7	89.5	90.1

Table 1: **P**recision, **R**ecall and **F**-score for individual parsers and their combination; sentence length $<=$ 40 words ($n = 2245$).

Parser	P	R	F
Stanford	86.4	85.0	85.7
OpenNLP	87.4	87.0	87.2
Collins	72.7	88.3	79.7
Combined	90.2	88.9	89.5

Table 2: **P**recision, **R**ecall and **F**-score for individual parsers and their combination; all sentences ($n = 2416$).

Consequently, the chosen analysis of the PP is that proposed in the first and third trees.

3.6 Evaluation

We evaluate the results of our approach using the PARSEVAL standard Evalb (Sekine et al., 2006). The input to the system is Section 23 of the Wall Street Journal (WSJ). All sentences are pre-tokenised to ensure standard input, though each parser executes its own part-of-speech tagging. The system outputs four sets of parse strings: one for each of the three constituent parsers, and one for the final combined result. The sets of parses are compared against the gold standard.

4 Results

We report the bracketing precision, recall and F-score for sentences of length less than 40 words in Table 1, and for all sentences in Table 2.

It is clear that the combined system performs the best. Considering all sentences, we have achieved an error reduction of 22% for precision and 5% for recall, along with an absolute F-score increase of 2.3% over the best single contributor. In order to compare our results with those of previous studies, we reproduce the results of Henderson and Brill (1999) and Sagae and Lavie (2006) alongside our own in Table 3. Our results are comparable directly with those of Henderson and Brill; Sagae and Lavie's scores are a compilation of their best scores across three separate systems tuned to maximise each dimension, hence the high increase in precision and recall.

As a further investigation, we employed a simple measure of confidence in a parse as a function of the number of parsers in the system, the total number of edges in the final chart and the total number of votes cast over just those successful edges:

$$\text{confidence} = \frac{\sum \text{votes}}{\sum \text{edges} \times \sum \text{parsers}}$$

Confidence will be highest if all the parsers agreed on each edge (had the same parse throughout) and will be lower the less they agree. Average confidence across our output is 0.88, which suggests that overall there was a high degree of agreement across parsers. The confidence measure also shows a significant correlation ($p < .001$) with the precision and recall scores across all sentences. This suggests that the system is most likely to be wrong when it is least confident in its output, and so the confidence metric is a good one.

5 Discussion

The performance values reported in Tables 1 and 2 show that the combined system produces more accurate results than the original individual parsers, as we had hoped. By simply taking a majority vote on constituents, our system results in more correct constituent analyses than those proposed by the individual parsers. However, the combined result is not a huge improvement over the highest performing of its contributing proposals.

It is of course possible that for the most part, all parsers get the same things wrong — the rare and infrequent syntactic constructions. This would present a simple voting system with no way to select the correct analysis. However, it is likely that systems that get the same things wrong do so in the same way. Such agreement on incorrectness still represents an agreement, which would provide a high level of confidence in the incorrect choice. However, looking at our confidence scores, this incorrect agreement does not appear to be the case: errors appear to follow a lack of confidence — where there is most disagreement.

The biggest weakness in our approach lies in the arbitrary decision-making procedure used in breaking tied situations. In such tied situations, if the wrong result is chosen, then all the constituent analyses below that point have a high likelihood of being incorrect. This is a particular weakness of our top-down approach, in contrast to Sagae and Lavie's bottom-up method.

There are a variety of ways in which the basic model developed here could be extended. Of course, one could extend the mechanism beyond the three parsers that we use to incorporate a larger number of parsers. However, a more interesting direction is to improve the voting mechanism. The greatest number of errors appears to stem from situations of low agreement, when voting is tied.[2]

One approach to resolving deadlocked situations such as these might be to employ a lookahead approach. As illustrated in our example in section 3.5 upon voting across the top level VP, the NP receives two votes. However, this is only because it was directly under the VP in two cases; the NP was in fact still present in the third analysis, but buried further down in the tree. In a tied situation, this fact would have argued for one analysis over the other.

Another approach is to observe that, while democracy is fair other things being equal, parsers are more akin to experts to be consulted. For example, we might think of each parser as having particular areas of expertise, in the sense that its performance on some kinds of constituents might be better than others. If a given parser has a track record of performing well in the analysis of particular kinds of constituents or substructures, then that parser's vote should carry more weight.

There are a number of approaches we might take to developing a more meritocratic decision procedure.

Track Record on This Parse: This is a general if superficial measure of performance. It assumes no external knowledge, and all parsers begin with an equal weighting. Weightings are increased automatically for every successful vote that a parser casts. If all parsers always agree, weights will remain equal. The more others in

[2]Note that these situations are even more likely to occur if the system were to employ an even number of parsers.

	P	*% decrease*	R	*% decrease*	F	*% increase*
Henderson and Brill, best individual	89.6		89.7		89.7	
Henderson and Brill, combination	92.4	*26.9*	90.1	*3.9*	91.3	*1.6*
Sagae and Lavie, best individual	91.3		90.6		91.0	
Sagae and Lavie, combination	95.1	*43.7*	91.9	*13.8*	92.1	*1.1*
Nowson and Dale, best individual	87.5		88.3		87.2	
Nowson and Dale, combination	90.2	*21.6*	88.9	*5.1*	89.5	*2.3*

Table 3: **P**recision, **R**ecall and **F**-score for parsers; The best individual parser from each study, plus the best combined results, and the differences between them.

the system agree with a parser, the more popular it becomes, and the heavier its weighting. The downside, however, is that should one system perform well early on, its weighting may be so much that a local maxima may be reached.

Previous Track Record: This is also a general measure of performance, but is static and relies on external information. Weightings are set based on the prior performance of a parser: those that have previously produced most accurate results will be trusted more and weighted higher. One source for this data would be previously published, preferably comparable, results. However, as we noted at the start of the paper, good performance in one domain or genre does not guarantee similar results in another.

Two other measures, as suggested by Henderson and Brill (1999), take context into account:

Constituent-Level Track Record: The previous approach gives higher weighting to the parsers that have previously performed best overall, but this does not mean they were the best at everything. In this approach, we narrow the focus to performance over individual constituent types: higher weighting is given to a parser's vote, if upon prior evaluation it has proven successful at selecting the specific nominated constituent. The prerequisite to this is that performance analysis must have been carried out at the level of individual constituents. Alternatives might include using machine learning techniques to automatically determine which parsers do best in which situations.

Structural-Level Track Record: The approach above could be further extended to take account of a larger amount of syntactic context; for example, it might be the case that some parers are better at subject NPs but less good at object NPs. Here we would need to compute weights based on past performance on correct annotation of subtrees in an analysis; clearly this could be done at varying levels of granularity, modulo the problem of sparse data.

6 Conclusion

This paper reports work concerned with combining parsers using a chart based representation and voting scheme. It has introduced the methodology we will employ in our future parsing work: the outputs from multiple parsers are transformed into a chart representation; by voting over children these charts are combined into a single chart combining those constituents for which there is the strongest evidence.

The combination process pursued here is based on the simplest interpretation of evidence, where we pursue a purely democratic approach. This approach is most obviously deficient when we have to deal with ties. Nonetheless, the resulting parses prove more accurate than the single nominees that contributed to their creation, and performance compares well to previous studies that employ more complex and sophisticated methods. This suggests our approach has considerable scope for subsequent improvement, some possible directions for which we have outlined in the latter part of this paper.

Acknowledgments

The authors acknowledge the financial support of the Capital Markets Cooperative Research Centre.

References

Jason Baldridge, Tom Morton, and Gann Bierner 2003. *OpenNLP toolkit*. Available from http://opennlp.sourceforge.net/

Daniel M. Bikel. 2004. Distributional Analysis of a Lexicalized Statistical Parsing Model. *Procedings of EMNLP 2004*, NJ.

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager and Abraham Ittycheriah 2003. In Question Answering, Two Heads Are Better Than One. *Proceedings of HLT-NAACL 2003*, 24–31.

Andrew B. Clegg and Adrian J. Shepherd. 2005. Evaluating and integrating treebank parsers on a biomedical corpus. *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD Dissertation, University of Pennsylvania.

J. Early. 1986. An efficient context-free parsing algorithm. In B. J. Grosz, K. Sparck-Jones & B. L. Webber (Eds), *Readings in natural language processing*, 25-33. Morgan Kaufmann, San Francisco, CA.

Hans van Halteren, Jakub Zavrel and Walter Daelemans. 1998 Improving data driven wordclass tagging by system combination. *Proceedings of the 17th International Conference on Computational Linguistics*, 491–497.

John C. Henderson and Eric Brill. 1999. Exploiting Diversity in Natural Language Processing: Combining Parsers. *Proceedings of EMNLP 1999*, 187–194.

John C. Henderson and Eric Brill. 2000. Bagging and Boosting a Treebank Parser. *Proceedings of NAACL 2000*, 34–41.

M. Kay. 1986. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. Sparck-Jones & B. L. Webber (Eds), *Readings in natural language processing*, 35-70. Morgan Kaufmann, San Francisco, CA.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Ted Pedersen. 2000. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambigusation. *Proceedings of NAACL 2000*, 63–69.

Roi Reichart and Ari Rappoport. 2007. Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, 616–623.

Roi Reichart and Ari Rappoport. 2007. An Ensemble Method for Selection of High Quality Parses. *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, 408–415.

Kenji Sagae and Alon Lavie. 2006. Parser Combination by Reparsing. *Proceedings of HLT-NAACL 2006*, 129–133.

Satoshi Sekine, Michael J. Collins, David Brooks and David Ellis 2006. *Evalb*. Available from http://nlp.cs.nyu.edu/evalb/

Daniel Zeman and Zdeněk Žabokrtský 2006. Improving Parsing Accuracy by Combining Diverse Dependency Parser. *Proceedings of the Ninth International Workshop on Parsing Technology*, 171–178.

Proceedings of the Australasian Language Technology Workshop 2007, pages 83-91

Two-Step Comprehensive Open Domain Text Annotation with Frame Semantics

Bahadorreza Ofoghi
Centre for Informatics and Applied Optimization, University of Ballarat
bofoghi@students.ballarat.edu.au

John Yearwood
Centre for Informatics and Applied Optimization, University of Ballarat
j.yearwood@ballarat.edu.au

Liping Ma
Centre for Informatics and Applied Optimization, University of Ballarat
l.ma@ballarat.edu.au

Abstract

With shallow semantic parsing tasks receiving more attention in many natural language applications, there is a need for labeled corpora for learning the specific tags under consideration. In this paper, we discuss a two-step semantic class and semantic role assignment based on the FrameNet elements over a subset of the AQUAINT collection with a reasonable coverage over the semantic frames in FrameNet. The quality of the annotation task is examined through inter-annotator agreement. The methodology described in this work for measuring inter-annotator agreement can be adapted for similar tasks. Some central aspects of the task are also detailed in this paper.

1 Introduction

Open domain text labeling with the tags of lexical semantic structures is a time consuming intensive task that is necessary to initiate the semi-(or fully-) automated process of text annotation with lexical semantic information. The extra knowledge that such labeling can contribute to texts develops a higher level of text understanding for high level applications in the domain of natural language processing e.g. question answering (QA), information extraction (IE), machine translation, and etc.

The learning parsers which can afford the automated semantic labeling of texts are highly dependent on the training sets with example annotated texts. The existing semantic parsers, better known in this context as *shallow semantic parsers*, rely on the syntactic and/or semantic similarities of the training sets and the test sentences (syntactical and semantic features) (Erk and Pado 2005) (Pradhan, Ward et al. 2004). The SHALMANESER parser (Erk and Pado 2006), as one of such parsers, benefits from two classifiers to add the FrameNet (Baker, Fillmore et al. 1998) Frames and Frame Elements (FEs) to open domain texts. It is mostly dependent on the syntactic features in the texts. The ASSERT shallow semantic parser (Pradhan, Ward et al. 2004) exploits the SVM classifiers trained on both syntactical and semantic features of texts to assign semantic roles to sentence elements (i.e. arguments).

One of the first issues that highly affect the accuracy of shallow semantic parsers is the quality of the training set. To avoid any unnecessary behavioral bias towards the annotation applied by a coder in the training set, it is prudent that the training annotation passes a reasonable inter-annotator agreement measure with respect to the different aspects of the labeling task.

In annotating a text collection with Frame Semantics (Fillmore 1976), the two main aspects are considered to be the frame and FE assignment to the text. While the first aspect is more a matter of word sense disambiguation (Erk 2004), the latter is a challenge with syntactic and semantic analysis of the constituents of a sentence and boundary detection for each argument of a predicate to be assigned to a semantic role (i.e. an FE).

In this paper, we describe the activity of annotating a subset of the AQUAINT collection with the

FrameNet frames and their corresponding FEs. The three major motivations of this task are as follows:

1- The labeled corpus can be exploited for the purpose of training shallow semantic parsers as it contains appropriate semantic class and role assignment to the text,
2- The methodology of measuring the inter-annotator agreement with regard to the tasks of frame assignment and FE labeling has not been tackled and formulated comprehensively before and our methodology can be adapted for similar tasks,
3- As the corpus contains the answer passages for a large amount of the TREC 2004 factoid question set (Voorhees 2004), it can be used for answer extraction technique learning and articulating the FrameNet-based answer candidate identification.

The task of adding frames and FEs to the texts has been performed in the two steps of *automated annotation* and *human expert augmentation*.

This paper is organized as follows. A brief definition of frame semantics in section 2 is followed by the description of the FrameNet-based text labeling in section 3. Section 4 analyzes the different aspects of our annotation task and section 5 concludes the paper.

2 Frame Semantics

Frame Semantics, basically developed from Charles Fillmore's Case Structure Grammar (Fillmore 1968) (Cook 1989), emphasizes the continuities between language and human experience (Fillmore 1976) (Lowe, Baker et al. 1997) (Petruck 1996). The main idea behind frame semantics is that the meaning of a single word is dependent to the essential knowledge related to that word. With such an understanding of frame semantics, the required knowledge about each single word is stored in a *frame*. In order to encapsulate frame semantics in such frames, the FrameNet project (Baker, Fillmore et al. 1998) has been developing a network of inter-related frames which is a lexical resource for English now used in many natural language applications.

The main entity in FrameNet is the frame which develops a kind of semantic normalization over concepts semantically related to each other. The semantic relation between concepts in a frame is realized with regard to the scenario of a real situation which may happen and cover the participant concepts rather than synonymy or other peer-to-peer relations. In this regard, the frames encode the base definitions necessary to understand the semantics and the scene of each contained term. In other words, real-world knowledge about real scenarios and their related properties are encoded in the frames (Lowe, Baker et al. 1997). Each frame contains some *frame elements* (FEs) as representatives of different semantic and syntactic roles regarding a target word inside the frame. The semantic roles are common properties among all of the terms that are inherited from a frame. This ensures a suitable inclusion over the English terms which either have similar meanings or share the context and/or the scenario in which they could occur in the sentences of the language.

The current version of the FrameNet data (release 1.3) contains about 795 semantic frames and more than 135,000 annotated sentences from the British National Corpus.

3 FrameNet-based Text Annotation

The task of shallow semantic parsing of sentences with respect to the predicates (e.g. verbs, nouns, etc.) mainly consists of two phases: i) sense disambiguation of the predicate, and ii) role assignment to the arguments of the predicate with regard to the specific sense (i.e. class) of it (Erk and Pado 2006). In the context of FrameNet, the class is realized as the specific frame which is evoked in the true sense of the context of the sentence (Erk 2004), while the roles are the different FEs in that frame.

Frame: Manufacturing		
Definition	A **Manufacturer** *produces* a **Product** from **Resource** for commercial purposes.	
FEs	FACTORY	Thos machines were *manufactured* **in the Miami plant**.
	MANUFACTURER	**General Electric** *produces* electric appliances.
	PRODUCT	The company *manufactured* **many T-shirts**.
Lexical units	fabricate.v, fabrication.n, industrial.a, make.v, maker.n, …, production.n	

Table 1. An example frame in FrameNet

Table 1 shows an example frame "Manufacturing" with its definition, core FEs, and lexical units. The main semantic roles that are necessary for the scenario to be complete are those known as the core FEs Factory, Manufacturer, and Product. Different predicates with different parts-of-speech (e.g.

noun, verb, and adjective) are inherited form this frame.

With respect to this frame, the annotation of an example sentence "*the company makes different types of doors in the Australia plant*" with respect to the predicate "make.v" consists of two stages: i) to identify the right semantic class (i.e. frame), and ii) to assign the different parts of the sentence to the semantic roles (i.e. FEs of the frame). Figure 1 visualizes this annotation task. There are different semantic classes that can be realized with the predicate "make.v" like arriving, building, cooking-creation, causation, manufacturing, etc. The task of finding the right frame from this set of related semantic classes is a problem formulated as word sense disambiguation in (Erk 2004). Having the right frame identified, the semantic role assignment to connect the FEs and the sentence constituents is the next challenge. As already mentioned, there are different studies which attack the problem using syntactic and/or semantic features in the text.

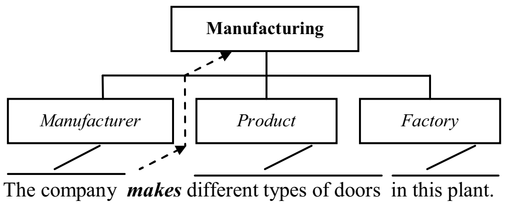


Figure 1. Shallow semantic analysis of an example sentence

The semantic annotation of the sentence in Figure 1, however, is not complete with regard to all of the existing predicates in the sentence. There are other frame-evoking elements: "company", "different", "types", "doors", and "plant" each of which can evoke at least a single frame in FrameNet. Figure 2 shows a complete annotation of frame evocation.

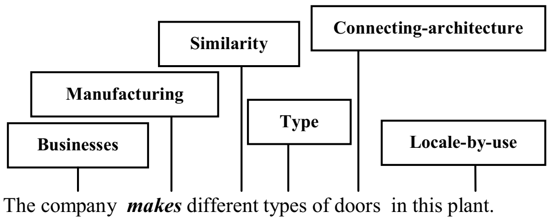


Figure 2. Complete frame annotation of an example sentence

The semantic annotations with respect to the FEs are eliminated in the figure above for readability, though in a complete annotation they need to be properly assigned to the corresponding sentence segments.

There are more than 135,000 annotated sentences in the FrameNet database. However, the standard of the annotations is different from that in Figure 2. The annotations in this FrameNet database are frame-oriented which result in annotating sentences per predicate inherited from the current frame. Consequently, there is not more than one frame evoked per example sentence.

In the task of automated shallow semantic parsing the training phase is vital. Most existing shallow semantic parsers are trained with the data provided in the FrameNet database. We believe that there are two main drawbacks on this training paradigm:

1- The concurrence of the frames is ignored as a meta semantic feature that could potentially be informative in correct semantic class identification,
2- The example sentences annotated per frame are semantically focused on the class covered by the frame.

The former could easily affect the context analysis of the sentences as a potentially useful feature that has to be eliminated in the training phase with the current FrameNet annotations. The latter produces bias in the classifiers which are supposed to be effective on the open domain text with lots of concurrent semantic scenarios.

We have been providing a comprehensive annotation with all of the frames and FEs evoked in an exhaustive manner to overcome the two above-mentioned shortcomings.

4 Comprehensive Text Annotation

With the first aim of training a frame-semantics-based answer extraction module for QA, we have comprehensively annotated a text collection. The output, which will be publicly available for research purposes, can be exploited for other natural language learner systems. The different aspects of the annotation task are explained in the following sub-sections.

4.1 Data

A subset of the AQUAINT text collection[1] which contains the news articles from the New York Times News Service (1998-2000), Xinhua News Service (1996-2000), and Associated Press Worldstream News Service (1998-2000) has been annotated using the method explained in section 4.2. The statistical information about the annotated corpus is summarized in Table 2.

Element	Measure
# Passage	1379
# Sentence	3451
Ave. # sentences per passage	2.502
# Single word	89434
Ave. # single words per sentence	25.915
# Single word (unique)	9291
# Predicate	53215
# Predicate (unique)	8121

Table 2. Annotated corpus statistics

The 1379 passages have been extracted in response to the information request of a subset of the TREC 2004 factoid questions including 143 questions (out of the entire set of 230 factoid questions) for which the retrieval systems retrieves passages actually containing the correct answers. The limitation for the task of passage retrieval was set to retrieve the top 10 passages per question. For a few questions, the retrieval system could not retrieve exactly 10 passages (in some occasions less number of passages) as there was not enough information text in the collection specifically related to the question. The modified version of the MultiText passage retrieval algorithm (Ofoghi, Yearwood et al. 2006) has been used for this purpose which interprets the passages as variable-sized strings starting and ending with pairs of query keywords at any position in the corpus documents.

4.2 Method

The annotation task of the corpus has been performed in a two-step process including *automated annotation* and *manual augmentation*.

In the automated annotation, the SHALMANESER shallow semantic parser (Erk and Pado 2006) version 1.0 has been used. The instance of SHALMANESER used in the task, benefits from the two learner classifiers FRED and ROSY trained with the FrameNet data release 1.2. The FRED classifier identifies the semantic class (i.e. the frame) of the frame-evoking elements while the ROSY classifier assigns segments of the sentences to the semantic roles (i.e. the FEs). According to the evaluations in (Erk and Pado 2006), FRED performs relatively better than the ROSY system in terms of precision and recall.

Step	System	Version
POS-tagging	TNT	2
Lemmatization	TreeTagger	-
Syntactic Parsing	Collins' Parser	1.0
Machine learning	Mallet	mallet 0.4

Table 3. SHALMANESER settings at each step

As SHALMANESER is a loosely coupled tool chain for automated annotation, there are different learner systems that are supported by the tool and can be exploited at its different modules. Table 3 shows the different systems that we have used for each task in the SHALMANESER settings.

To enhance the semantic class and role labeling accuracy on the output SALSA/TIGER xml files (Erk and Pado 2004), an intensive manual augmentation over the automated outputs has been conducted using the SALSA annotation tool known as SALTO (Burchardt, Erk et al. 2006).

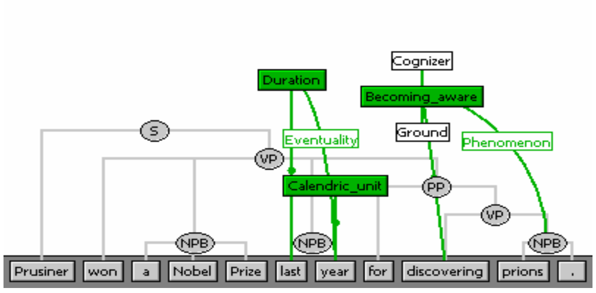


Figure 3. Incomplete automated annotation of an example sentence

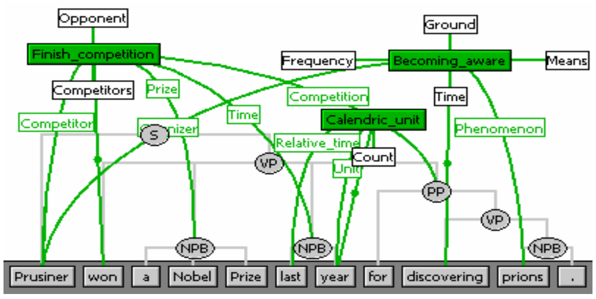


Figure 4. Comprehensive annotation of an example sentence after manual augmentation

The task of manual annotation (i.e. augmentation of the automated annotation) is an exhaustive process which thoroughly examines each sentence word-by-word. It includes:

[1] http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/

- *Frame evocation*: if a predicate could have evoked a correct frame in FrameNet with respect to the sense of the predicate,
- *Frame change*: in case the frame already evoked (by SHALMANESER/FRED) is not of the correct semantic class of the predicate,
- *FE assignment*: when parts of a sentence could have been assigned to FEs,
- *FE assignment correction*: where there is a need for changing the connectivity of the sentence segments to the FEs of a frame (as indicated by SHALMANESER/ROSY).

Figure 3 shows an example sentence annotated by SHALMANESER visualized in SALTO. Figure 4 shows that after the task of manual augmentation.

In the manual annotation process of the example sentence, the frame “Finish-competition” has been added, the wrongly assigned frame “Duration” has been eliminated, and the FEs of the two frames “Calendric-unit” and “Becoming-aware” have been corrected in their corresponding sentence segments.

The manual annotation, in order to develop the most comprehensive and up-to-date annotation, uses the FrameNet data version 1.3 with 795 semantic frames.

4.3 Statistics of Annotation

The manual annotation process includes changing many of the frames and FEs assignments. To have a better picture of the task, the two subtasks of the manual augmentation (i.e. the frame changing and the FEs corrections) are separately analyzed in a statistical approach as shown in Table 4.

Frame level	Frames changed	FEs changed
SHALMANESER evoked frames	N/A	35.741
Manually augmented frames – FN1.2	59.926	158.356
Manually augmented frames – FN1.3	74.160	181.860

Table 4. Average number of frames and FEs changed in manual augmentation per 10 passages (per question) – raw measures

As shown in Table 4, the FE-oriented augmentation of the frames evoked by SHALMANESER includes changing of the 35.741 FEs of those frames on average, where there are no frames added. In the two next levels, there are frames added with respect to the two versions of the FrameNet data. It is obvious that with respect to the FrameNet dataset 1.3, there are more frames and FEs that can be added to the text.

Table 5 translates the measures in Table 4 when normalized against the sentences.

Frame level	Frames changed	FEs changed
SHALMANESER evoked frames	N/A	1.560
Manually augmented frames – FN1.2	2.547	6.844
Manually augmented frames – FN1.3	3.182	7.848

Table 5. Average number of frames and FEs changed in manual augmentation per 10 passages (per question) – sentence number normalized

To have more FrameNet-oriented statistical sense of the annotation, Table 6 shows the overall measures with respect to the frames and FEs assigned to the corpus sentences.

Element	Measure
# Frames evoked	21741
# Frames evoked (unique)	592
# FEs assigned	40589
# FEs assigned (unique)	2586

Table 6. FrameNet-oriented statistics of the annotated corpus

The total number of unique frames evoked in the corpus, i.e. 592, covers 74.465% of the total frames in the FrameNet data release 1.3 containing 795 semantic frames. On the other hand, the overall frame count on the corpus (i.e. 21741) represents the concurrency rate of the frames over the sentences as 6.299 frames per sentence on average.

Having all this statistical information about the manual annotation of the SHALMANESER outputs, we have analyzed the accuracy of SHALMANESER and the other levels of annotation against the human level annotation. Table 7 considers the baseline human level augmentation with respect to the FrameNet data 1.2 and Table 8 shows the average accuracy of the tasks with taking the FrameNet data 1.3 frames into account in the baseline human level annotation.

$$Accuracy = \frac{\# correct\_items}{\# items} \qquad (1)$$

In order to calculate the accuracy of the labeling task at each level, the percentage of correct frames and/or FEs assigned at each level is measured over the total number of items (i.e. frames or FEs) assigned at the highest level of assignment (i.e. the human level). Equation 1 shows the formulations to measure the accuracies over the item (i.e. frame and FE) assignment at each level where *correct_items* refers to the items correctly assigned at the level under consideration and *items* is the whole set of assignments at that level. This formula is used for both frames and FEs.

As shown in Table 7 and Table 8, the overall accuracy of the fully automated shallow semantic parsing on the open domain texts of the AQUAINT collection is not promisingly high. The task of FE assignment, especially, seems to be a challenging process where the overall accuracy is not reaching more than 17.000%.

The low performance of the automated shallow semantic parsing is an evidence for the need for more comprehensively labeled training sets of text annotations. We believe that SHALMANESER could have achieved much higher accuracies if it had been trained against a more extensive training set, though the classification features are of importance as well. Our work is in line with this requirement (i.e. a comprehensive labeled training set with concurrent frames evoked) for current and future semantic parsers.

Frame level	Frame evocation accuracy	FE assignation accuracy
SHALMANESER evoked frames	41.765%	17.000%
SHALMANESER evoked frames – FEs augmented	41.765%	43.539%
Manually augmented frames – FN1.2	100%	100%

Table 7. Average accuracy of annotation at each frame level – FN 1.2 in baseline human level annotation

Frame level	Frame evocation accuracy	FE assignation accuracy
SHALMANESER evoked frames	38.003%	15.665%
SHALMANESER evoked frames – FEs augmented	38.003%%	40.042%
Manually augmented frames – FN1.3	100%	100%

Table 8. Average accuracy of annotation at each frame level – FN 1.3 in baseline human level annotation

4.4 Quality

An important aspect of the manual augmentation is the quality of the output annotation with respect to the two main subtasks namely frame evocation and FE assignment to the sentence segments. The manual augmentation process, in our work, has been conducted by one coder; however, there has been a method for validating the output annotation with respect to the inter-annotator agreement rates.

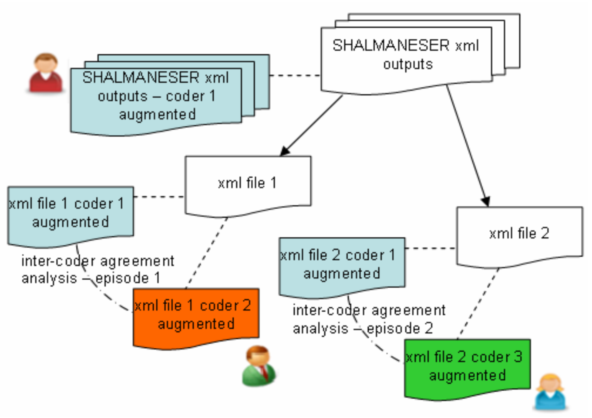

SHALMANESER xml outputs
SHALMANESER xml outputs – coder 1 augmented
xml file 1
xml file 1 coder 1 augmented
xml file 2
xml file 2 coder 1 augmented
inter-coder agreement analysis – episode 1
xml file 1 coder 2 augmented
inter-coder agreement analysis – episode 2
xml file 2 coder 3 augmented

Figure 5. Inter-annotator agreement analysis scenario

After finishing the manual augmentation task by the sole coder, two separate portions (10 passages each) of the same SHALMANESER outputs (not the whole set) have been annotated by two other coders (three coders in total). Each portion is then augmented by a coder i.e. portion 1 by coder 2 and portion 2 by coder 3. With this setting, there are two portions annotated by two coders where the pairs are coder1-coder2 and coder1-coder3. In two separate episodes, the inter-annotator agreement has been measured in the sense of frame evocation and FE assignation. Figure 5 depicts the scenario.

The overall agreement has been then calculated as the average values on the two measure sets.

The alpha statistics has been used in other similar tasks for frame agreement calculation between annotators (Erk, Kowalski et al. 2003). In this task, we use the Kappa statistics (Cohen 1960) as shown in Equation 2 where *P(A)* indicates the observed agreement among the coders (i.e. the probability of the agreed items over the total number of items coded) and *P(E)* is the expected agreement.

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \quad (2)$$

The computation of *P(E)* as the probability of agreement among coders by chance is the challenging part in the Kappa statistics which can be approached in different ways. We benefit from the Siegel and Castellan's agreement table (Eugenio and Glass 2004) to compute the expected agreement *P(E)*.

$$P(E) = \sum_j \left(\frac{\sum_i n_{ij}}{Nk}\right)^2 \quad (3)$$

Equation 3 shows how they calculate the *P(E)* measure for any number of possible labels, where *N* is the total number of observations, *k* is the total number of labels that coders can assign to each item, and n_{ij} is the number of codings of label *j* to item *i*.

For each predicate in the corpus (i.e. items), we consider 4 labels *no-frame*, $frame_{coder1}$, $frame_{coder2}$, and *frame*, where *no-frame* is used for the predicates that are not assigned to any frame by the coders, $frame_{coder1}$ indicates that a frame has been chosen by the coder1 which is not the same as the frame selected by the coder2 as $frame_{coder2}$ indicates. In cases that the coders agree on the same frames, the tag *frame* is chosen for both coders. Table 9 depicts an example agreement table according to 10 predicates {P1, P2, …, P10}.

Predicate	No-frame	$\text{Frame}_{\text{coder1}}$	$\text{Frame}_{\text{coder2}}$	Frame
P1	1	1	0	0
P2	1	1	0	0
P3	1	0	1	0
P4	2	0	0	0
P5	2	0	0	0
P6	2	0	0	0
P7	0	1	1	0
P8	0	0	0	2
P9	0	0	0	2
P10	0	0	0	2

Table 9. An example frame agreement table for 10 predicates and 2 coders

With this example agreement table, $K_{S\&C}$ is calculated as follows. First, *P(A)* is calculated as 6/10=0.600 (at 6 rows there are agreements indicated by the number 2). Second, for each label *j*, p_j (i.e. the proportion of predicates assigned to label *j*) is calculated using the formula in Equation 4.

$$p_j = \frac{1}{Nk} \sum_i n_{ij} \quad (4)$$

With k=4 and N=10, we have $p_{no\text{-}frame}$=0.225, $p_{frame\text{-}coder1}$=0.075, $p_{frame\text{-}coder2}$=0.050, and p_{frame}=0.150. Having these values per label, the overall *P(E)* is equal to 0.079. Finally, the $K_{S\&C}$ measure is (0.600-0.079)/(1-0.079)=0.565.

There are different possibilities for measuring the frame evocation agreement with regard to the total number of predicates (i.e. *N*). We calculate the agreement with respect to three predicate counts: i) all predicates in the corpus, ii) all FEEs (frame evoking elements) from the coders' point of view, and iii) all FEEs from the FrameNet point of view. The second set is the maximum number of FEEs identified by either coder, where the third set contains all of the predicates which inherit from an existing frame in FrameNet.

The inter-annotator agreement on the FE assignment task is, however, more problematic due to a few challenges:

- The different coders may assign slightly different string segments to the same FEs as there is no boundary detection performed to identify and unify the set of arguments in the sentences prior to the manual annotation,
- The task of comparison between the FEs assigned by the two coders is not very well addressed as it is not obvious which FEs need to be aligned,
- The total number of FEs over which the agreement is calculated is not constant. That is, the identification of a baseline set of the FEs to calculate the agreement on is a challenge.

Analysis episode	Frame agreement		
	All predicates	FEEs – coders' view	FEEs – FN view
coder1-coder2	0.804	0.387	0.661
coder1-coder3	0.789	0.378	0.708
Average agreement	*0.796*	*0.382*	*0.684*

Table 10. Inter-annotator frame agreement rates

With respect to the above-mentioned challenges, we have set different measurement strategies for FEs agreement calculation. We consider both exact and partial matches between the instances (arguments) assigned to the FEs. On the other hand, we consider two overall sets of FEs to calculate the agreement over: i) the union set of the FEs assigned by the two coders, and ii) the maximum set (i.e. number) of the FEs assigned by either coder. The method of calculation of the FE agreement is based on the percentage of the agreed FEs over the total number of FEs according to one of the two overall sets mentioned above.

Frame level	FE agreement (%)			
	Exact match		Partial match	
	Max	Union	Max	Union
coder1-coder2	17.100	14.420	25.278	21.316
coder1-coder3	29.032	31.629	36.363	39.616
Average agreement	*23.066*	*23.024*	*30.820*	*30.466*

Table 11. Inter-annotator FE agreement rates

Table 10 and Table 11 summarize the two episodes of the agreement analysis for the frame and FE agreement in the annotations. We expect that the calculated agreements over the sub-corpora can be generalized to the whole set of annotation.

The overall agreement on frame evocation for the predicates is much higher than that of the FE assignments to the text of the corpus. This was expected though not such a drastic difference as in the two tables. We believe that the low FE agreement is due to three main reasons: i) different coders' skills on the annotation task results in different standards of annotation which damage the FE assignment task more than the frame evocation process. This happens as the total number of FE assignations is much more than that in terms of frames, ii) different coders' knowledge in frame semantics and more specifically in FrameNet initiates different understandings of the annotation task. Once again, this is more strongly affecting the FE assignment task as there are lots of FEs with different definitions in FrameNet, and iii) dissimilar interpretations of the sentences and clauses by the coders yield an undesired difference in annotations.

5 Conclusion

In line with the current trend for natural language applications based on a high level of semantic information, the necessity of the labeled training sets which include reasonable amounts of annotation has emerged. In this paper, the different aspects of a comprehensive open domain text annotation task are described. We have performed an exhaustive annotation on a subset of the AQUAINT collection which can be used for both shallow semantic parser training and answer extraction module tuning in QA systems that work on the basis of semantic class identification and role labeling approaches. The annotated corpus contains the tags of frames and FEs of FrameNet with a reasonable coverage over the total number of the semantic frames in the FrameNet dataset 1.3 (see section 4.3).

We have approached the comprehensive annotation in a two-step process of automated shallow semantic parsing and manual augmentation of the outputs of the first sub-process. We have shown the different statistical information of the corpus and its annotated version which will be publicly available soon. In addition, the complete inter-annotator agreement calculation methodology has been detailed with respect to the two main subtasks of FrameNet-based annotation (i.e. frame agreement and FE agreement). One of the next goals in this direction is to study and formulate methods to reach higher levels of inter-annotator agreement, especially with respect to the task of FE assignments. We believe that the methodological attributes of our task can shed more light on the similar activities and their challenges. We may re-conduct the inter-annotator agreement calculation process with the annotators with more specific knowledge in the FrameNet concepts.

Acknowledgment

We would like to thank Katrin Erk from the SALSA project, Saarland University, for providing us with SHALMANESER and her assistance with SHALMANESER and SALTO.

References

Baker, C. F., C. J. Fillmore and J. B. Lowe. 1998. *The Berkeley FrameNet project*. In *Proceedings of the International Conference on Computational Linguistics*, pp. 86 - 90.

Burchardt, A., K. Erk, A. Frank, A. Kowalski and S. Pado. 2006. *SALTO -- a versatile multi-level annotation tool*. In *Proceedings of the LREC-06*, 2006.

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

Cook, W. A. (1989), *Case Grammar Theory*, Georgetown University Press.

Erk, K. 2004. *Frame assignment as word sense disambiguation*. In *Proceedings of the 2006 IAENG International Workshop on Computer Science, IWCS'06*, Tilburg University, Tilburg, The Netherlands, 2004.

Erk, K., A. Kowalski, S. Pado and M. Pinkal. 2003. *Towards a Resource for Lexical Semantics: A Large German Corpus with Extensive Semantic Annotation*. In *Proceedings of the ACL-03*, Sapporo, Japan, 2003.

Erk, K. and S. Pado. 2004. *A powerful and versatile XML format for representing role-semantic annotation*. In *Proceedings of the LREC-04*, Lisbon, Portugal, 2004.

Erk, K. and S. Pado. 2005. *Analysing models for semantic role assignment using confusability*. In *Proceed-*

ings of the HLT/EMNLP-05, Vancouver, B.C., Canada, 2005.

Erk, K. and S. Pado. 2006. *SHALMANESER - a toolchain for shallow semantic parsing*. In *Proceedings of the LREC-06*, 2006.

Eugenio, B. D. and M. Glass. 2004. The Kappa Statistics: A Second Look. *Association for Computational Linguistics* **30**(1): 95-101.

Fillmore, C. J. 1968. The case for case. *Universals in Linguistic Theory*: 1-88.

Fillmore, C. J. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech* **280**: 20-32.

Lowe, J. B., C. F. Baker and C. J. Fillmore. 1997. *A frame-semantic approach to semantic annotation*. In *Proceedings of the SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, April 4-5, 1997.

Ofoghi, B., J. Yearwood and R. Ghosh. 2006. *A semantic approach to boost passage retrieval effectiveness for question answering*. In *Proceedings of the 29th Australian Computer Science Conference*, January 16-19, 2006. pp. 95-101.

Petruck, M. R. L. (1996), *Frame semantics*. Philadelphia, John Benjamins.

Pradhan, S., 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, May 2-7, 2004.

Voorhees, E. M. 2004. *Overview of the TREC 2004 question answering track*. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC-2004)*, 2004.

Proceedings of the Australasian Language Technology Workshop 2007, pages 92-99

Question Prediction Language Model

Luiz Augusto Pizzato and **Diego Mollá**
Centre for Language Technology
Macquarie University
Sydney, Australia
{pizzato, diego}@ics.mq.edu.au

Abstract

This paper proposes the use of a language representation that specifies the relationship between terms of a sentence using question words. The proposed representation is tailored to help the search for documents containing an answer for a natural language question. This study presents the construction of this language model, the framework where it is used, and its evaluation.

1 Introduction

Although Information Retrieval (IR) can be helped by NLP techniques such as named entity (NE) recognition, phrase extraction and syntax parsing (Strzalkowski, 1999), they are not generally used due to their high complexity. One such task that people can perform somewhat easily whilst still being hard for computers is the answering of factoid questions based on textual content. The Question Answering (QA) Track of TREC (Voorhees, 2005) focuses on answering questions using the AQUAINT corpus (Graff, 2002), which contains 375 million words from three different sources of newswire data: Xinhua News Service (XIE) from People's Republic of China, the New York Times News Service (NYT), and the Associated Press Worldstream News Service (APW).

For the QA task, not only it is important to find an answer in a document, but also to find the documents that might contain the answer in the first place. Most QA systems take the approach of using off-the-shelf IR systems to return a list of documents that may contain an answer, and then processing the list of documents to look for the required answer. Normally the processing time for every question in these systems is long because of the sheer amount of work that is required after the list of document is returned.

Many QA systems focus on the input and output of IR systems. For example, Dumais et al. (2002) perform a passive-to-active voice transformation of the question, in an attempt to bring the IR query closer to the document it is expected to retrieve. Some IR work focuses on improving QA by passage retrieval re-ranking using word overlap measures. For instance, Tellex et al. (2003) compare a group of passage retrieval techniques and conclude that those that apply density-based metrics[1] are the most suitable to be used for QA.

Some work has been done on IR models that specifically aid the QA task. The work of Monz (2004) defines a weighting scheme that takes into consideration the distance of the query terms. Murdock and Croft (2004) propose a translation language model that defines the likelihood of the question being the translation of a certain document. Tiedemann (2005) uses a multi-layer index containing more linguistic oriented information and a genetic learning algorithm to determine the best parameters for querying those indexes when applied for the QA task. Tiedemann argues that since question answering is an all-natural language task, linguistic oriented IR will help finding better documents for QA.

In this paper we propose a language representa-

[1] Ranking of passages based on the number of query words and the proximity between them.

tion that when used in the IR stage of a question answering system improves its results. As a consequence it helps to reduce the processing time due to a better retrieval set and because it has the capacity of giving answer cues.

This paper is divided into five sections. The next section presents the Question Predication Language Model and some of its features. Section 3 introduces how the the model is used and how the necessary resources for its usage were built. Section 4 describes some experiments and present some preliminary results. Section 5 presents the concluding remarks and future work.

2 Question Prediction Language Model

We describe a language model that focuses on extracting a simple semantic representation of an English text that can be easily stored in digital databases and processed by Information Retrieval (IR) tools. We focus on extracting a particular kind of semantic that help us to find the location of a text that has some likelihood of answering a question. The model and its semantic are defined as Question Prediction (QP).

The Question Prediction Language Model (QPLM) represents sentences by specifying the semantic relationship among its components using question words. In this way, we focus on dividing the problem of representing a large sentence into small questions that could be asked about its components. In other words, we represent the relationship among key words of a sentence as short questions. For instance, the sentence "*Jack eats ham*" could be represented by the following two triples: $Who(eat, Jack)$ and $What(eat, ham)$. Using this model it is possible to answer short questions that focus on relations existent inside a sentence context, such as "*Who eats ham?*" and "*What does Jack eat?*".

The QPLM represents sentences as semantic relations expressed by triples $q(w, a)$ where q is a question word, w is the word that concerns the question word q and a is the word that answers the relation q about w. For instance the relation $Who(eat, Jack)$ tells us that the person who eats is Jack. The representation of our semantic relations as triples $Q(w, a)$ is important because it allows the representation of sentences as directed graphs of semantic relations. This representation has the capacity of generating questions about the sentence being analysed. Figure 1 shows such a representation of the sentence: "*John asked that a flag be placed in every school*".

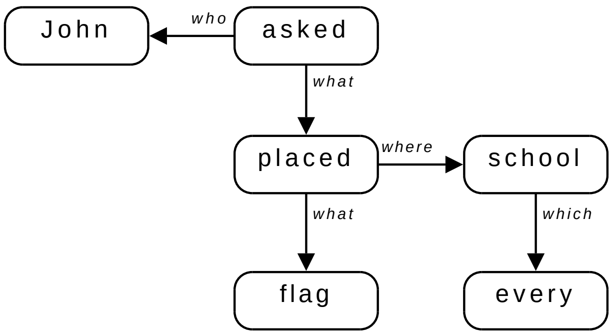


Figure 1: Graph Representation

Having the sentence of Figure 1 and removing a possible answer a from any relation triple, it is possible to formulate a complete question about this sentence that would require a as an answer. For instance, we can observe that removing the node $John$ we obtain the question "*Who asked for a flag to be placed in every school?*" where Who was extracted from the triple $Who(ask, John)$. The same is valid for other relations, such as removing word *school* to obtain question "*Where did John asked for a flag to be placed?*". The name Question Prediction for this model is due to its capability of generating questions regarding the sentence that has been modeled.

In this section, we have shown how our model represents the semantic information. In the next section we focus on the implementation of QPLM and its usage.

3 Building and using QPLM

As observed in Figure 2, a training set of QPLM triples was created using mapping rules from a corpus of semantic role labels. Using a syntactic parser and a NE recognizer with our training set, we were able to learn pattern rules that we further applied in the processing of the AQUAINT corpus.

PropBank (Palmer et al., 2005) is a corpus with annotated predicate-argument relations from the same newswired source of information as the Penn Treebank[2]. We used PropBank as our starting point

[2]http://www.cis.upenn.edu/ treebank

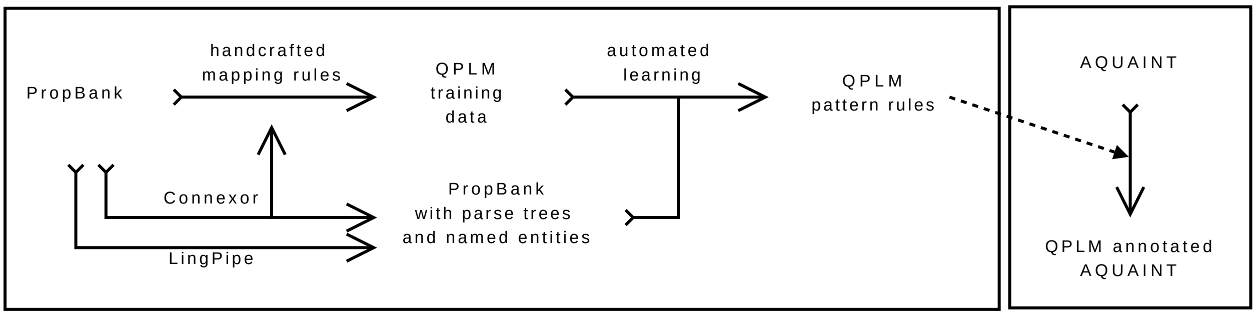


Figure 2: Creation and usage of pattern rules.

because it comprises the same textual style, and the predicate-argument relations (also referred to as semantic roles) can be mapped to QPLM triples.

We studied the possibility of using semantic role labeling tools to perform the semantic annotation, however our experiments using these tools showed us that they have not yet achieved a reasonable speed performance. For instance, the SwiRL semantic role labeling system[3] would take a couple of years to fully process the AQUAINT corpus. In contrast, our system takes a couple of days if all the necessary information is already at hand; adding the time required for syntactic parsing and NE recognition, the total processing period is not longer than two weeks.

3.1 Training corpus

PropBank is processed through a set of mapping rules from the predicate-argument relations to QPLM. Using a PropBank map as our training data gives us the benefit of a large training set, but at the same time it will only create relations that are present in PropBank, therefore excluding some relations that we wish to include. For instance, relations that do not involve any action, such as the ownership relation in $(Whose(car, Maria))$ and the quantity relation in $(HowMany(country, twenty)))$, among others.

PropBank defines relations between predicate and arguments without properly defining their meaning. On the other hand, it does keep a format where the argument number 0 represents the agent acting upon something and argument number 1 represents patients or themes. PropBank was manually annotated according to the PropBank Marking Guidelines (Babko-Malaya, October 2006). The guidelines represent an effort to build a consistent set of relations, however a closer look at the corpus shows that consistency is a hard task to achieve, particularly with the vaguely defined arguments number 3 onwards. For those cases the inclusion of a function tag proved to be useful[4].

Observing how arguments and predicates relate to each other, we created a set of rules mapping from argument-predicate relations to the QPLM. The basic differences between both models is that the QPLM triple contains a label representing a more specific semantic relation, and that it associates only the head of the linked phrases. For instance, the sentence "*The retired professor received a lifetime achievement award*" is represented as:

(1) **Semantic Roles**: [The retired professor]^{ARG0} [received]^{pred} [a lifetime achievement award]^{ARG1}.

(2) **QPLM**: Who(receive, professor), What(receive, award)

As can be observed in (1), semantic role labeling does not provide information about which is the main term (normally the head of a phrase) of each argument, while in (2), QPLM represents relations between the phrase heads. In order to find the phrase head, we applied a syntactic parser (Connexor[5]) to PropBank sentences. However, the phrase heads are not always clearly defined (particularly when the syntactic parse tree is broken due to problems in the parser) creating an extra difficulty for the mapping process. When a syntactic path cannot be found between predicates and any of the words from the argument, we then try to find the head of the phrase

[3]`http://swirl-parser.sourceforge.net/`

[4]A function tag is information attached to the arguments representing relations such as negation, location, time and direction.

[5]`http://www.connexor.com`

by syntactically parsing the phrase by itself. If this also fails to provide us with a head, we simply use the first available non-stopword if possible.

The stage of finding the related phrases heads showed to be quite important, not only because we would be defining which words relate to each other, but also because if a broken parse tree is found, no rules could be learnt from the resulting QPLM triple. An analysis of the data showed us that 68% of the QPLM triples derived from PropBank were generated from an unbroken parse, while the rest used some of the other methods.

We understand that even though our model has similarities with Semantic Role Labeling, we are taking a step further in the sense of semantic representation. QPLM has a finer semantic representation meaning that a predicate argument relation in PropBank might have different representations in QPLM. Our mapping rules takes into consideration not only the number of the argument but also the predicate involved and the POS or NE of the related words.

Even though we cover different aspects of PropBank in our mapping, we observed that many predicates hold different meanings for the same arguments which creates a problem for our mapping strategy. This problem was not fixed because of the prohibitive amount of work needed to manually mark all the different meanings for the same predicate in different sentences. In these cases, where the same predicates and the same argument represent different semantics according to the QPLM, we chose the one most representative for the set of sentences using that predicate and argument. For instance, the argument number 3 of predicate *spend* for the majority of the cases represents a quantity of money that was spent (a HowMuch label), however we have one case where the argument is *cash* (a What label). This type of mapping compromises the accuracy of our conversion, however a randomly selected set of 40 documents was manually evaluated showing that nearly 90% of the QPLM triples were correctly converted.

After the mapping was finalized we obtained a training set of rules with 60,636 rules, and 39 types of semantic relations (Table 1).

aboutwhat	do	outofwhat
adv	forwhat	overwhat
afterwhat	fromwhat	subj
againstwhat	how	towhat
aroundwhat	howlong	towhom
aswhat	howmuch	underwhat
atwhat	howold	what
behindwhat	intowhat	when
belowwhat	inwhat	where
beneathwhat	likewhat	who
betweenwhat	obj	whom
beyondwhat	ofwhat	why
bywhat	onwhat	withwhat

Table 1: QPLM Semantic Relations

Original: John kicked the ball bought by Susan.
QPLM: Who(kick, John), What(kick, ball), What(buy, ball), Who(buy, Susan)
Parse Tree: $John_{np}{}^{subj} \rightarrow kick_{va} \leftarrow^{obj} ball_{nn} \leftarrow^{det} the_{det}$ $ball_{nn} \leftarrow^{mod} buy_{vp} \leftarrow^{agt} by_{prep} \leftarrow^{pcomp} Susan_{np}$
Named Entities: <ENAMEX Type=NAME> John </ENAMEX> kicked the ball bought by <ENAMEX Type=NAME> Susan </ENAMEX>.

Table 2: Training Files

3.2 Rule learning

The PropBank corpus, after being automatically converted to QPLM triples, is used to learn the rules that are used to find the QPLM information of plain text. The QPLM annotation relies on the output of a syntactic parser and of a named-entity recognizer for its annotation and for the rule learning process. We are currently using Connexor for syntax parsing and LingPipe[6] to recognize NEs. Our semantic model uses pattern rules (PRules) created from the representation of the same sentence as syntactic parse trees, MUC style named entity, and a list of QPLM triples. Table 2 presents the different information that we use for training.

Having these representations at hand, a set of rules is learned using the following process (see Figure 3 for an example):

1. replace the part of speech information with the respective named entity category in the syntactic parse tree;

[6]http://www.alias-i.com/lingpipe/

2. identify leaf-to-root links along the combined syntactic and named-entity (S+NE) path between w and a for every triple $Q(w, a)$;
3. for the existing S+NE paths, replace w and a by a marker in both the triples and the paths, registering those as pattern rules (PRule);

   repeat steps 2 to 3 for all triples and documents;
4. combine all PRules found, calculate their frequency of occurrence and group them by common triples. It is important to note that if we have a sentence such as "*Jack eats*", we would have a frequency of two (2×) for the pattern $a_{person}{}^{subj} \rightarrow w_{va}$.

1. $John_{\mathbf{person}}{}^{subj} \rightarrow kick_{va}$
2. $Who(kick, John)$: $John_{person}{}^{subj} \rightarrow kick_{va}$
3. $Who(w, a)$: $a_{person}{}^{subj} \rightarrow w_{va}$
4. $Who(w, a)$:
   - 1× : $a_{person}{}^{subj} \rightarrow w_{va}$
   - 1× : $a_{person}{}^{pcomp} \rightarrow by_{prep}{}^{agt} \rightarrow w_{vp}$

Figure 3: Process example

After computing all the training files we would have a resulting PRule file containing all possible S+NE paths that can generate the manually defined triples. If an S+NE path could not be found then a PRule cannot be generated and the current training triple is skipped.

3.3 Applying QPLM

Using the training corpus described above, we found all the PRules needed in order to generate the semantic triples when having an S+NE representation. The rules are grouped by QPLM triples, having their S+NE paths attached with a frequency value. This frequency value represents how many times an S+NE path was used to generated a PRule in the training corpus.

To convert S+NE files into QPLM, we start by applying those PRules that have the highest frequency values. These PRules are believed to be the most significant ones. Also it is important to observe that if an S+NE path generates different QPLM triples, we only need to apply the one with the higher frequency. For instance, if the pattern $w_{person}{}^{subj} \rightarrow a_{va}$ is associated with the triple $Who(w, a)$ with frequency of 8 and with the triple $Where(w, a)$ with a frequency of 2, the S+NE path will only generate the Who triple. Because frequency is the decisive factor, in the previous example we have 20% of chance of wrongly assigning an incorrect semantic label.

We observed that more precise PRules could be created taking into account that some verbs constantly generate a different QPLM triple for the same S+NE path. These new PRules (which we refer to as FW) are defined with a fixed w becoming less frequent but at the same time more precise. The precision of FW rules combined with the generality of the previous ones (which we refer to as GN) assure us that we have a correct analysis of a known verb as well as fair guess of an unseen one. To ensure that known verbs are evaluated first by the more precise FW rules, we assign a much higher weight to those rules than GN ones. An evaluation using the combination of both types of rules has shown us that assigning a weight 800 times higher to FW than to GN gives us the best results.

We also observed that due to the large amount of learnt PRules, the process for creating the QPLM was slow. In order to improve the speed performance of the process, we decided to compromise our system precision and recall by removing the least important rules, i.e. those with a frequency equal to one. The lower number of PRules caused a decrease of recall which is more noticeable when taking into account the FW rules. Even though we experienced a decrease of precision, removing low frequent PRules causes the removal of abnormal PRules that were generated by parsing errors.

In the next section we describe the environment where QPLM was applied, followed by some experimental results.

4 Evaluation

It is possible to evaluate our model implementation on how well it performs the task of assigning the correct semantic labels to a certain text. However because the model was designed so it would improve the IR stages of a QA system, we believe that the most significant evaluation at this point is in terms of how well it helps us solving this specific problem.

Since we have not yet implemented lexical semantic substitution or any other IR techniques such as stemming or lemmatization, a comparison with a full-fledged state-of-the-art IR system is not relevant. The lack of these techniques makes it somewhat harder to boost the confidence on single sentences or windows of text if the proper context is not recorded. However, we have confidence that the model can help to provide cues of possible answers by substituting the partial match between a question word and document representation. For instance, if the question "*Where will a flag be placed?*" is presented and the sentence in Figure 1 is considered, a partial match *school* can be considered as a possible answer.

4.1 The IR model comparison

We have compared our QPLM triples with bag-of-words (unigram) and syntactic dependency triples. In all three cases the indexing and retrieval methods were the same, only the indexed units were different. We have implemented an IR framework that can hold relational information such as n-grams, syntactic, semantic role label and QPLM. Our framework is implemented so that it supports fast indexing (hundreds of documents per second in a low-end desktop machine) and it retrieves using a vector space model. The framework allows distributed indexing and retrieval under other models but they are not yet implemented.

During the development and test phases of our framework, we have implemented different vector space models of retrieval. We have implemented the unigram model, along with syntactic relations, semantic role labeling and QPLM.

The unigram model associates words with documents as well as it adds the position of them within the document. The inclusion of position information allows the use of a ranking function based on proximity. We also implemented a syntactic model using the output of Connexor. The model associated words with documents as well as words with their respective heads and/or modifiers. Since we are using a vector space model, computing TF and IDF over this type of relation has a different meaning. TF for full matching triples would be how many of them with the same $Syntactic - Relation$, $head$ and $modifier$ are found, while TF could also mean partial matches where one or two elements are not needed to match. IDF in this setup would be similar to TF but with the scope of the whole document set.

In the semantic role labeling model, a complete match of predicate and argument is not expected; because of this we only consider partial matches (if a word appears as the correct argument or predicate). However we expect a larger weight for a document when all the words properly match. In this model IDF and TF are calculated by taking into account the words found in the right context.

In the QPLM we have a very similar model to the syntactic relation one. However in QPLM not all the words will relate to each other, causing a large number of words to be missing in the final representation. To overcome this problem, we compute IDF and TF as the syntactic relations and we also add the TF/IDF weights from an unigram model. Because QPLM relations are much less frequent, when a match is found they will have higher weights than unigrams. Unigrams and QPLM are combined so that we do not discard relevant documents when they contain important keywords that are missing in the QPLM representation.

4.2 Evaluation over IR and QA

We have shown in the previous sections that the QPLM analysis relies on a syntactic parser and on a named-entity recognizer, in order to build a training set used to look for pattern rules and then to analyse text files into this model. We have not analysed the correlation among the performance of the syntactic parser nor the named entity recognizer, however we observed that our model has problems learning rules when these components offer poor results. As explained previously in section 3.1, if we cannot find a rule connecting two nodes into the same parse tree, we cannot learn a conversion rule. These cases account for 42,609 out of 135,537 rules, reducing in practical matters our training set to only 68% of its original size. Many of these cases are due to broken parse trees returned by Connexor. We have not yet experimented with different parsers, however a possible outcome of such experiment might be that having a broken structure and therefore losing the training instance is more desirable than having the full parse, but with the wrong dependencies.

We have also filtered out the pattern rules that have the same S+NE path but are not the most frequent one regarding their QPLM triple. By doing this we discard 12% of all the rules (20% of GN and 4% of FW). We do not use the rules when their frequency values are equal to one, this will cause an extra drop in the number of rules used to 44%. As expected the removal of low frequency rules have a stronger impact on FW rules than in GN rules (54% and 33% respectively).

This information is important because we can then predict what the upper limit of our performance is when measuring the recall using the set of rules we built as a training and testing set. According to the values presented, using all the rules with a frequency of 2 or more we have an upper limit of recall of 38%. A ten-fold evaluation over the training set has given us a value of 24% for recall.

When comparing the PropBank mapped files with the files analysed by our technique, it is possible to observe that the amount of QPLM triples in our semantic analysis is much larger than the ones mapped from PropBank. The reason is that PropBank only marks certain predicates in a sentence, while QPLM also provides relation among other verbs and other sentence words. Because of this we performed a manual evaluation of the precision of our technique over a set of 20 randomly selected documents and we found that 50% of the relations can be seen as correct. We also observed that many of the relations that were wrongly generated were due to some errouneous S+NE path. Filtering out this wrong pattern from the rule file will improve our precision. The important fact is that even though our performance over the analysis of the QPLM does not appear to be very high, the generated rules show to be very useful when applied to IR and the QA task.

We have retrieved one hundred documents for the set of 1449 questions of the TREC 2004, 2005 and 2006 QA track (Voorhees, 2005; Voorhees and Dang, 2006) and verified the existence of the answer string in each of these documents. We have performed this retrieval process for the unigram, syntactic relations and QPLM models. Due to data storage constraints at this moment we only have the results for the XIE and APW newswire corpus.

The comparison between the three models shows that we can obtain better documents to answer natural language questions if we take into consideration the same linguistic relations in the question and in the terms present in the documents. We measure our results using coverage and redundancy measures (Roberts and Gaizauskas, 2004). Coverage tells us how much of the question set we can answer using the top-N documents, while redundancy tells us how many documents per question contain an answer. These results are presented in Table 3.

Coverage

n docs.	5	10	25	50	100
UNI	0.2227	0.2718	0.3246	0.3591	0.3885
SYN	0.2307	0.2752	0.3325	0.3691	0.3964
QPLM	0.2310	0.2787	0.3370	0.3757	0.4040

Redundancy

n docs.	5	10	25	50	100
UNI	0.5421	1.0062	2.2279	4.0363	7.2158
SYN	0.5715	1.0535	2.3080	4.1091	7.2686
QPLM	0.5729	1.0584	2.3567	4.2486	7.6405

Table 3: Results for APW and XIE

As we observe in Table 3, for a large collection of documents and questions our system performs consistently better than unigram and syntactic relations. We have performed a paired t-test for statistic significance using the results of the individual questions for QPLM and unigrams showing that there is a 99% percent of chance that the improvement is not random for the results in the APW corpus. However, a paired t-test did not reject the null hypothesis in the test performed with the XIE corpus. This may be an indication that the XIE Newswire Corpus is written with a linguistic style that our system has not been able to take advantage of. Perhaps it strongly differs from the style present in our training set (PropBank) causing our rules not to be successfully used. Further work is needed to understand the main differences among these corpora. By understanding this we might find ways to adjust our system towards different textual styles.

Even though coverage and redundancy are good measures for evaluating a retrieval set for QA, we have observed that these measurements do not always relate to each other (Pizzato et al., 2006). For this reason we have applied the retrieval set to a QA system in order to observe if it does help to improve its results. Using the retrieval sets generated by the different models in the AnswerFinder QA system (van Zaanen et al., 2007) showed us that the QPLM performed 25% better than the unigram model and 9.3% better than the syntactic model. Even though AnswerFinder is not among the best performing QA systems, it does give us some insight on what a retrieval set should contain.

5 Concluding Remarks

In this paper we have presented a semantic model that represents the relations among sentence components by labeling them with question words. This model was built to assist the task of question answering, particularly at the IR stages. We understand that by providing documents that are better suited towards finding an answer for a natural language question, QA system would not only return better answers but also become faster.

The work presented here shows that the QPLM can be used effectively in IR frameworks, and a comparison with the unigram and syntactic model demonstrates that we are able to improve the overall IR results. We have already implemented the predicate-argument model in our IR framework and we plan to compare it with QPLM. Because the current semantic role labeling systems are impractically slow when applied to large corpora, the comparison will be done using a reduced number of documents.

In this work, we focused on the single impact of our technique on the retrieval stages of a QA system. In future work we will include different retrieval methods in our IR framework to enable a valid comparison with state-of-the-art IR systems. We also plan to manually study the PRules so as to identify the ones causing some drops in the precision and recall of our model, and to construct an automatic method that would help this process.

As explained, we had data storage constraints which made the evaluation more difficult. As future work we plan to distribute the retrieval process and to perform evaluations with the whole AQUAINT corpus and with the NYT documents. We also intend to evaluate the impact that different retrieval sets have in a broader range of QA systems.

References

O. Babko-Malaya. October 2006. Propbank annotation guidelines.

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. 2002. Web question answering: is more always better? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, Tampere, Finland. ACM Press.

D. Graff. 2002. *The AQUAINT Corpus of English News Text*. Linguistic Data Consortium, Philadelphia.

C. Monz. 2004. Minimal span weighting retrieval for question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

V. Murdock and W.B. Croft. 2004. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguist*, 31(1):71–106.

L. A. Pizzato, D. Molla, and C. Paris. 2006. Pseudo relevance feedback using named entities for question answering. In *Proceedings of the Australasian Language Technology Workshop 2006.*, Sydney.

I. Roberts and R. J. Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *ECIR*, volume 2997 of *Lecture Notes in Computer Science*, pages 72–84. Springer.

T. Strzalkowski. 1999. *Natural Language Information Retrieval*. Kluwer Academic Publishers, Norwell, MA, USA.

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47, Toronto. ACM Press.

J. Tiedemann. 2005. Optimizing information retrieval in question answering using syntactic annotation. In *Proceedings of RANLP 2005*, pages 540–546, Borovets, Bulgaria.

M. van Zaanen, D. Molla, and L. A. Pizzato. 2007. Answerfinder at trec 2006. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.

E. M. Voorhees and H. T. Dang. 2006. Overview of the TREC 2005 question answering track. In *Text REtrieval Conference*.

E. M. Voorhees. 2005. Overview of the TREC 2004 question answering track. In *Text REtrieval Conference*.

Exploring Abbreviation Expansion for Genomic Information Retrieval*

Nicola Stokes, Yi Li, Lawrence Cavedon and Justin Zobel
National ICT Australia, Victoria Research Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Victoria 3010, Australia.
{nstokes, yli8, lcavedon, jz}@csse.unimelb.edu.au

Abstract

Abbreviations are commonly found instances of synonymy in Biomedical journal papers. Information retrieval systems that index paragraphs rather than full-text articles are more susceptible to term variation of this kind, since abbreviations are typically only defined once at the beginning of the text. One solution to this problem is to expand the user query automatically with all possible abbreviation instances for each query term. In this paper, we compare the effectiveness of two abbreviation expansion techniques on the TREC 2006 Genomics Track queries and collection. Our results show that for highly ambiguous abbreviations the *query collocation* effect isn't strong enough to deter the retrieval of erroneous passages. We conclude that full-text abbreviation resolution prior to passage indexing is the most appropriate approach to this problem.

1 Introduction

Query expansion is a well-known technique used in Information Retrieval (IR) to address the problem of lexical variation between the query and semantically related terms in relevant documents (Efthimiadis, 1996). While on average query expansion methods, such as *relevance feedback* (Ruthven and Lalmas, 2003), have been shown to improve retrieval performance, there are many examples where query effectiveness has been significantly downgraded. However, in terminology rich domains where word sense distributions are heavily skewed, query expansion has been shown to have more of a consistent positive effect on retrieval performance. This trend is particularly evident in the passage retrieval task investigated at the TREC (Text REtrieval Conference) Genomics Track (Hersh et al., 2006).

In this paper, we investigate the impact of various expansion term types on passage retrieval effectiveness in the biomedical domain. Our results show that expanding with ontologically related words (synonyms, hypernyms, hyponyms) significantly improves performance; however, abbreviation expansion shows more inconsistent results similar to those seen in general domain expansion experiments. One would expect that the performance of IR systems that index paragraphs rather than full-text articles would greatly benefit from this sort of expansion, since abbreviations are typically only defined once in an entire document.

We report the results of our investigation on the TREC 2006 Genomic retrieval task. We compare two abbreviation expansion techniques: the first adds abbreviations found in the ADAM database of abbreviations (Zhou et al., 2006a); the second, uses a pseudo relevance feedback strategy to identify query term abbreviations in the full-text documents of an initial set of retrieved passages. Despite the benefit of mutual disambiguation across query terms, referred to as the *query term collocation effect*

*National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

(Krovetz and Croft, 1992), both approaches reduce retrieval effectiveness, leading to the conclusion that abbreviation resolution in the document collection is more appropriate than expansion.

Another contribution of this paper is our novel concept-based IR ranking method. This ranking method is an adaptation of the Okapi method, enhanced so as to deal with multi-concept queries derived from natural language questions. Our method ensures that passages containing at least one occurrence of all the query concepts out-rank passages that contain many occurrences of only one of the concepts. We also describe a paragraph reduction strategy that increases the TREC defined answer extraction accuracy score of our system. Finally, we discuss our plans for future work.

2 Information Retrieval for Functional Genomics

Biomedical text retrieval is a very active area of research, driven by the biomedical community's need for high precision systems that answer specific biological questions not captured in the plethora of database resources (of varying quality) containing different types of biological information. Two distinct user information needs have been recently investigated by the IR community: *clinical text retrieval* (which supports patient-centred clinical research or care) and *functional genomic text retrieval* (which supports researchers involved in laboratory experiments). In this paper, we focus on genomic retrieval. An interesting overview of evidence-based medical retrieval in the clinical domain can be found in (Lin and Demner-Fushman, 2006).

Functional Genomics is the study of gene and protein function and interaction at a molecular level, and the effects of this interaction on biological processes that results in phenotypic outcomes (such as disease) in organisms. An important yet very time-consuming part of the functional genomics pipeline for researchers involves arriving at biologically motivated explanations for the output of bioinformatics-based clustering techniques such as gene expression profiling. Since a single experiment can involve thousands of genes, even a competent biologist needs to turn to a search engine to determining whether the functional dependencies found in these clusters make sense.

The TREC Genomics Track was established in 2003 with the aim of supporting the evaluation of information retrieval systems capable of answering the types of questions typically posed by genomicists such as:

- What is the role of gene A in disease B?
- What effect does gene A have on a particular biological process?
- How do genes A and B interact in the function of a specific organ?
- How do mutations in gene A influence a particular biological process?

Each of these four query templates were investigated at the 2006 Genomics Track. In all, 28 queries were evaluated on a collection of full-text journal papers, where the task was to retrieved relevant answer passages rather than full-text documents. In the following section we describe our novel genomic retrieval system.

3 System Description

In this section, we describe the different components in our Genomic IR architecture. Our IR system is a version of the Zettair engine[1] that we have specifically modified for passage retrieval and biomedical query term expansion.

Collection Preprocessing

The TREC collection consists of full-text journal articles obtained by crawling the Highwire site[2]. The full collection contains 162,259 documents and is about 12.3 GB in size when uncompressed. After preprocessing, the whole collection becomes 7.9 GB. The collection is pre-processed as follows:

Paragraph Segmentation: for evaluation purposes the Genomics Track requests that the ranked answer passages must be within specified paragraph boundaries.

[1]`http://www.seg.rmit.edu.au/zettair/`
[2]`http://www.highwire.org`

Sentence Segmentation: all sentences within paragraphs are segmented using an open source tool.[3]

Character Replacement: Greek characters represented by gifs are replaced by textual encodings; accented characters such as “À” or “Á” are replaced by “A”; Roman numbers are replaced by Arabic numerals. These replacements are very important for capturing variations in gene names.

Removal: all HTML tags, very short sentences, paragraphs with the heading *Abbreviations*, figures, tables and some special characters such as hyphens, slashes and asterisks are removed: (Trieschnigg et al., 2006) has shown that small changes in the tokenisation strategy such as these improve the performance of biomedical IR.

Query Expansion

Once the collection has been indexed, querying can begin. In the 2006 Genomics Track, each query or topic contains at least two biological concepts or entities which could be a gene (“NM23”), a protein (“p53”), a disease (“ovarian cancer”) or a biological process (“ethanol metabolism”). TREC simplifies the query preprocessing task by ensuring that all topics conform to the query templates discussed in Section 2. The following is a sample query, Topic 173 from the 2006 track, which contains two concepts: “PrnP” (a gene) and “mad cow disease” (a disease):

What is the role of **PrnP** *in* **mad cow disease***?*

Our query expansion process proceeds as follows. First, each gene or protein in the query is expanded with entries from the Entrez Gene database.[4] Since the same gene may occur in many different species, and many of their synonyms only differ with respect to capitalisation, we choose the first entry retrieved that belongs to the species type *Homo sapien*. Then, terms in the *Official Symbol*, *Name*, *Other Aliases* and *Other Designations* fields, for the gene, are added to the query.

For all disease and biological process mentions in the query, we use the MeSH[5] taxonomy of medical terms to find their synonyms (using the *Entry Terms* and *See Also* fields). The terms’ hyponyms (descendants) and hypernyms (ancestors) in the MeSH tree structure are also used as expansion terms.

Gene Variant Generation

As well as expanding with synonyms, we use a “gene variant” generation tool to generate all the possible variants for both original query terms and expanded terms. Our segmentation rules are similar to those used by (Buttcher et al., 2004). We describe our rules as follows:

Given a gene name containing a hyphen or punctuation, or a change from lower case to upper case, or from a character to a number (or vice versa), or a Greek character (e.g. “alpha”), we call this a *split point*. A word is split according to all its split points, and all variants are generated by concatenating all these split parts, optionally with a space inserted. Greek characters are also mapped to English variants, e.g. “alpha” is mapped to “a”.

For example, on the query term “Sec61alpha”, we would generate the following lexical variants which are also commonly used forms of this term in the collection: “Sec 61alpha”, “Sec61 alpha”, “Sec 61 alpha”, “Sec 61a”, “Sec61 a”, “Sec 61 a”, “Sec61a”;

In phrases, we replace hyphens (“-”), slashes (“/”) and asterisks (“*”) in the queries with spaces. For example, “subunit 1 BRCA1 BRCA2 containing complex” is a variant of “subunit 1 BRCA1/BRCA2-containing complex”.

Concept-based Query Normalisation

Our document ranking method is based on the Okapi model (Robertson et al., 1994). Many participant systems at the TREC Genomics track use the Okapi method for ranking documents with respect to their similarity to the query. However, there are two fundamental problems with using this model on TREC Genomic queries.

The first problem regards Okapi not differentiating between concept terms and general query terms

[3]`http://l2r.cs.uiuc.edu/~cogcomp/atool.php?tkey=SS`

[4]`http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene`

[5]`http://www.nlm.nih.gov/mesh`

in the query. For example, consider two documents, one containing the terms “mad cow disease” and “PrnP”, and the other containing the terms “role” and “PrnP”. Clearly the first document containing the two biological concepts is more relevant. The second problem occurs because TREC 2006 topics contain more than one concept term. It is possible that a short paragraph that discusses one concept only will be ranked higher than a longer paragraph which mentions two concepts. Again this is an undesirable outcome.

To overcome these problems, a *Conceptual IR* model was proposed in (Zhou et al., 2006b). In this paper we propose another method called the *concept-based query normalisation* which is based on the Okapi model and similar to the method introduced in (Li, 2007; Stokes et al., 2008) for geospatial IR.

The first problem is solved by dividing query terms into two types: *general terms* t_g and *concept terms* t_c. Given a query with both concept and general terms, the similarity between a query Q and a document D_d is measured as follows:

$$\mathit{sim}(Q, D_d) \quad = \quad \mathit{gsim}(Q, D_d) + \mathit{csim}(Q, D_d)$$

where $\mathit{gsim}(Q, D_d)$ is the *general similarity score* and $\mathit{csim}(Q, D_d)$ is the *concept similarity score*. The general similarity score is given by:

$$\mathit{gsim}(Q, D_d) \quad = \quad \sum_{t \in Q_g} \mathit{sim}_t(Q, D_d) = \sum_{t \in Q_g} r_{d,t} \cdot w_t \cdot r_{q,t}$$

where Q_g is the aggregation of all general terms/phrases in the query. The concept similarity score is given by:

$$\begin{aligned} \mathit{csim}(Q, D_d) &= \sum_{C \in Q_c} \mathit{sim}_c(Q, D_d) \\ &= \sum_{t \in C, C \in Q_c} \mathit{Norm}(\mathit{sim}_{t_1}(Q, D_d), \ldots, \mathit{sim}_{t_N}(Q, D_d)) \\ &= \sum_{t \in C, C \in Q_c} (\mathit{sim}_{t_1} + \frac{\mathit{sim}_{t_2}}{a} + \cdots + \frac{\mathit{sim}_{t_N}}{a^{N-1}}) \end{aligned}$$

where Q_c is the aggregation of all concepts in the query, C is one concept in Q_c, and t_i is a term/phrase in the query, after expansion, which belongs to the concept C; the t_i are listed in descending order according to their Okapi similarity scores sim_{t_1}, …, sim_{t_N}:

$$\mathit{sim}_t(Q, D_d) \quad = \quad r_{d,t} \cdot w'_t \cdot r_{q,t}$$

where

$$\begin{aligned} r_{d,t} &= \frac{(k_1 + 1) \cdot f_{d,t}}{k_1 \cdot [(1-b) + b \cdot \frac{W_d}{avgW_d}] + f_{d,t}} \\ w'_t &= \log \frac{N - \max(f_t, f_{t_q}) + 0.5}{\max(f_t, f_{t_q}) + 0.5} \\ r_{q,t} &= \frac{(k_3 + 1) \cdot f_{q,t}}{k_3 + f_{q,t}} \end{aligned} \qquad (1)$$

where k_1 and b are usually set to 1.2 and 0.75 respectively, and k_3 can be taken to be ∞. Variable W_d is the length of the document d in bytes; $avgW_d$ is the average document length in the entire collection; N is the total number of documents in the collection; f_t is the number of documents in which term t occurs; and $f_{\{d,q\},t}$ is the frequency of term t in either a document d or query q.

Note that (1) is an adjustment of the calculation for the weight w'_t of an *expansion* term t appearing in the query: for expansion term t, its own term frequency f_t and the corresponding original query term's frequency f_{t_q} are compared, and the larger value used — this ensures the term contributes an appropriately normalised “concept weight”.

To solve the second problem, we use the following rules to ensure that for two passages P_1 and P_2, where one contains more unique concepts than the other, the number of concepts *ConceptNum(P)* will override the Okapi score *Score(P)* and assign a higher rank to the passage with more unique concepts:

```
if $\mathit{ConceptNum}(P_1) > \mathit{ConceptNum}(P_2)$ then
    $\mathit{Rank}(P_1) > \mathit{Rank}(P_2)$
else if $\mathit{ConceptNum}(P_1) < \mathit{ConceptNum}(P_2)$ then
    $\mathit{Rank}(P_2) > \mathit{Rank}(P_1)$
else if $\mathit{Score}(P_1) \geq \mathit{Score}(P_2)$ then
    $\mathit{Rank}(P_1) > \mathit{Rank}(P_2)$
else
    $\mathit{Rank}(P_2) > \mathit{Rank}(P_1)$
```

Abbreviation Finder

Although MeSH and Entrez Gene contain many synonyms and related terms, one important type of lexical variant, *abbreviations*, has very low coverage in both databases. For example, “AD” is a commonly used abbreviation for “Alzheimer’s Disease”. Since the long and short form (“Alzheimer’s Disease (AD)”) only appear together at the beginning of each journal document, many relevant passages will contain “AD” only and so will appear less relevant than they should against a query containing “Alzheimer’s Disease”. Hence, expanding the given query with “AD” should improve retrieval effectiveness.

As already mentioned, there are two methods for collecting abbreviations from the literature: the first uses the static resource ADAM (Zhou et al., 2006a), while the second uses our pseudo relevance feedback method for extraction these abbreviations during run time. The advantage of the latter approach is that it dynamically collects abbreviations and so does not suffer from the coverage and update problems of static resources like ADAM. The following is an overview of how our abbreviation feedback step contributes to the retrieval process:

1. Retrieve the first 1000 documents which include at least one instance of each concept in the query.

2. From this subset of documents, find terms which fit the pattern “*Term* (*Abbr*)”, where “*Term*” is a concept in the query (original or expanded) and “*Abbr*” is the abbreviation or synonym defined in the text.

3. Among all the detected abbreviations or synonyms, remove all the multi-word terms, terms that do not have any overlapping characters with the original term, and terms which occur less than three times.

4. For all remaining abbreviations or synonyms, use the above generation tool to formulate all their lexical variants, and add them to the query. The expanded query is then re-submitted to the retrieval engine, and the passage extraction step, described below, is applied.

Passage Extraction

As already mentioned the 2006 Genomics Track defined a new question answering-type task that requires short full-sentence answers to be retrieved in response to a particular query. However, before answer passages can be generated, we first retrieve the first 1000 ranked paragraphs for each topic, and use the following simple rules to reduce these paragraphs to answer spans.

Two methods are examined in this paper which are best described with an example. Given a paragraph consisting of a set of sentences $\{(s_1, i), (s_2, i), (s_3, r), (s_4, r), (s_5, i), (s_6, r), (s_7, i), (s_8, i), (s_9, r), (s_{10}, i)\}$, where r is relevant (that is, mentions at least one query term) and i is irrelevant. *Method A* shortens a paragraph by removing irrelevant sentences from its start and end until a relevant sentence is detected. Hence, it would produce the following passage of sentences: $\{(s_3, r), (s_4, r), (s_5, i), (s_6, r), (s_7, i), (s_8, i), (s_9, r)\}$.

This extraction method does not split a paragraph into multiple passages if irrelevant sentences occur within the resultant passage. *Method B*, on the other hand, addresses this issue by splitting a passage if there are two or more consecutive irrelevant sentences within this span. Hence, Method B would produce the following two passages for this paragraph: $\{(s_3, r), (s_4, r), (s_5, i), (s_6, r)\}$ and $\{(s_9, r)\}$.

After one of these passage extraction techniques has been applied for a particular topic, we re-rank passages by re-indexing them, and re-querying the topic against this new index, using the global statistics from the original indexed collection, i.e. using term frequency f_t and the average paragraph length $avgW_d$.

4 Experimental Methodology

4.1 Data and Evaluation Metrics

We used the TREC 2006 Genomics Track evaluation resources to determine the effectiveness of our system. The TREC 2006 collection consists of 162,259 full-text documents from 49 journals publish electronically via the Highwire Press website[6]. The track also provided 28 topics expressed as natural

[6]More information on the TREC dataset can be found at: `http://ir.ohsu.edu/genomics/2006data.html`

language questions, formatted with respect to seven general topic templates. Participants were asked to submit the first 1,000 ranked passages returned by their system for each of the topics (Hersh et al., 2006). Passages in this task are defined as text sequences that cannot cross paragraph boundaries (delimited by HTML tags), and are subsets of the original paragraphs in which they occur. As is the custom at TREC, human judges were used to decide the relevance of passages in the pooled participating system results. These judges also defined exact passage boundaries, and assigned topic tags called *aspects* from a control vocabulary of MeSH terms to each relevant answer retrieved.

Mean Average Precision, or MAP, is a popular IR metric for evaluating system effectiveness. The TREC Genomics Track defines three versions of the MAP score calculated at various levels of granularity: *Document*, *Passage* and *Aspect*. Traditionally the MAP score is defined as follows: first, the average of all the precision values at each recall point on a topic's *document* ranked list is calculated; then, the mean of all the topic average precisions is determined. Since the retrieval task at the Genomics Track is a question answering-style task, a metric that is sensitive to the length of the answer retrieved was developed.

Passage MAP is similar to document MAP except average precision is calculated as the fraction of characters in the system passage overlapping with the gold standard answer, divided by the total number of characters in every passage retrieved up to that point in the ranked list. Hence, a system is penalised for all additional characters retrieved that are not members of the human evaluated answer passage.

The TREC organisers also wanted to measure to what extent a particular passage captured all the necessary information required in the answer. Judges were asked to assign at least one MeSH heading to all relevant passages. Aspect average precision is then measured as the number of aspects (MeSH headings) captured by all the relevant documents up to the recall point in the ranked list for a particular query. Relevant passages that did not contribute any new aspect to the aspects retrieved by higher ranked passages were removed from the ranking. Aspect MAP is defined as the mean of these average topic precision scores.

4.2 Experimental Results

In this section, we examine the increased effectiveness obtained when different expansion information is added to the original query. We also evaluate the effect of our proposed abbreviation feedback technique, and our novel answer expansion module, on system performance.

As explained in Section 3, our system uses Entrez Gene for expansion of genes to their synonymous instances. In addition, all term variants are generated for their abbreviations as described in Section 3, while other biological entities in the query (e.g., diseases) are expanded using MeSH. Table 1 presents the MAP scores for the following system runs:

- `Baseline`: Zettair system with no expansion
- `SYN`: query expansion using Entrez gene and MeSH expansion (*Synonym* and *See Also* entries in MeSH) of query terms
- `SYN+HYPO`: query expansion using Entrez gene and MeSH expansion, including *Hyponyms* (i.e., specialisations)
- `SYN+HYPER`: query expansion using Entrez gene and MeSH expansion, including *Hypernyms* (i.e., generalisations)
- `SYN+HYPER+VAR`: query expansion using Entrez gene, *Gene Variant Generation*, and MeSH expansion, including *Hypernyms*

All expansion run MAP scores show a statistically significant[7] improvement over the baseline MAP. The only expansion experiment that does not incrementally improve the results is the addition of hyponym terms (i.e. specialisation) from MeSH. On the other hand, hypernyms (i.e. generalisations) improve the performance of the `SYN` run by nearly 5%. This result may be explained by the fact that at a passage level, generalised expressions are commonly used to refer to query terms that have been discuss earlier in the document. For example, the following sentence is clearly relevant to the *mad cow disease* query presented in Section 3: "These *prion diseases* are characterised by the accumulation of an abnormal (aberrantly folded) isoform of a cellular host

[7]We use a paired Wilcoxon signed-rank test at the 0.05 confidence level to determine significance.

Table 1: Table showing improvement in MAP score obtained over baseline MAP when the query is expanded with various combinations of related terms: synonyms (SYN), hyponyms (HYPO), hypernyms (HYPER) and gene lexical variants (VAR)

Run	Passage MAP			Aspect MAP			Document MAP		
Baseline	0.0480			0.1838			0.3355		
SYN	0.0888†	+85.0%	$P = 0.005$	0.3499†	+90.3%	$P < 0.001$	0.4711†	+40.4%	$P = 0.008$
SYN+HYPO	0.0878†	+83.0%	$P = 0.007$	0.3417†	+85.9%	$P = 0.001$	0.4632†	+38.1%	$P = 0.02$
SYN+HYPER	0.0933†	+94.4%	$P < 0.001$	0.3695†	+101%	$P < 0.001$	0.4843†	+44.3%	$P = 0.002$
SYN+HYPER+VAR	0.0949†	+97.6%	$P < 0.001$	0.3827†	+108%	$P < 0.001$	0.5080†	+51.4%	$P < 0.001$

protein PrPC”. However, it would only be ranked highly if the generalisation relationship from *mad cow disease* to *prion disease* has been established. Expanding the query term with the immediate parent terms in the different MeSH hierarchies usually results in a few focussed terms being added to the query. In contrast, adding specialisations may result in a much larger number of term additions, depending on the generality of the query term. For example, the term *neurons* has 18 unique subcategories one level below its position in the MeSH hierarchy and many more beyond this level.

Our best system run (SYN+HYPER+VAR) used ontological and gene variant expansion, and achieved a 97.6% increase in Passage MAP over the baseline run. Similarly large increases in Aspect and Document MAP were also observed. A detailed analysis showed that many passages had been either missed or ranked lower than expected by our system due to the occurrence of query term abbreviations in the relevant passage. These abbreviations were not captured in either of our ontological resources.

Table 2 compares the performance of the two abbreviation expansion strategies described in Section 3. Ontological expansion using the ADAM abbreviation database reduces our best Passage MAP score by 36%. Our abbreviation feedback loop performs better, producing a small increase in Document MAP over the baseline, but slightly lower Aspect and Passage MAPs. In some respects, this feedback result is disappointing as a manual analysis of the added abbreviations shows that many useful synonyms were added to the query, which should, in theory, help to retrieve additional passages and boast the rankings of other relevant passages.

However, there is one big drawback to abbreviation expansion that isn’t characteristic in other types of expansion we have explored: abbreviations are much more ambiguous. For example, the abbreviation “AD” is a very commonly used reference to “Alzheimer’s disease”; however, according to ADAM, “AD” has 35 unique long forms defined in MedLINE abstracts. For example, “AD” can also refer to the phrases “after discharge”, “autosomal dominant”, “autistic disorder”, and other unrelated concepts.

IR researchers have found that query-term ambiguity is less of a problem than one might expect because of the *query term collocation effect* (Krovetz and Croft, 1992): query terms mutually disambiguate each other because their intended senses tend to co-occur together in relevant documents in the collection. For example, for the query term “cell”, adding the term “blood” to the query ensures that documents using the biological sense are ranked higher. Hence, one would expect that despite abbreviation ambiguity, great gains in IR effectiveness would be possible using expansion. However, when the total number of possible unabbreviated forms is factored into the expansion process, it is clear that an excessive amount of ambiguity is added in.

A manual analysis of the results backs up this observation: although new relevant passages containing abbreviations are being retrieved, paragraph ranking is being affected to such an extent that previously retrieved passages are “dropping out” of the top 1000 items in the ranked list.

However, our results also show that dynamic abbreviation expansion does not degrade performance as dramatically as expansion with ADAM. The feedback process ensures that only abbreviations that occur in documents of high ranked passages, mentioning all query concepts, are added to the query. Thus, these abbreviations have the highest potential for providing positive impact on retrieval effectiveness.

Table 2: Table showing effect on system performance when additional expansion terms are added from the ADAM abbreviation (+Adam) database and our system Abbreviation feedback loop (+Abbr).

Run	Passage MAP			Aspect MAP			Document MAP		
`SYN+HYPER+VAR`	0.0949			0.3827			0.5080		
`SYN+HYPER+VAR+Adam`	0.0600†	-36.8%	$P < 0.001$	0.2387†	-37.6%	$P < 0.001$	0.4105†	-19.2%	$P = 0.001$
`SYN+HYPER+VAR+Abbr`	0.0920	-3.06%	$P = 0.3$	0.3784	-1.12%	$P = 0.4$	0.5171	$+1.79\%$	$P = 0.3$

Table 3: Table showing effect of two passage extraction strategies A and B on system performance

Run	Passage MAP			Aspect MAP			Document MAP		
`Best`	0.0920			0.3784			0.5171		
`Best+A`	0.1100†	$+19.6\%$	$P < 0.001$	0.3673	-2.93%	$P = 0.3$	0.5123	-0.93%	$P = 0.3$
`Best+B`	0.1175†	$+27.7\%$	$P < 0.001$	0.3518†	-7.03%	$P = 0.004$	0.5021	-2.90%	$P = 0.08$

The general conclusion from these abbreviation expansion experiments is clear: knowledge of these synonymous instances is obviously beneficial, but a method that reduces the impact of their high ambiguity is necessary. We discuss our proposed solution to this problem in Section 5.

Our final experiment (see Table 3) shows that the TREC's Passage MAP score can be increased by capturing the exact answer span in each relevant paragraph. Section 3 proposed two methods for achieving this: *Method A* finds the longest text span in paragraph that contains all query terms; *Method B* splits the span and remove sentences if there is a distance of one or more sentences between consecutive mentions of any of the query terms. Both reduction methods show improvements in Passage MAP, but at the expense of the other two metrics. This is to be expected, especially in the case of *Method B*, since splitting paragraphs means some relevant passages may get a lower rank or even drop out of the top 1000 passages.

Table 4 shows how our best run (`Best+B`) performs with respect to systems that participated in the official TREC 2006 Genomics Track. TREC_MEDIAN is the median value for each MAP score reported at TREC. UIC_TREC[8] was the top performing system submitted by the University of Illinois at Chicago, and UIC_SIGIR is the best post-submission Passage MAP score which was also published by the same group (Zhou et al., 2007). If our system had participated at TREC track we would have ranked 6th for Passage MAP, 3rd for Aspect MAP and 4th for Document MAP out of 92 submitted runs.

[8]The official name for this run was UICGenRun3.

Table 4: Table showing performance of our best Passage MAP scoring run Best+B with the top performing TREC systems on the Genomics Track

Run	Passage MAP	Aspect MAP	Document MAP
`UIC_SIGIR`	0.1823	0.3811	0.5391
`UIC_TREC`	0.1479	0.3492	0.5320
`Best+B`	0.1175	0.3518	0.5021
`TREC_MEDIAN`	0.0345	0.1581	0.3083

5 Discussion and Conclusions

The most successful systems at the TREC Genomics Track 2006 used a combination of expansion techniques from external resources such as publically-available and hand-crafted thesauri, in addition to lexical variant generation techniques similar to the one described in this paper. One of the principal contributions of this paper is our detailed analysis of what types of ontologically related terms (synonyms, hyponyms, hypernyms, lexical variants, abbreviations) provide the most impact when used as expansion terms. In particular, we have focussed on abbreviation expansion, which has high potential for impact when passages rather than full documents are being retrieved. However, our experiments show that their high ambiguity can in some cases reduce retrieval effectiveness.

There are two possible solutions to the abbreviation ambiguity problem: all abbreviations in the collection are identified in advance of indexing, and a unique identifier is assigned to each long-form and its corresponding abbreviated short-form. Hence, when the query is expanded, the unique identifier rather than the lexical form of the abbreviation is added to the query. Similarly, all abbreviations in the collection will be replaced by their identifier before passage indexing occurs. Another possible approach would be to explicitly add the long-forms of abbreviations in a passage to its index entry. This is a document expansion rather than a query expansion strategy. We plan to investigate both of these methods in our future work.

Another area for potential improvement that we wish to investigate further is paragraph reduction. Passage MAP is severely affected by long-answer text spans. Paragraph reduction is similar to answer extraction in factoid-based Question-Answering tasks. However, researchers have only recently begun to investigate answer extraction for more complex question types such as *Why* or *How* questions in an ad hoc retrieval setting (Allan, 2005). The Document Understanding Conference (DUC), which focusses on summarisation tasks, is also looking at complex questions; however, answers are typically generated by collating information from multiple documents (Dang, 2006).

References

J. Allan. 2005. Hard track overview in TREC 2005: High accuracy retrieval from documents. In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*.

S. Buttcher, C.L.A. Clarke, and G.V. Cormack. 2004. Domain-specific synonym expansion and validation for biomedical information retrieval. In *The Thirteen Text REtrieval Conference (TREC 2004) Proceedings*.

H.T. Dang. 2006. Overview of duc 2006. In *The Document Understanding Conference Workshop Proceedings*.

E. N. Efthimiadis. 1996. Query expansion. *Annual Review of Information Science and Technology*, 31:121–187.

W. Hersh, A. Cohen, P. Roberts, and H. Rekapalli. 2006. Trec 2006 genomics track overview. (Voorhees and Buckland, 2006).

Robert Krovetz and W. Bruce Croft. 1992. Lexical ambiguity and information retrieval. *Information Systems*, 10(2):115–141.

Yi Li. 2007. Probabilistic toponym resolution and geographic indexing and querying. Masters thesis, The University of Melbourne.

Jimmy Lin and Dina Demner-Fushman. 2006. The role of knowledge in conceptual retrieval: a study in the domain of clinical medicine. In *SIGIR ’06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 99–106.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *The Third Text Retrieval Conference (TREC 3) Proceedings*, Gaithersburg, Maryland, November.

I. Ruthven and M. Lalmas. 2003. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145.

Nicola Stokes, Yi Li, Alistair Moffat, and Jiawen Rong. 2008. An empirical study of the effects of NLP components on Geographic IR performance. *International Journal of Geographical Information Science*. To appear.

D. Trieschnigg, W. Kraaij, and F. de Jong. 2006. The influence of basic tokenization on biomedical document retrieval. In *SIGIR 2007 Proceedings*, Amsterdam, The Netherlands, July.

E. M. Voorhees and Lori P. Buckland. 2006. *The Fifteenth Text REtrieval Conference (TREC 2006) Proceedings*. NIST, Gaithersburg, Maryland.

W. Zhou, V. I. Torvik, and N. R. Smalheiser. 2006a. ADAM: another database of abbreviations in MEDLINE. *Bioinformatics*, 22(22):2813–2818.

W. Zhou, C. Yu, V. Tovik, and N. Smalheiser. 2006b. A concept-based framework for passage retrieval in genomics. (Voorhees and Buckland, 2006).

Wei Zhou, Clement Yu, Neil Smalheiser, Vetle Torvik, and Jie Hong. 2007. Knowledge-intensive conceptual retrieval and passage extraction of biomedical literature. In *SIGIR ’07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 655–662, New York, NY, USA. ACM Press.

Proceedings of the Australasian Language Technology Workshop 2007, pages 109-116

Parsing Internal Noun Phrase Structure with Collins' Models

David Vadas and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dvadas1, james}@it.usyd.edu.au

Abstract

Collins' widely-used parsing models treat noun phrases (NPs) in a different manner to other constituents. We investigate these differences, using the recently released internal NP bracketing data (Vadas and Curran, 2007a). Altering the structure of the Treebank, as this data does, has a number of consequences, as parsers built using Collins' models assume that their training and test data will have structure similar to the Penn Treebank's.

Our results demonstrate that it is difficult for Collins' models to adapt to this new NP structure, and that parsers using these models make mistakes as a result. This emphasises how important treebank structure itself is, and the large amount of influence it can have.

1 Introduction

Collins' parsing models (Collins, 1999) are widely used in natural language processing (NLP), as they are robust and accurate for recovering syntactic structure. These models can be trained on a wide variety of domains and languages, such as biological text, Chinese and Arabic. However, Collins' models were originally built for the Penn Treebank (Marcus et al., 1993), and as such, are predisposed to parsing not only similar text, but also similar structure.

This paper deals with the effects of assuming such a structure, after the Treebank has been altered. We focus on noun phrases (NPs) in particular, for two reasons. Firstly, there are a number of intricacies as part of Collins' model in this area. Indeed, a Collins-style parser uses a different model for generating NPs, compared to all other structures. Secondly, we can make use of our previous work in annotating internal NP structure (Vadas and Curran, 2007a), which gives us a ready source of Penn Treebank data with an altered structure.

Using this extended corpus, we make a number of alterations to the model itself, in an attempt to improve the parser's performance. We also examine the errors made, focusing on the altered data in particular, and suggest ways that performance can be improved in the future.

Next, we look at what effect the NP bracketing structure has. The existing Penn Treebank uses a generally flat structure (especially in NPs), but when internal NP brackets are introduced this is no longer the case. We determine what is the best representation to use for these new annotations, and also examine why that is.

Finally, we experiment with the task of identifying apposition within NPs, using manually annotated data as a gold standard. Although we find that this is a relatively easy task, the parser's performance is very low. The reasons for why Collins' models are unable to recover this structure are then explored.

The work described here raises questions about how researchers create NLP models, which may be for any task and not just parsing. Implicitly assuming that data will always retain the same structure can, as the results described here show, cause many problems for researchers in the future.

2 Background

We present here a brief introduction to Collins' (1999) models, focusing in particular on how they generate NPs. All of the models use a Probablistic Context Free Grammar (PCFG), with the Penn Treebank training data used to define the grammar, and to estimate the probabilities of the grammar rules being used. The CKY algorithm is then used to find the optimal tree for a sentence.

The grammar is also lexicalised, i.e. the rules are conditioned on the token and the POS tag of the head of the constituent being generated. Estimating probabilities over the grammar rule, the head and the head's POS is problematic because of sparse data problems, and so generation probabilities are broken down into smaller steps. Thus, instead of calculating the probability of the entire rule, we note that each rule can be framed as follows:

$$P(h) \rightarrow L_n(l_n) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_m(r_m) \tag{1}$$

where H is the head child, $L_n(l_n) \ldots L_1(l_1)$ are its left modifiers and $R_1(r_1) \ldots R_m(r_m)$ are its right modifiers. If we make independence assumptions between the modifiers, then using the chain rule yields the following equations:

$$P_h(H|Parent, h) \tag{2}$$

$$\prod_{i=1 \ldots n} P_l(L_i(l_i)|Parent, h, H) \tag{3}$$

$$\prod_{i=1 \ldots m} P_r(R_i(r_i)|Parent, h, H) \tag{4}$$

So the head is generated first, then the left and right modifiers (conditioned only on the head[1]) afterwards.

Now this differs for base-NPs, which use a slightly different model. Instead of conditioning on the head, the current modifier is dependant on the previous modifier, resulting in a sort of bigram model. Formally, equations 3 and 4 above are changed as shown below:

$$\prod_{i=1 \ldots n} P_l(L_i(l_i)|Parent, L_{i-1}(l_{i-1})) \tag{5}$$

$$\prod_{i=1 \ldots m} P_r(R_i(r_i)|Parent, R_{i-1}(r_{i-1})) \tag{6}$$

[1]There are also distance measures, the subcategorisation frame, etc, but they are not relevant for this discussion as they do not affect NPs.

There are a few reasons given by Collins for this. Most relevant for this work, is that because the Penn Treebank does not fully bracket NPs, the head is unreliable. When generating `crude` in the NP `crude oil prices`, we would want to condition on `oil`, the true head. However, `prices` is the head that would be found. Using the special NP submodel thus results in the correct behaviour. As Bikel (2004) notes, the model is not conditioning on the previous modifier *instead* of the head, the model is treating the previous modifier *as* the head.

The separate NP submodel also allows the parser to learn NP boundaries effectively, i.e. it is rare for words to precede `the` in an NP; and creates a distinct X-bar level, which Collins notes is helpful for the parser's performance.

In order to implement the separate base-NP submodel, a preprocessing step is taken wherein NP brackets that do not dominate any other non-possessive NP nodes are relabelled as NPB. For consistency, an extra NP bracket is inserted around NPB nodes not already dominated by an NP. These NPB nodes are reverted before evaluation.

In our previous work (Vadas and Curran, 2007a), we annotated the full structure of NPs in the Penn Treebank. This means that the true head can be identified, which may remove the need to condition on the previous modifier. We will experiment with this in Section 3.2.

2.1 Treebank Structure

Other researchers have also looked at the effect of treebank structure upon parser performance. Collins himself notes that binary trees would be a poor choice, as the parser loses some context sensitivity, and the distance measures become ineffective. He advocates one level of bracketing structure per X-bar level.

Goodman (1997) explicitly converts trees to a binary branching format as a preprocessing step, in order to avoid these problems. Johnson (1998) finds that the performance of simple PCFGs can be improved through tree transformations, while Klein and Manning (2001) observe that some simple tree transforms can increase parser speed. The variation shown in these approaches, all to the same task, highlights the difficulty in identifying optimal tree stucture.

	PREC.	RECALL	F-SCORE
Original PTB	88.88	88.85	88.86
NML and JJP bracketed PTB	88.55	88.15	88.35
Original structure	88.81	88.88	88.85
NML and JJP brackets only	76.32	60.42	67.44

Table 1: Parser performance

The issue of treebank structure extends to other languages as well, and implies further difficulties when comparing between languages. Kübler (2005) investigates two German treebanks with different annotation schemes, and finds that certain properties, such as having unary nodes and flatter clauses, increase performance. Rehbein and van Genabith (2007) suggest that the treebank structure also affects evaluation methods.

3 Internal NP Brackets

We begin by analysing the performance of Collins' model, using the Vadas and Curran (2007a) data. This additional level of bracketing in the Penn Treebank consists of NML and JJP brackets to mark internal NP structure, as shown below:

```
(NP (NML (NN crude) (NN oil) )
  (NNS prices) )

(NP (NN world) (NN oil) (NNS prices) )
```

In the first example, a NML bracket has been inserted around `crude oil` to indicate that the NP is left-branching. In the second example, we do not explicitly add a bracket around `oil prices`, but the NP should now be interpreted implicitly as right-branching.

The experiments are carried out using the Bikel (2004) implementation of the Collins (1999) parser. We use Sections 02-21 for training, and report labelled bracket precision, recall and F-scores for all sentences in Section 00.

Firstly, we compare the parser's performance on the original Penn Treebank to when it is retrained with the new NML and JJP bracketed version. Table 1 shows that the new brackets make parsing marginally more difficult overall (by about 0.5% in F-score). However, if we evaluate only the original structure, by excluding the new NML and JJP brackets, then we find that the F-score has dropped by only a negligible amount. This means that the drop in overall performance results from low accuracy on the new NML and JJP brackets.

	PREC.	RECALL	F-SCORE
Overall	88.09	87.77	87.93
Original structure	87.92	88.68	88.30
NML and JJP brackets only	100.00	53.54	69.74

Table 2: Parser performance with relabelled brackets

Bikel's parser does not come inbuilt with an expectation of NML or JJP nodes in the treebank, and so it is not surprising that these new labels cause problems. For example, head-finding for these constituents is undefined. Also, as we described previously, NPs are treated differently in Collins' model, and so changing their structure could have unexpected consequences.

In an attempt to remove any complications introduced by the new labels, we relabelled NML and JJP brackets as NP and ADJP, and then retrained again. These are the labels that would be given if internal NP structure was originally bracketed with the rest of the Penn Treebank. This relabelleling means that the model does not have to discriminate between two different types of noun and adjective structure, and for this reason, we might expect to see an increase in performance. This approach is also easy to implement, and eliminates the need for any change to the parser itself.

The results in Table 2 show that this is not the case, as the overall F-score has dropped another 0.5%. The NML and JJP brackets cannot be evaluated directly in this experiment, but we can compare against the corpus without relabelling, and count correct bracketings whenever a test NP matches a gold NML. The same is done for ADJP and JJP brackets. This results in a precision of 100%, because whenever a NML or JJP node is seen, it has already been matched against the gold-standard. Also, some incorrect NP or ADJP nodes are in fact false NML or JJP nodes, but this difference cannot be recovered.

We carried out a visual inspection of the errors that were made in this experiment, but which hadn't been made when the NP and NML labels were distinct. It was noticable that many of these errors occurred when a company name or other entity needed to be bracketed, such as `W.R. Grace` in the example NP below:

```
(NP
  (ADVP (RB formerly) )
  (DT a) (NML (NNP W.R.) (NNP Grace) )
  (NN vice) (NN chairman) )
```

	PREC.	RECALL	F-SCORE
Overall	88.51	88.07	88.29
Original structure	88.78	88.86	88.82
NML and JJP brackets only	75.27	58.33	65.73

Table 3: Performance with correct head-finding

We conclude that the model was not able to generalise a rule that multiple tokens with the NNP POS tag should be bracketed. Even though NML brackets often follow this rule, NPs do not. As a result, the distinction between the labels should be retained, and we must change the parser itself to deal with the new labels properly.

3.1 Head-finding Rules

The first and simplest change we made was to create head-finding rules for NML and JJP constituents. In the previous experiments, these nodes would be covered by the catch-all rule, which chooses the leftmost child as the head. This is incorrect in most NMLs, where the head is usually the rightmost child.

We define NML and JJP rules in the parser data file, copying those used for NPs and ADJPs respectively. We also add to the rules for NPs, so that child NML and JJP nodes can be recursively examined, in the same way that NPs and ADJPs are. This change is not needed for other labels, as NMLs and JJPs only exist under NPs. We retrained and ran the parser again with this change, and achieve the results in Table 3.

Once again, we are surprised to find that the F-score has been reduced, though only by 0.06% overall in this case. This drop comes chiefly from the NML and JJP brackets, whose performance has dropped by about 2%. As before, we scanned the errors in search of an explanation; however, there was no readily apparent pattern. It appears that conditioning on the incorrect head is simply helpful when parsing some sentences, and instances where the correct head gives a better result are less frequent.

3.2 The Base-NP Submodel

The next alteration to the parser is to turn off the base-NP submodel. Collins (1999, page 179) explains that this separate model is used because the Penn Treebank does not fully annotate internal NP structure, something that we have now done. Hopefully, with these new brackets in place, we can remove the NP submodel and perhaps even improve performance in doing so.

		PREC.	RECALL	F-SCORE
	Overall	72.11	87.71	79.14
1	Original structure	72.09	88.19	79.33
	NML /JJP brackets only	72.93	69.58	71.22
	Overall	87.37	87.17	87.27
2	Original structure	87.75	87.65	87.70
	NML /JJP brackets only	72.36	69.27	70.78
	Overall	86.83	86.46	86.64
3	Original structure	86.90	88.66	87.77
	NML /JJP brackets only	48.61	3.65	6.78

Table 4: Performance with the base-NP model off

```
            NP
           /  \
         NP    PP
        /  \
      NP    PP
```

Figure 1: An unlikely structure

We experimented with three different approaches to turning off the base-NP model. All three techniques involved editing the parser code:

1. Changing the $isBaseNP()$ method to always return false. This means that the main model, i.e. equations 3 and 4 are always used.

2. Removing the preprocessing step that creates NPB nodes. This alteration will have the same effect as the one above, and will also remove the distinction between NP and NPB nodes.

3. Changing the $isNP()$ method to return true for NMLs. This will affect which NPs are turned into NPBs during the preprocessing step, as NPs that dominate NMLs will no longer be basal.

The third change does not turn the base-NP model off as such, but it does affect where it functions.

The results are in Table 4, and in all cases the overall F-score has decreased. In the 1st change, to $isBaseNP()$, performance on NML and JJP brackets has actually increased by 3.78% F-score, although the original structure is almost 10% worse. The 2nd change, to the preprocessing step, results in a much smaller loss to the original structure, but also not as big an increase on the internal NP brackets. The 3rd change, to $isNP()$, is most notable for the large drop in performance on the internal NP structure.

There are a few reasons for these results, which demonstrate the necessity of the base-NP submodel. Collins (1999, §8.2.2) explains why the distinction between NP and NPB nodes is needed: otherwise,

ERROR	#	%	FP	FN	EXAMPLE	
Modifier attachment	213	38.04	56	157		
NML	122	21.79	21	101	lung cancer deaths	×
Internal Entity Structure	43	7.68	24	19	(Circulation Credit) Plan	×
Appositive Title	29	5.18	6	23	(Republican Rep.) Jim Courter	√
JJP	10	1.79	4	6	(More common) chrysotile fibers	√
Company/name postmodifiers	9	1.61	1	8	(Kawasaki Heavy Industries) Ltd.	√
Mislabelling	92	16.43	30	62	(ADJP more influential) role	√
Conjunctions	92	16.43	38	54	(cotton and acetate) fibers	√
Company names	10	1.79	0	10	(F.H. Faulding) & (Co.)	√
Possessives	61	10.89	0	61	(South Korea) 's	√
Speech marks and brackets	35	6.25	0	35	(" closed-end ")	√
Clearly wrong bracketing	45	8.04	45	0		
Right-branching	27	4.82	27	0	(NP (NML Kelli Green))	×
Unary	13	2.32	13	0	a (NML cash) transaction	×
Conjunction	5	0.89	5	0	(NP a (NML savings and loan))	×
Structural	8	1.43	3	5	(NP . . . construction spending) (VP (VBZ figures) . . .	×
Other	14	2.50	8	6		
Total	560	100.00	180	380		

Table 5: Error analysis

structures such as that in Figure 1, which *never* occur in the Treebank, are given too high a probability. The parser needs to know where NPs will not recurse anymore (when they are basal), so that it can generate the correct flat structure. Furthermore, the 3rd change effectively treats NP and NML nodes as equivalent, and we have already seen that this is not true.

3.3 Error Analysis

So far, all our changes have had negative results. We need to look at the errors being made by the parser, so that any problems that appear can be solved. Accordingly, we categorised every NML and JJP error through manual inspection. The results of this analysis are shown in Table 5, together with examples of the errors being made. Only relevant brackets and labels are shown in the examples, while the final column describes whether or not the particular bracketing shown is correct.

The most common bracketing error results in a modifier being attached to the wrong head. In the example, because there is no bracket around `lung cancer`, there is a dependency between `lung` and `deaths`, instead of `lung` and `cancer`. We can further divide these errors into general NML and JJP cases, and instances where the error occurs inside a company name or in a person's title.

These errors occur because the ngrams that need to be bracketed simply do not exist in the training data. Looking for each of the 142 unique ngrams that were not bracketed, we find that 93 of them do not occur in Sections 02-21 at all. A further 17 of the ngrams do occur, but not as constituents, which would make reaching the correct decision even more difficult for the parser. In order to fix these problems, an outside source of information must be consulted, as the lexical information is currently not available.

The next largest source of errors is mislabelling the bracket itself. In particular, distinguishing between using NP and NML labels, as well as ADJP and JJP, accounts for 75 of the 92 errors. This is not surprising, as we noted during the preparation of the corpus that the labels of some NPs were inconsistant (Vadas and Curran, 2007a). The previous relabelling experiment suggests that we should not evaluate the pairs of labels equally, meaning that the best way to fix these errors would be to change the training data itself. This would require alterations to the original Penn Treebank brackets, which is not feasible here.

Conjunctions are another significant source of errors, and are quite a difficult problem. This is because coordinating multi-token constituents requires brackets around each of the constituents, as well as a further bracket around the entire conjunction. Getting just a single decision wrong can mean that a number of these brackets are in error.

Another notable category of errors arises from possessive NPs, which always have a bracket placed around the possessor in our annotation scheme. The parser is not very good at replicating this pattern, perhaps because these constituents would usually not be bracketed if it weren't for the possessive. In particular, NML nodes beginning with a determiner are rare, only occurring when a possessive follows.

The parser also has difficulty in replicating the constituents around speech marks and brackets. We suspect that this is due to the fact that Collins' model does not generate punctuation as it does other constituents. There is also less need for speech marks and brackets to be correct, as the standard evaluation does not find an error when they are placed in the wrong constituent. The justification for this is that during the annotation process, they were given the lowest priority, and are thus inconsistant.

There are a number of NML and JJP brackets in the parser's output that are clearly incorrect, either because they define right-branching structure (which is not bracketed explicitly) or because they dominate only a single token. Single token NMLs exist only in conjunctions, but unfortunately the parser is too liberal with this rule.

The final major group of errors are structural; that is, the entire parse for the sentence is malformed, as in the example where `figures` is actually a noun.

From this analysis, we can say that the modifier attachment problem is the best to pursue. Not only is it the largest cause of errors, but there is an obvious way to reduce the problem: find and make use of more data. This data does not need to be annotated, as we demonstrated in previous NP bracketing experiments (Vadas and Curran, 2007b), which attained a positive result. However, incorporating the data into Collins' model is still difficult. Our previous work only implemented a post-processor that ignored the parser's output. There is still room for improvement in this area.

4 Bracket Structure

We have now seen how a Collins-style parser performs on internal NP structure, but a question remains about the structure itself: is it optimal for the parser? It may be argued that a better representation is to explicitly bracket right-branching structure. For example, in the NP `the New York Stock Exchange`, if there was a bracket around `New York Stock Exchange`, then it would be useful training for when the parser comes across `New York Stock Exchange composite trading` (which it does quite often). The parser should learn to add a bracket in both cases. Explicit right-branching brackets could also remove some of the clearly wrong bracket errors in Table 5.

	PREC.	RECALL	F-SCORE
Overall	87.33	86.36	86.84
Original structure	87.96	88.06	88.01
NML and JJP brackets only	82.33	74.28	78.10

Table 6: Explicit right-branching structure

The current bracketing guidelines do not mark right-branching constituents, they are simply assumed implicitly to be there. We can automatically add them in however, and then examine what difference this change makes. We find, in Table 6, that overall performance drops by 1.51% F-score.

This was surprising result, as there are a number of easily recoverable brackets that are introduced by making right-branching structure explicit. For example, a POS tag sequence of DT NN NN is always right-branching. This explains the more than 10% increase in F-score when evaluating internal NP brackets only. As Rehbein and van Genabith (2007) found, increasing the number of non-terminal nodes has caused an increase in performance, though we may question, as they do, whether performance has truly increased, or whether the figure is simply inflated by the evaluation method.

On the other hand, the increased number of brackets has had a deleterious effect on the original brackets. This result suggests that it is better to leave right-branching structure implicit.

5 Appositions

The results in this paper so far, have been attained using noun modifier data. This final set of experiments however, focuses upon a different kind of internal NP structure: appositions. We thus show that the effects of treebank structure are important for a wider range of constructions, and demonstrate once again the difficulty experienced by a model that must adapt to altered data.

Appositions are a very common linguistic construction in English. They have been used in areas such as Information Extraction (Sudo et al., 2003) and Question Answering systems (Moldovan et al., 2003), however there is little work on automatically identifying them. Researchers have typically used simple patterns for this task, although the accuracy of this method has not been determined. We will compare how well Bikel's parser performs.

As in our previous experiments, we use the Penn Treebank, as it contains numerous appositions, such as the (slightly edited) example shown below:

```
(NP-SBJ
  (NP (NNP Darrell) (NNP Phillips) )
  (, ,)
  (NP (NN vice) (NN president) ))
```

Appositional structure is, of course, not annotated in the Penn Treebank, and so we manually added gold-standard apposition brackets to the corpus. This was actually done during the annotation process for the Vadas and Curran (2007a) data. For example, we add a new bracket labelled APP to the previous noun phrase:

```
(NP-SBJ
  (APP
    (NP (NNP Darrell) (NNP Phillips) )
    (, ,)
    (NP (NN vice) (NN president) )))
```

We should note that we only look at *nonrestrictive* apposition, i.e. where NPs are separated by punctuation (usually a comma, but also a colon or dash). Cases of *close* apposition, as in `the vice president Darrel Phillips` have been shown to have a different interpretation (Lee, 1952) and also present more difficult cases for annotation.

There are 9,082 APP constituents in the corpus, out of the 60,959 NPs (14.90%) that were manually inspected during the annotation process. We measured inter-annotator agreement by comparing against a second annotator on Section 23. This resulted in an F-score of 90.41%, with precision of 84.93% and recall of 96.65%. The precision was notably low because the second annotator inserted APP nodes into NPs such as the one shown below:

```
(NP
  (APP
    (NP
      (QP ($ $) (CD 1.7) (CD million) )
      (-NONE- *U*) )
    (, ,)
    (CC or)
    (NP
      (NP (CD 21) (NNS cents) )
      (NP-ADV (DT a) (NN share) ))))
```

and while these cases are appositive, the first annotator did not insert an APP node when a conjunction was present.

	PREC.	RECALL	F-SCORE
Overall	86.24	87.60	86.92
APP brackets only	69.79	66.37	68.04
All other brackets	86.43	87.86	87.14

Table 7: Parser results with appositions

5.1 Experiments

Our first experiment uses a pattern matching technique, simply identifying appositions by looking for a pair of NPs separated by a comma or colon. This rule was then expanded to include other similar constituent labels: NML, UCP, ADJ, NNP, and APP itself, after noticing that errors were occurring in these cases.

Evaluating this approach, using the entire Penn Treebank as a test set, we achieved an F-score of 95.76%, with precision 95.53% and recall 95.99%. This result is very good for such a simplistic approach, and could be improved further by adding some additional patterns, such as when an adverb appears between the comma and the second NP.

Having set this very high baseline, we once again used Bikel's (2004) parser in an attempt to find an improvement. This experiment includes the NML and JJP brackets in the data, and the parser is in its original state, without any of the alterations we made earlier. The results are shown in Table 7.

The parser's performance on APP brackets is almost 30% F-score below the pattern matching approach, and it has also dropped 1.21% counting only the non-APP constituents. The reason for this very low performance arises from the way that Collins' models treats punctuation, i.e. all tokens with a POS tag of . or :. The Collins (1997) model does not generate punctuation at all, and later models still do not treat punctuation the same way as other tokens. Instead, punctuation is generated as a boolean flag on the following constituent. As a result, the parser cannot learn that a rule such as APP $\rightarrow$ NP , NP should have high probability, because this rule is never part of the grammar. For this reason, the parser is unable to replicate the performance of a simple rule.

6 Conclusion

The results of this paper emphasise the strong relationship between a statistical model and the structure of the data it uses. We have demonstrated this by changing two different constructions in the Penn Treebank: noun modifiers, and appositions.

The annotation structure of the Vadas and Curran (2007a) data has also been validated by these results, which is actually a complement for the BioMedical guidelines (Warner et al., 2004), on which ours were based. It is not neccessary to explicitly bracket right-branching constituents, and furthermore, it is harmful to do so. In addition, separating the NML and NP labels is advantageous, although our results suggest that performance would increase if the original Treebank annotations were made more consistent with our internal NP data.

Our results also demonstrate the neccessity of having a base-NP submodel as part of Collins' models. This specialised case, while seeming unneccesary with our new internal NP brackets, is still required to attain a high level of performance.

Instead, the error analysis in Section 3.3 shows us the true reason why the parser's performance on internal NP brackets is low. NML and JJP brackets are difficult because they require specific lexical information, i.e. the exact words must be in the training data. This is because POS tags, which are very important when making most parsing decisions, are uninformative here. For example, they do not help at all when trying to determine whether a sequence of 3 NNs is left or right-branching.

Our previous work in NP bracketing (Vadas and Curran, 2007b) is positive evidence that this is the correct direction, although incorporating such a submodel back into Collins' model in place of the existing NP submodel, would be a further improvement. It also remains to be seen whether the results observed here would apply to other parsing models.

This work has demonstrated the large effect that data structure can have on a standard NLP tool. It is important that any system, not just parsers, ensure that they perform adequately when faced with changing data. Otherwise, assumptions made today will cause problems for researchers in the future.

Acknowledgements

We would like to thank Matthew Honnibal for his annotation of the apposition data; and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Projects DP0453131 and DP0665973.

References

Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* Ph.D. thesis, University of Pennsylvania.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23.

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

Joshua Goodman. 1997. Probabilistic feature grammars. In *Proceedings of the International Workshop on Parsing Technologies.*

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D. Manning. 2001. Parsing with treebank grammars: empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 338–345. Toulouse, France.

Sandra Kübler. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005.* Borovets, Bulgaria.

Donald W. Lee. 1952. Close apposition: An unresolved pattern. *American Speech*, 27(4):268–275.

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

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. COGEX: A logic prover for question answering. In *Proceedings of HLT-NAACL 2003*, pages 166–172.

Ines Rehbein and Josef van Genabith. 2007. Treebank annotation schemes and parser evaluation for German. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 630–639.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL-03)*, pages 224–231.

David Vadas and James R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07).*

David Vadas and James R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-2007).* Melbourne, Australia.

Colin Warner, Ann Bies, Christine Brisson, and Justin Mott. 2004. Addendum to the Penn Treebank II style bracketing guidelines: BioMedical Treebank annotation. Technical report.

Proceedings of the Australasian Language Technology Workshop 2007, pages 117-124

An Empirical Investigation into Grammatically Constrained Contexts in Predicting Distributional Similarity

Dongqiang Yang | David M.W. Powers
School of Informatics and Engineering
Flinders University of South Australia
Po Box 2100, Adelaide 5001, South Australia
{Dongqiang.Yang|David.Powers@flinders.edu.au}

Abstract

The utility of syntactic dependencies in computing distributional similarity has not yet been fully investigated. Most research based on syntactically conditioned co-occurrences simply ignores the salience of grammatical relations and effectively merges syntactic dependencies into one 'context'. Through calculating distributional similarity, we design two experiments to explore and evaluate the four major types of contexts that are conditioned on grammatical relations. The consistent results show that the head-modifier dependency plays an important role in predicting the semantic features of nouns and verbs, in contrast to other dependencies.

1 Introduction

The roles of grammatical relations in predicting semantic similarity via distributional similarity have not been fully analysed. Most approaches simply chained these syntactic dependencies into one unified context representation for computing distributional similarity, such as in Word Sense Disambiguation (WSD) (Yarowsky, 1993; Lin, 1997; Resnik, 1997), word sense induction (Pantel and Lin, 2002), automatic thesaurus construction (Grefenstette, 1992; Lin, 1998; Curran, 2003), finding the predominant sense (McCarthy et al., 2004), etc.

It is clear that these approaches weighed each dependency through its frequency statistics, e.g. in the weighted (Grefenstette, 1992) or mutual information based (Lin, 1998) Jaccard's coefficient. Although they proposed to replace the unordered context with the syntactically conditioned one, the linguistic specificity of grammatical relations in semantics prediction is often overlooked. Except for the extraction of syntactically conditioned contexts, they in fact make no differentiation between grammatical relations, which work analogously as computing distributional similarity with unordered contexts. Without distinguishing the linguistic specificity of grammatical relations, the advantage of using the syntactic constrained context has not yet been fully exploited when yielding statistical semantics from word distributions. Our goal is thereof to study the salience of these syntactic dependencies in regulating statistical semantics, which can improve the acquisition of semantic knowledge in the Vector Space Model (VSM).

2 Related work

Padó and Lapata (2007) attempted to investigate the role of each single type of syntactic dependency in their syntactically conditioned VSM. They assumed a direct dependency as an undirected path (with a length of 1) in the graph of syntactic dependencies. In addition to this, they experimented a predefined (oblique) weighting scheme (Keenan and Comrie, 1977) in ranking dependencies, i.e. subject to verb: 5, object to verb: 4, prepositional phrase to verb: 3, etc. The optimal VSM they derived was equipped with inversely weighting dependencies within the path length less than 3, rather than this predefined scheme.

Although they investigated a commonly adopted case of syntactic dependencies with the path length equal to 1, the mapping function for reducing data sparseness and dimensionality of their VSM, e.g. congregating any paths ending with the same word, has obscured distinguishing the dependences in predicting semantic similarity. Their work has not completely shown to what extent one single type of syntactic dependency can contribute to statical semantics.

Another similar work was conducted by Plas and Bouma (2005) in enriching Dutch EuroWordNet through clustering distributionally similar words. They investigated the major types of grammatical relationships for nouns in Dutch, and found the predicate-object relation performing best against others such as subject-predicate and adjective-noun. Hoverer, the dependencies exposed to verbs has not been explored.

The goal of our work is to explore the utility of the major types of grammatical relations in predicting semantic similarity. Accordingly, distributional similarity is computed directly from each individual syntactic set rather than on a subtractive or additive fusion. To derive German semantic verb classes with distributional grammatical relations, Schulte im Walde (2006) uses additive fusion to merge syntactic and semantic features including pure verb subcategorization frames, prepositional preferences, and selectional preferences one-by-one into a final verb representation (on the condition that the features have been thoroughly studied in verb semantics). Since the distributional features of individual dependency set has not yet been fully explored, we will not go to seeking for the prime word representation through the subtractive or additive fusion, which could be the next phase of our work.

In the following, we first describe how to give rise to word representation using syntactic dependencies. In the two 'gold-standard' datasets, we evaluate each single type of dependency straight through correlating distributional similarity with human judgements. Without the 'gold-standard' data, we then employ automatic thesaurus construction to evaluate these dependencies in lexical acquisition.

3 Syntactic dependency

Word meaning can be represented as a function of co-occurrence frequencies within different contexts, and similar words share similar contexts (Harris, 1985). In a VSM, the dimensionality of a semantic space can be syntactically conditioned (i.e. syntactic dependencies) or unconditioned (i.e. a bag of words). Different methodologies of distributional similarity under these two context settings, have been systematically surveyed, e.g. for a bag of words (Sahlgren, 2006) and for syntactic dependencies (Curran, 2003; Weeds, 2003). Moreover, the difference between the two kinds of contexts is also contrasted in a framework (Padó and Lapata, 2007), with a preliminary conclusion that the syntactically conditioned VSM outperformed the unconditioned one.

Instead of arguing the states and advantages of these context representations in applications, we focuses on the roles of major types of grammatical relations in the syntactic constrained VSM. The major types of these relations mainly embodied either in head-modifier, i.e. adjective to noun (**AN**) and adverb or the nominal head in a prepositional phrase to verb (**RV**) or in grammatical roles of verb-object (**VO**) and subject-verb (**SV**). The premises mainly rely on the following: (1) the meaning of a noun could depend on its modifiers such as adjectives, nouns, and the nominal head in a prepositional phrase as well as the grammatical role of a noun in a sentence as a subject or object; and (2) the meaning of a verb could be determined by its direct object, subject, or the head of a prepositional phrase.

3.1 Classification and parsing

To capture these relations accurately we employ a widely used and freely available parser based on link grammar (Sleator and Temperley, 1991).

In Link Grammar each word is equipped with 'left-pointing' and/or 'right-pointing' connectors. Based on the crafted rules of the connectors in validating word usages, a link between two words can be formed in reflecting a dependency relation. Apart from these word rules, 'crossing-links' and 'connectivity' are the two global rules working on interlinks, which respectively restrict a link from starting or ending in the middle of pre-existed links and force all the words of a sentence to be traced along links.

There are in total 107 major link types in the Link Grammar parser (ver. 4.1), whereas there are also various sub-link types that specify special cases of dependencies.

Using this parser, we extracted and classified the following link types into the four main types of dependencies:

- **RV**
  1. *E*: verbs and their adverb pre-modifiers
  2. *EE*: adverbs and their adverb pre-modifiers
  3. *MV*: verbs and their post-modifiers such as adverbs, prepositional phrase

- **AN**

1. *A*: nouns and their adjective pre-modifiers
2. *AN*: nouns and their noun pre-modifiers
3. *GN*: common nouns and their proper nouns e.g. Prime Minister Howard.
4. *M*: nouns and their various post-modifiers such as prepositional phrases, adjectives, and participles

- **SV**

1. *S*: subject-nouns/gerunds and their finite verbs. There are also some sub-link types under S, for example, *Ss*g* stands for gerunds and their predicates, and *Sp* plural nouns and their plural verbs
2. *SI*: the inversion of subjects and their verbs in questions

- **VO**

1. *O*: verbs and their direct/indirect objects
2. *OT*: verbs and their time objects
3. *P*: verbs and their complements such as adjectives and passive participles

Note that except for **RV**, we define the **AN**, **SV**, and **VO** dependencies almost identically to shallow parsers (Grefenstette, 1992; Curran, 2003), or a full parser of MINIPAR (Lin, 1998) but we retrieve them instead through the Link Grammar parser.

Given different methodologies to implementing parsing, it is hardly fair to justify a syntactic parser. Molla and Hutchinson (2003) compared the Link Grammar (LG) parser and the Conexor Functional Dependency Grammar (CFDG) parser with respect to intrinsic and extrinsic evaluations. In the intrinsic evaluation the performance of the two parsers was compared and measured in terms of the precision and recall of extracting four types of dependencies, including subject-verb, verb-object, head-modifier, and head-complement. In the extrinsic evaluation a question-answering application was used to contrast the two parsers. Although the LG parser is inferior to the CFDG parser in locating the four types of dependencies, they are not significantly different when applied in question answering. Given that our main task is to study the difference of the syntactic dependencies: **RV**, **AN**, **SV**, and **VO**, acquired with the same LG parser, in predicting semantics, it is appropriate to use the LG parser to extract these dependencies.

3.2 Matrix construction

After parsing the 100 million-word British National Corpus (BNC) and filtering out non-content words and morphology analysis, we separately extracted and clustered the relationships to construct 4 parallel raw matrixes Xs (co-occurrence sets) in terms of the 4 syntactic dependencies above (hereafter the syntactically conditioned co-occurrences, denoted as R_X: $\mathbf{RV}_X$, $\mathbf{AN}_X$, $\mathbf{SV}_X$, and $\mathbf{VO}_X$). The row vectors of R_X denoted respectively $\mathbf{Rv}_X$, $\mathbf{An}_X$, $\mathbf{Sv}_X$, and $\mathbf{Vo}_X$, whereas the column vectors of R_X are denoted as $\mathbf{rV}_X$, $\mathbf{aN}_X$, $\mathbf{sV}_X$, and $\mathbf{vO}_X$ respectively.

The four matrices treat contexts with semantic contents in the frame of the syntactic dependencies. These additional constraints yield rarer events than word co-occurrences in a bag of words. The four syntactic matrices are extremely sparse with nulls in over 95% of the cells. However, they impose more accurate or meaningful (grammatical) relationships between words providing the parser is reasonable accurate. Instead of eliminating the triples with lower frequencies, we kept all co-occurrences to avoid worsening data sparseness.

3.3 Dimensionality reduction

We first substituted the frequency of cell $X_{i,j}$—freq($X_{i,j}$) with its information form using log(freq($X_{i,j}$)+1) to retain sparsity ($0 \rightarrow 0$). It can produce "a kind of space effect" (Landauer and Dumais, 1997) that can lessen the gradient of the frequency-rank curve in Zipf's law (1965), reducing the gap between rarer events and frequent ones.

We then applied Single Value Decomposition (SVD) to smooth the matrices and reduce their dimensionalities to 250, commonly adopted in NLP or LSA (on the word by document matrix). We do not normalize the documents by document entropy as we are not dealing with whole documents but small contexts.

In effect, we map a word-by-word matrix into two word-by-concept (uncorrelated component) matrices after SVD. Consider $\mathbf{SV}_X$ a *m* by *n* matrix representing subject-verb dependencies between *m* subjects and *n* verbs. The **SV** relation can be demonstrated by either using the rows ($\mathbf{Sv}_X$ or $\{X_{i,*}\}$) of $\mathbf{SV}_X$ corresponding to nouns conditioned as subjects of verbs in sentences, or the columns ($\mathbf{sV}_X$ or $\{X_{*,j}\}$) to verbs conditioned by nouns as subjects. The cell $X_{i,j}$ shows the frequency of the *i*th subject with the *j*th verb. The *i*th row $X_{i,*}$ of $\mathbf{SV}_X$ is a profile of the *i*th subject in terms of its all verbs and the *j*th column $X_{*,j}$ of **SV**$_X$ profiles the *j*th verb versus its subjects. We represent the **SV** relation respectively using the rows (**Sv**$_X$ or $\{X_{i,*}\}$) of **SV**$_X$ corresponding to nouns conditioned as subjects of verbs in sentences (*m* by 250 after SVD), and the columns (**sV**$_X$ or $\{X_{*,j}\}$) to verbs conditioned by nouns as subjects (*n* by 250 after SVD).

With respect to the mutual effect of the dependencies on words, the distributional features of nouns mainly focus on **aN**$_X$, **An**$_X$, **vO**$_X$, and **Sv**$_X$, whereas the verbs focus on **Vo**$_X$, **rV**$_X$, and **sV**$_X$. Distributional similarity can be evaluated on these dependency sets.

Note that we also concatenated these dependency sets into one united set (denoted as **All**$_X$) respectively for nouns and verbs, which indicates the common case of combining all dependencies in computing distributional similarity. **All**$_X$ also functioned as a baseline in the following evaluations.

We consistently employed the cosine similarity of word vectors as used in LSA and commonly adopted in assessing distributional similarity. Our contribution is to explore and contrast the semantic features of different syntactic dependencies consistently with one similarity method—the *cosine*, rather than to compare different distributional similarity measures with one united syntactic structure that combines all the dependencies together. Although taking into account more similarity measures in the evaluations can solidify conclusions, this would take us beyond the scope of the work.

4 Human similarity judgement

Rubenstein and Goodenough , in an experiment of investigating distributional similarity, constructed an evaluation dataset with word pairs and their semantic similarity scores. They hired 51 college undergraduates divided into two groups to measure 65 pairs of nouns with the similarity score ranging from 0 to 4. The higher the similarity number, the more similar the nouns were in their meanings. Many researchers (cf. Budantisky and Hirst (2006) and Pedersen et al. (2004) for some popular taxonomy similarity methods) validated semantic similarity methods using the human *group* similarity judgments on the standard dataset of the 65 noun-pairs.

Another source available is provided by Yang and Powers (2006) in their verb similarity work, where 130 pairs of verbs were scored by 6 subjects with a Likert scale from 0 to 4 (from non-similar to nearly synonymous). This dataset was acquired through the analogous instruction in the 65 noun-pairs similarity judgement.

Instead of answering if two words are synonymous or not, we compare to what extent distributional similarity derived from each dependency set correlate well with the human judgements on these 65 (noun) and 130 (verb) pairs. Finkelstein et al. (2002) created another dataset—a large volume of 353 word pairs. But these pairs are not strictly rated with semantic similarity rather than with word association strength, for example there are many word associations such as *Maradonna-football* and *FBI-investigation*. Therefore, we did not attempt to include it to evaluate distributional similarity.

4.1 Results

In this task, we tested distributional similarity (the *cosine*) respectively on the 65 noun-pairs with four sub-syntactic sets: **aN**$_X$, **An**$_X$, **vO**$_X$, and **Sv**$_X$ where nouns are mainly represented, as well as on the 130 verb pairs with the three sub-syntactic sets: **Vo**$_X$, **rV**$_X$, and **sV**$_X$ where verbs as row vectors can be represented with their objects, modifiers, and subjects.

	aN$_X$	**An**$_X$	**vO**$_X$	**Sv**$_X$	**All**$_X$	*Sim*$_{WN}$
r	**0.73**	0.63	0.47	0.41	0.62	0.90
ρ	**0.72**	0.63	0.43	0.38	0.68	0.85

(a) The correlations on '65 nouns'

	rV$_X$	**Vo**$_X$	**sV**$_X$	**All**$_X$	*Sim*$_{WN}$
r	**0.59**	0.49	0.41	0.57	0.84
P	**0.51**	0.44	0.38	0.53	0.77

(b) The correlations on '130 verbs'

Table 1: The value/rank correlation (r/ρ) on the syntactically conditioned dependencies

After calculating the cosine similarity of two word vectors in each subset, we then computed Pearson's correlation (r) and Spearman's correlation (ρ) between human average scores and distributional similarity (the *cosine*) scores. The results in different sub-synsets are shown in Table 1. Note that in Table 1 we also listed the taxonomy-based similarity measures proposed by Yang and Powers (2005; 2006), shortened for *Sim*$_{WN}$ that is based on a lexical knowledge base (WordNet) and can be referred in the next section. *Sim*$_{WN}$ can be taken as the upper bands for the 2 tasks, because Yang and Powers results on both ‘130 verbs’ and ‘65 nouns’ were competitive against others popular methods coded in the WordNet similarity package (Pedersen et al., 2004).

4.2 Discussion on the noun task

Note that unless otherwise specified we ran the paired T-test at the significance level of $\alpha = 0.05$ in the following sections. As to the ‘65’ dataset in Table 1-(a), distributional similarity in **aN**$_X$ with correlations over 72% predicted more accurate semantic similarity than the other three subsets: **An**$_X$, **Sv**$_X$, and **vO**$_X$. Nonetheless, **aN**$_X$ only significantly outperformed **Sv**$_X$ and **vO**$_X$ rather than **An**$_X$. Note that **An**$_X$ was significantly better in correlating with human judgments than **vO**$_X$ but not **Sv**$_X$. Both **Sv**$_X$ and **vO**$_X$ performed on a par without significant difference. The multiple linear regression shows that the combined model with **aN**$_X$, **An**$_X$, **vO**$_X$, and **Sv**$_X$ ($r = 0.74$) was significantly better than guessing the mean ($F = 15.394$, $p < 0.001$), where **An**$_X$, **vO**$_X$, and **Sv**$_X$ contributed little to the linear combination ($p > 0.05$) and **aN**$_X$ was the only significant contributor to the model ($p < 0.001$).

In contrast to the upper band of Sim_{WN}, an approach to taxonomic similarity, distributional similarity on **aN**$_X$, **An**$_X$, **vO**$_X$, and **Sv**$_X$ both significantly underperformed Sim_{WN} in correlating with human judgements.

Table 1-(a) also contains the correlation of the baseline **All**$_X$ with human ratings in this task ($r = 0.62$). Without any fusion, distributional similarity on **aN**$_X$ correlated better with human judgments than **All**$_X$, whereas **An**$_X$ performed nearly identically with **All**$_X$.

4.3 Discussion on the verb task

As shown in Table 1-(b), the *cosine* similarity in **rV**$_X$ with the correlation of about 60% predicted relatively more accurate semantic similarity than other two subsets: **sV**$_X$ and **Vo**$_X$, but the differences in their correlations were not significant. With the multiple linear regression on **Vo**$_X$, **rV**$_X$, and **sV**$_X$, we observed that 38.4% of variations in human judgement was accounted for in the combined model ($F = 26.151$, $p < 0.001$) that strongly correlated with the observed values ($r = 0.62$). Both **rV**$_X$ and **Vo**$_X$ made a significant contribution in the model with the exception of **sV**$_X$.

As for the taxonomic similarity in Table 1-(b), distributional similarities on **Vo**$_X$, **rV**$_X$, and **sV**$_X$ were significantly inferior to Sim_{WN} in terms of correlations with human judgements on the 130 pairs.

With respect to the united dependency set, consisting of **Vo**$_X$, **rV**$_X$, and **sV**$_X$, only **rV**$_X$ performed competitively against the baseline **All**$_X$.

4.4 Frequency bias

Due to the hypothesis of distributional representations, distributional similarity of words should correlate with the common features they share (Harris, 1985). We defined and collected the Intersection Attribute Frequency (**IAF**), which indicates on average how many common attributes any two words share in each dependency set R_X. For the 65 noun pairs, **IAF** on **aN**$_X$ (65.2) was larger than it on **An**$_X$ (49.2), **vO**$_X$ (26.6), and **Sv**$_X$ (20.9), which corresponded well to their orders of the correlations in Table 1-(a). For the 130 verb pairs, **IAF** on **rV**$_X$ (168.9) was greater than it on **Vo**$_X$ (139.1) and **sV**$_X$ (105.1), which tallied with the relatively higher correlation on **rV**$_X$ ($r = 0.59$) than on **Vo**$_X$ ($r = 0.49$) and **sV**$_X$ ($r = 0.41$) in Table 1-(b). This is in accordance with the intuition that the more features words share, the more similar they are, which could account for the difference between the dependencies in predicting semantic features.

5 Thesaurus Construction

Instead of comparing distributional similarity with the ‘gold-standard’ of human similarity judgement, one of the application-style evaluations on distributional similarity is to automatically produce a thesaurus entry for each target word, through which the accuracy of synonyms or near-synonyms captured can indirectly measure the capabilities of the syntactic dependencies in predicting lexical semantics.

The usual way of creating an automatic thesaurus is to extract the top *n* words in the similar word list of each target as the entries of its thesaurus, after calculating and ranking the distributional similarity between the target and all of the other words. The accuracy and coverage of thesauri inevitably depend on the size and domains of the corpora used, as well as the measures of computing distributional similarity.

Given the same distributional similarity (*cosine*) across the dependency sets, the results of thesaurus construction can test semantic constraints of grammatical relations. Instead of a normal thesaurus with a full coverage of PoS tags, we only compile the thesaurus entries of nouns and verbs that account for the major part of published thesauri.

5.1 Candidate words

	Similar words
aN$_X$	*Imprisonment term utterance penalty excommu-nication syllable words punishment prison prisoner phrase detention hospitalisation fisticuffs banish-ment verdict Minnesota meaning adjective warder*
An$_X$	*words syllable utterance clause nictation word swarthiness paragraph text homograph dis-course imprisonment nonce phrase hexagram adjective verb niacin savarin micheas*
vO$_X$	*soubise cybele sextet cristal raper stint concatenation kohlrabi tostada apprenticeship ban contrivance Guadalcanal necropolis misanthropy roulade gasworks curacy jejunum punishment*
Sv$_X$	*ratel occurrence cragsman jingoism shiism Oklahoma genuineness unimportance language gathering letting grimm chaucer accent taxation ultimatum arrogance test verticality habituation*
All$_X$	*Imprisonment utterance penalty excommu-nication punishment prison prisoner detention hospitali-sation banishment Minnesota meaning contrariety phoneme consonant counter-intelligence starvation fine cathedra lifespan*

(a) The similar words to *sentence* (as a noun)

	Similar words
rV$_X$	*assault rape criticize arm slaughter abduct mortar accuse defend fire avow lash badmouth blaspheme slit singe flame kidnap persecute*
Vo$_X$	*raid criticise bomb realign outwit beleaguer guard raze bombard criticize resemble spy pulse misspend reformulate alkalinise meta-stasise placard ruck glory*
sV$_X$	*Ambush invade fraternize palpitate patrol wound pillage bomb billet shell fire liberate kidnap raid garrison accuse assault arrest slaughter outnumber*
All$_X$	*raid bomb assault criticize ambush accuse fire guard bombard patrol rape storm infiltrate wound kidnap criticise garrison alkalinise torture spy*

(b) The similar words of *attack* (as a verb)

Table 2: A sample of the distributional 'thesauri'

We select 100 nouns and 100 verbs with term frequencies of around 10,000 times in BNC. Highly frequent words are likely to be functional words and the less frequent words may not hap-pen in the semantic sets. In fact, the average fre-quency of the nouns in **An**$_X$, **aN**$_X$, **Sv**$_X$, and **vO**$_X$ are respectively about 3400, 5600, 1200, and 1700, and the verbs in **rV**$_X$, **Vo**$_X$, and **sV**$_X$ 3000, 3300, and 2000, as we only extracted syntactic dependencies from BNC.

For a target word in each sub-syntactic set, we produced and ranked the top 20 words as candi-dates for the automatic thesaurus after computing distributional similarity of the target with all other words in each sub-syntactic set. The popu-lation of the nouns or the verbs consists of 2000 words. In Table 2, we exemplify the top 20 simi-lar words of *sentence* (as a noun) and *attack* (as a verb).

5.2 Evaluation

It is not a trivial work to evaluate distributional thesauri in the absence of a benchmark set. After constructing a 'gold standard' dataset consisting of Roget's Thesaurus (1911), Macquarie's The-saurus, and Webster's 7th dictionary, Grefen-stette (1993) evaluated his automatic thesaurus extracted from Grolier's Encyclopaedia using distributional similarity on syntactic dependen-cies. If two words were located under the same topic in Roget or Macquarie, or shared two or more terms in their definitions in the dictionary, they were counted as a successful hit for syno-nyms or semantic-relatedness.

To improve the coverage of the 'gold standard' dataset in the experiment, Curran (2003) incor-porated more thesauri: Roget Thesaurus (both the free version provided by Project Gutenberg (1911) and the modern version of Roget's II), Moby Thesaurus, The New Oxford Thesaurus of English, and The Macquarie Encyclopedic The-saurus.

Instead of simply matching with the 'gold standard' thesauri, Lin (1998) proposed to com-pare the structures of his automatic thesaurus to WordNet and Roget through his taxonomic simi-larity approach, i.e. taking into account the order of the similar words produced from distributional similarity. Inspired by Lin's work (1998), we also defined two different similarity measures to compare the automatic thesaurus with the 'gold standard', i.e. Sim_{WN} for WordNet and Sim_{RT} for Roget. Instead of recording the similarity scores produced in Sim_{WN} and Sim_{RT} we counted the number of similar words within similarity thresholds.

- Sim_{WN}: There are numerous noun similarity methods in the WordNet similarity package of Pedersen et al. (2004). However, since the similarity method proposed by Yang and Powers (2005; 2006) was competitive and also worked on the 130 verb pairs unlike other algorithms, we employed their algo-rithm in the evaluation. Note that their methods were in fact based on edge-counting in the taxonomy of WordNet. In the task, we set up a shorter searching depth limit $\gamma = 4$ for nouns to identify words that are more similar, and $\gamma = 2$ for verbs. If two distributionally similar words are syn/antonym or connected with each other in the taxonomy with the shortest path length less than the depth limit, we counted them as a successful hit, i.e. semantic relatedness.

- *Sim$_{RT}$*: Roget's Thesaurus divides its hierarchy top class to the bottom topic, and stores topic-related words under one of 1000 topics. We counted it a hit if two words are situated under the same topic or the higher level of the same section, i.e. the distance between two words was no more than 2 levels.

5.3 Results

		WordNet						Roget	Total
		SA	*D1*	*D2*	*D3*	*D4*	∑		
Noun	**aN**$_X$	2.8	7.5	10.0	8.2	5.3	33.7	27.5	46.7
	An$_X$	1.5	5.5	9.6	8.6	5.3	30.6	22.3	43.4
	vO$_X$	1.6	4.5	5.9	5.1	4.1	21.2	17.9	33.0
	Sv$_X$	1.1	2.9	4.8	5.0	3.7	17.4	14.1	29.2
	All$_X$	3.0	7.3	11.2	8.7	5.6	36.1	30.1	46.9
Verb	**rV**$_X$	5.3	16.5	13.7			35.5	31.1	46.2
	Vo$_X$	4.1	13.8	13.3			31.1	26.9	43.4
	sV$_X$	2.7	9.6	12.0			24.3	24.1	37.7
	All$_X$	4.0	20.0	12.8			36.7	30.2	47.9

Table 3: The evaluation results of noun and verb thesauri.

The results of our automatic thesauri for the nouns and verbs in the sub-syntactic sets are listed in Table 3. For *Sim$_{WN}$*, *SA* denotes the accuracy on the syn/antonyms of the targets, and *DI* the accuracy on the words with exactly *I* link distance to targets (for nouns $I \leq \gamma = 4$; for verbs $I \leq \gamma = 2$); ∑ denotes the overall accuracy. For *Sim$_{RT}$*, Roget indicates the overall accuracy in Roget, and Total the overall accuracy in both WordNet and Roget.

5.4 Discussion

In Table 3 both the noun thesaurus from **aN**$_X$ and the verb thesaurus from **rV**$_X$ achieved the highest overall accuracy in WordNet, Roget, and Total. The paired-sample T-test on the accuracy of each target in each sub-syntactic set showed that (1) distributional similarity extracted significantly more similar nouns from **aN**$_X$ than other three dependency sets: **An**$_X$, **vO**$_X$, and **Sv**$_X$, and from **An**$_X$ than the other two sets: **vO**$_X$ and **Sv**$_X$; (2) there were not significant difference between **rV**$_X$ and **Vo**$_X$ in retrieving real similar verbs through distributional similarity, but both of them were significantly better than **Sv**$_X$.

The baseline **All**$_X$, incorporating more grammatical relations into one representation, i.e. **aN**$_X$, **An**$_X$, **vO**$_X$, and **Sv**$_X$ for nouns and **Vo**$_X$, **rV**$_X$, and **sV**$_X$ for verbs, retrieved more synonyms or near-synonyms in its automatic thesauri than other single dependency set. The advantage of **All**$_X$ against others is not a surprise given the syntactic dependencies it combined. However **All**$_X$ vs. **aN**$_X$ and **rV**$_X$ shows no significant discrepancy on accuracy, which also implied the strength of the head-modifier relations on dominating lexical semantics.

We further varied the threshold from 20 to 50 words increasingly with 10 words to study the effect of the size of term clusters on accuracy. We found that the results were similar, and the drop of the overall accuracy of nouns and verbs was on average 4% and not significant ($p < 0.05$).

These homogeneous results in retrieving semantically similar or related words, together with those in judging semantic similarity, indicated that the head-modifier relations strongly correlates with semantic properties for nouns and verbs.

5.5 Frequency bias

As indicated in the previous evaluation, we also collected the **IAF** statistics of 2,000 noun and 2,000 verb pairs in these dependency sets, which can signify to what extent two words share common distributional structures in each dependency set. The highest **IAF** 135.4 in **aN**$_X$ (respectively 92.4 in **An**$_X$, 35.9 in **vO**$_X$, and 28.1 in **Sv**$_X$) and 87.6 in **rV**$_X$ (53.1 and 45.8 in **Vo**$_X$ and **sV**$_X$) corresponds to the highest accuracy of each dependency set in yielding automatic thesaurus construction. These results were consistent with them in the relatively small data sets of 65 noun-pairs and 130 verb-pairs from the previous section, where **IAF** is proportional to the correlations of distributional similarity on each type of grammatical relations with human similarity judgements.

6 Conclusion

Through human similarity judgements and automatic thesaurus construction, we study the major types of syntactic dependencies in expressing semantic salience. The consistent results show that semantic features of nouns and verbs are most strongly characterised by the head-modifier relations. The distinctive linguistic features of these syntactic dependencies provide an empirical basis for how to better model word meanings. Our future work would be to fuse these features in the distributional representation of words, and tailor them for specific applications.

References

Budantisky, Alexander and Graeme Hirst (2006). Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics 32(1): 13-47.

Curran, James R. (2003). From Distributional to Semantic Similarity. Ph.D thesis

Finkelstein, Lev, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman and Eytan Ruppin (2002). Placing Search in Context: The Concept Revisited. ACM Transactions on Information Systems 20: 116-131.

Grefenstette, Gregory (1992). Sextant: Exploring Unexplored Contexts for Semantic Extraction from Syntactic Analysis. In the 30th Annual Meeting of the Association for Computational Linguistics, 324-326. Newark, Delaware.

Grefenstette, Gregory (1993). Evaluation Techniques for Automatic Semantic Extraction: Comparing Syntactic and Window Based Approaches. In the Workshop on Acquisition of Lexical Knowledge from Text, 143-153.

Harris, Zellig (1985). Distributional Structure. In The Philosophy of Linguistics J. J. Katz, (ed). New York, Oxford University Press: 26-47.

Keenan, Edward and Bernard Comrie (1977). Noun phrase accessibility and universal grammar. Linguistic Inquiry 8: 62-100.

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

Lin, Dekang (1997). Using Syntactic Dependency as a Local Context to Resolve Word Sense Ambiguity. In the 35th Annual Meeting of the Association for Computational Linguistics, 64-71. Madrid, Spain.

Lin, Dekang (1998). Automatic Retrieval and Clustering of Similar Words. In the 17th International Conference on Computational Linguistics, 768-774. Montreal, Quebec, Canada.

McCarthy, Diana, Rob Koeling, Julie Weeds and John Carroll (2004). Finding Predominant Senses in Untagged Text. In the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), 267-287. Barcelona, Spain.

Molla, Diego and Ben Hutchinson (2003). Intrinsic versus Extrinsic Evaluations of Parsing Systems. In European Association for Computational Linguistics(EACL), workshop on Evaluation Initiatives in Natural Language Processing, 43-50. Budapest, Hungary.

Padó, Sebastian and Mirella Lapata (2007). Dependency-based construction of semantic space models. To appear in Computational Linguistics 33(2).

Pantel, Patrick and Dekang Lin (2002). Discovering Word Senses from Text. In the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 613-619. New York, NY, USA.

Pedersen, Ted, Siddharth Patwardhan and Jason Michelizzi (2004). WordNet::Similarity - Measuring the Relatedness of Concepts. In the Nineteenth National Conference on Artificial Intelligence (AAAI-04), 1024-1025. San Jose, CA.

Plas, Lonneke van der and Gosse Bouma (2005). Contexts for Finding Semantically Similar Words. In the 20th International Conference on Computational Linguistics, 173-186. Geneva, Switzerland.

Resnik, Philip (1997). Selectional Preference and Sense Disambiguation. In ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?, 52-57. Washington, USA.

Sahlgren, Magnus (2006). The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-Dimensional Vector Spaces. Ph.D thesis

Schulte im Walde, Sabine (2006). Experiments on the Automatic Induction of German Semantic Verb Classes. Computational Linguistics 32(2): 159-194.

Sleator, Daniel and Davy Temperley (1991). Parsing English with a Link Grammar, Carnegie Mellon University.

Weeds, Julie Elizabeth (2003). Measures and Applications of Lexical Distributional Similarity. Ph.D thesis

Yang, Dongqiang and David M.W. Powers (2005). Measuring Semantic Similarity in the Taxonomy of WordNet. In the Twenty-Eighth Australasian Computer Science Conference (ACSC2005), 315-322. Newcastle, Australia, ACS.

Yang, Dongqiang and David M.W. Powers (2006). Verb Similarity on the Taxonomy of WordNet. In the 3rd International WordNet Conference (GWC-06), 121-128. Jeju Island, Korea.

Yarowsky, David (1993). One Sense per Collocation. In ARPA Human Language Technology Workshop, 266-271. Princeton, New Jersey.

Zipf, George Kingsley (1965). Human Behavior and the Principle of Least Effort: an Introduction to Human Ecology. N.Y., Hafner Pub. Co.

Dictionary Alignment for Context-sensitive Word Glossing

Willy Yap and **Timothy Baldwin**
Department of CSSE
University of Melbourne
{willy,tim}@csse.unimelb.edu.au

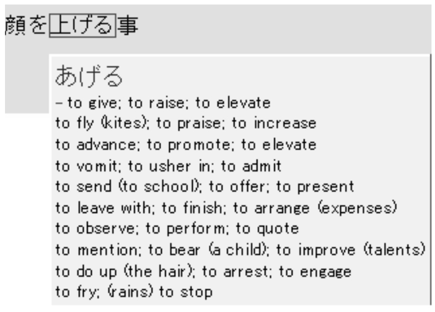
Abstract

This paper proposes a method for automatically sense-to-sense aligning dictionaries in different languages (focusing on Japanese and English), based on structural data in the respective dictionaries. The basis of the proposed method is sentence similarity of the sense definition sentences, using a bilingual Japanese-to-English dictionary as a pivot during the alignment process. We experiment with various embellishments to the basic method, including term weighting, stemming/lemmatisation, and ontology expansion.

1 Introduction

In a multi-lingual environment such as the Internet, users often stumble across webpages authored in an unfamiliar language which potentially contain information of interest. While users can consult dictionaries to help them understand the content of the webpages, the process of looking up words in unfamiliar languages is at best time-consuming, and at worst impossible due to a range of reasons. First, the writing system of the language may be unfamiliar to the user, e.g. the Cyrillic alphabet for a monolingual English speaker. Second, the user may not be familiar with the non-segmenting nature of languages such as Chinese and Japanese, and hence be incapable of delimiting the words to look up in the dictionary in the first place. Third, the user may be unable to lemmatise the word to determine the form in which it is listed in a dictionary.

There are several alternatives to help decipher webpages in unfamiliar languages. The first one is to use an online machine translation system such as Altavista's Babel Fish[1] or Google Translate.[2] While web-based machine translation services occasionally produce good translations for linguistically-similar languages such as English and French, they do not perform very well in translating languages which are removed from one another (Koehn, 2005).


顔を上げる事

あげる
- to give; to raise; to elevate
to fly (kites); to praise; to increase
to advance; to promote; to elevate
to vomit; to usher in; to admit
to send (to school); to offer; to present
to leave with; to finish; to arrange (expenses)
to observe; to perform; to quote
to mention; to bear (a child); to improve (talents)
to do up (the hair); to arrest; to engage
to fry; (rains) to stop

Figure 1: Multiple translations for the Japanese word 上げる [*ageru*] produced by `rikai.com`. The correct translation in this context is "to raise".

The second alternative is a pop-up glossing application. The application takes raw text or a URL, parses the words, and returns the pop-up translation of each word as the mouse hovers over it. Some example pop-up glossing applications for Japanese source text and English glosses are Rikai[3] and POPjisyo.[4] With the aid of these pop-up translations, the manual effort of segmenting words (if necessary) and looking up each can be avoided. This application is also useful as an educational aid for learners of that language.

The drawback with these applications is they display all possible translations of a given word irrespective of context. Faced with the task of determining the correct translation themselves, users frequently misinterpret words. An illustration of this situation is given in Figure 1.

[1] `http://babelfish.altavista.com/`
[2] `http://www.google.com/translate_t`
[3] `http://www.rikai.com/perl/Home.pl`
[4] `http://www.popjisyo.com`

We propose a context-sensitive dictionary glossing application to enhance the utility of on-line glossing applications by sensitising the presented glosses to the context of use. The proposed method works by combining a monolingual word sense disambiguation (WSD) system (Baldwin et al., to appear) with an automatically induced cross-lingual sense alignment table. Based on the prediction(s) of the WSD system, our application presents the corresponding set of context-sensitive glosses to the user dictionary glossing by analysing the output of the alignment process.

This paper focuses on the cross-lingual sense alignment aspect of the application. We take separate sense inventories for two distinct languages (Japanese and English in our case) and align the senses between the two. The basis of the alignment process is overlap in sense definitions. By adjusting a threshold for the required level of match, we are able to adjust the precision and recall of the alignment. In preliminary experimentation, we achieve promising results.

The remainder of this paper is structured as follows. We review previous research on dictionary alignment in Section 2, and outline the various resources we utilise during the alignment process in Section 3. We then describe the proposed basic sense-to-sense alignment method, along with various enhancements (Section 4), and present our experimental method and the results of our experiments (Sections 5 and 6, respectively). Finally we discuss our results and future research in Section 7.

2 Previous Research

There has been a significant amount of research on bilingual dictionary alignment using a third language as a pivot. For example, Shirai et al. (2001) built Japanese–French and Japanese–Korean dictionaries using English as the pivot language. In other research, Paik et al. (2001) used English and Chinese as pivots to generate a Korean–Japanese dictionary: English because of the accessibility of Korean–English and Japanese–English dictionaries, and Chinese because of the high overlap in orthography between Korean and Japanese, based on Chinese hanzi.

There have been numerous attempts to manually develop multilingual resources that include cross-lingual sense alignments (Vossen, 1998; Stamou et al., 2002), and the import of cross-lingual semantic alignment has been ably demonstrated by the high impact of these resources. Due to the high overhead in manually constructing such resources, there have been various attempts at automatic cross-lingual sense alignment. The methods are predominantly corpus-driven, based either on cross-lingual distributional similarity in a comparable corpus (e.g. Ngai et al. (2002)) or word alignment over a parallel corpus (e.g. Gliozzo et al. (2005)).

There is a lesser amount of research on cross-lingually aligning ontologies without using large-scale corpus data, which we discuss in greater detail as it is more closely related to that proposed in this research. Asanoma (2001) aligned the Japanese Goi-Taikei ontology with WordNet by first translating a significant subset of the WordNet synonym sets (synsets) into Japanese, automatically matching these based on (monolingual Japanese) lexical overlap, and "filling in the gaps" for the remaining classes based on their hierarchical positioning relative to the aligned classes. Knight and Luk (1994) aligned Spanish and English senses based on: (1) overlap in sets of translations corresponding to each sense of a given Spanish word, with synsets in WordNet; and (2) domain codes in the Spanish and English ontologies. They additionally aligned monolingual English dictionaries based on overlap in the definitions of each sense. The former cross-lingual case assumes a sense-discriminated bilingual dictionary, which we do not have access to. The latter case is similar to our research in that it compares definition sentences, but differs in that the definitions are in the same language. The most closely related work to our research is that of Nichols et al. (2005), who aligned Lexeed senses with WordNet synsets as a by-product of the Lexeed ontology induction task (see Section 3.1), although they do not provide an explicit evaluation of the Lexeed–WordNet alignment for direct comparison.

3 Resources

In this section, we review the key resources used in this research.

Word	竜 *ryuu*		
POS	noun		
Sense 1	Lexical-type	`noun-lex`	
	Definition	想像/上/の/動物/ 。 体/は/巨大/な/蛇/に/似/、/ 4 /本/の/足/と/角/を/持つ 。 海中/や/湖/や/沼/に/棲み、/空/に/昇っ/て/雲/を/起こし/雨/を/降ら/せる/と/言う。	
		An imaginary animal. Dragons are like enormous snakes with 4 legs and horns. Dragons live in the sea, lakes and ponds, and are said to form clouds and cause rain when they fly up into the sky.	
	Hypernym	ANIMAL	
Sense 2,3,4	[...]		
Sense 5	Lexical-type	`noun-lex`	
	Definition	将棋/で/、/飛車/の/成っ/た/もの。	
		In shogi, a promoted rook.	
	Domain	SHOGI	

Figure 2: A partial view of the Lexeed entry for 竜 [*ryuu*] (with English glosses)

3.1 The Lexeed semantic database of Japanese

The Lexeed Semantic Database of Japanese is a machine-readable dictionary consisting of the most commonly-used words in Japanese (Kasahara et al., 2004). In total, there are 28,000 words in Lexeed, and a total of 46,437 senses. Associated with each sense is a set of definition sentences, constructed entirely using the closed vocabulary of the 28,000 words found in Lexeed, such that 60% of the 28,000 words occur in the definition sentences (Tanaka et al., 2006). In addition to the definition sentences, Lexeed also contains part of speech (POS), lexical relations between the senses (if any) and an example sentence, also based on the closed vocabulary of 28,000 words. All content words in the definition and example sentences are sense annotated.

Automatic ontology acquisition methods have been applied to Lexeed to induce lexical relations between sense pairs, based on the sense-annotated definition sentences (Nichols et al., 2005) and comparison with both the Goi-Taikei thesaurus and WordNet 2.0.

An example Lexeed entry for the word *ryuu* is given in Figure 2.

3.2 EDICT

EDICT is a free machine-readable Japanese-to-English dictionary (Breen, 1995). The project is highly active and has been extended to other target languages such as German, French and Russian. EDICT contains more than 170,000 Japanese entries, each of which is associated with one or more English glosses. It also optionally contains information such as the pronunciation of the entry, POS, and domain of application.

3.3 WordNet

WordNet is an electronic semantic lexical database of English (Fellbaum, 1998). It is made up of more than 100,000 synsets, with each synset representing a group of synonyms. Its entries are categorised into four POS categories: nouns, verbs, adjectives and adverbs. Each POS is described in a discrete lexical network.

Every synset in WordNet has a definition sentence, and sample sentence(s) are provided for most of the synsets; in combination, these are termed the WordNet gloss. Semantic relations connect one synset to another, and include relation types such as hypernym, hyponymy, antonymy and meronymy. The majority of these relations do not cross POS boundaries.

Since we only experiment with hypernyms (and, symmetrically, hyponyms), we provide a simple review of this relation. A synset A is a hypernym of a synset B iff B is a kind of A. For example, *vehicle* is a hypernym of *car*, while *perceive* is a hypernym of *hear, sight, touch, smell, taste*.[5]

[5]Strictly speaking, *hear*, etc. are troponyms of *perceive*, i.e. they denote specific ways of *perceiving*. Because WordNet

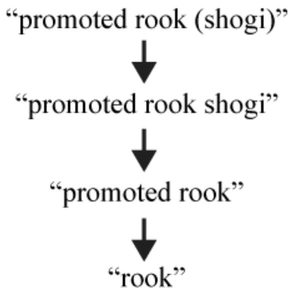


Figure 3: Example of normalisation of the translation string; we stop at "rook" as WordNet has a matching entry for it

When building the baseline for our evaluation, we used the SemCor corpus—a subset of the Brown corpus annotated with WordNet senses—to derive the frequency counts of each WordNet synset (Landes et al., 1998). Section 5 discusses this process in more detail.

4 Proposed Methods

Our basic alignment method, along with various extensions, is outlined below.

4.1 Basic alignment method using cosine similarity

In this paper, we align a semantic database of Japanese (Lexeed) with a semantic network of English (WordNet) at the sense level. First, we use Lexeed to find all possible senses of a given word, and retrieve the definition sentences for each.

Since all the definition sentences are in Japanese, we use EDICT as a pivot to convert Lexeed definition sentences into English. In this process, all possible translations of all Japanese words found in the definition sentences are returned, along with their POS classes. For every translation returned, we find entries in WordNet that match the translation and POS category. If there is no match for the given POS, we relax this constraint and search for entries in WordNet that match the translation but not the POS.

Problems arise when WordNet does not have a matching entry for the translation. This situation usually happens when the translation returned by EDICT is comprised of more than one English word. For a Japanese verb, e.g., the English translation in EDICT almost always begins with the auxiliary *to* (e.g. *nomu* is translated as *to drink*). WordNet does not contain a verbal entry for *to drink*, but does contain an entry for *drink*. To handle this case of partial match, we locate the longest right word substring of the EDICT translation which is indexed in WordNet.

A related problem is when the translation contains domain or collocational information in parentheses. For example, *ryuu* is translated as both *dragon* and *promoted rook (shogi)*. The first translation has a matching entry in WordNet but the second translation does not. In this second case, there is no right word substring which matches in WordNet, as we end up with *rook (shogi)* and then *(shogi)*, neither of which is contained in WordNet. In order to deal with this situation, we first normalise the translation strings by removing all the brackets and query WordNet with the normalised string. Should there be a matching entry, we stop here. If not, we then remove all strings between brackets, and apply the longest right word substring heuristic as above. An illustration of this process is given in Figure 3.

In the worst case of WordNet not having a matching entry for any right word substring, we discard the translation.

At this point, we have aligned a given Japanese word with (hopefully) one or more English words, but are still no closer to inducing sense alignment pairs. In order to produce the sense alignments, we generate all pairings of Lexeed senses with WordNet synsets for each WordNet-matched word translation. For each such pair, we compile out the Lexeed definition sentence(s) word-translated into English, and the WordNet glosses, and convert each into a simple vector of term frequencies. We then measure the similarity of each vector pair using cosine similarity. An overview of this alignment process is presented in Figure 4.

4.2 Weighting terms using TF-IDF mechanism

The basic alignment method does not use any form of term weighting, and thus overemphasises common function words such as *the, which* and *and*, and downplays the impact of rare words. As we expect to have a large amount of noise in the word-

doesn't distinguish between hyponyms and troponyms, however, we treat the two identically.

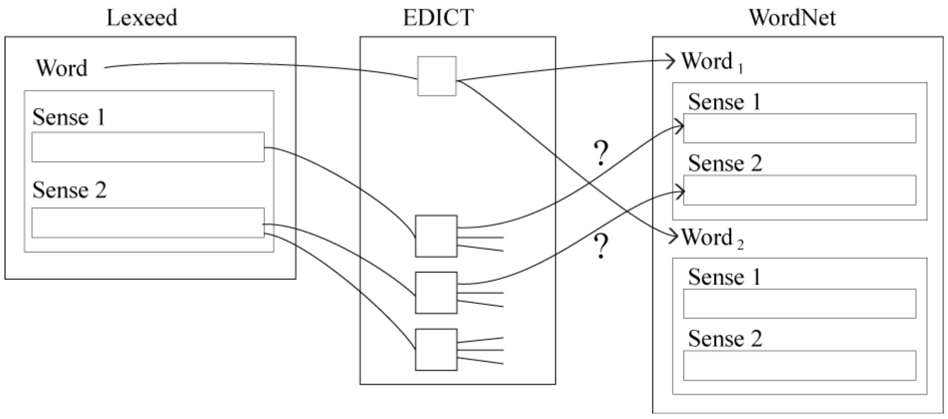


Figure 4: Overview of the Lexeed–WordNet sense alignment method

translated Lexeed definition sentences, including spurious translations for Japanese function words such as *ka, ga* and *no* that have no literal translation in English, we predict that an appropriate form of term weighting should improve the performance of our method.

As a first attempt at term weighting, we experimented with the classic SMART formulation of TF-IDF (Salton, 1971), treating the vector associated with each definition sentence as a single document.

4.3 Word stopping

As mentioned in the previous section, commonly-occurring semantically-bleached words are a source of noise in the naive cosine similarity scoring method. One conventional way of countering their impact is to filter them out of the vectors, based on a stop word list. For our experiments, we use the stop word list provided by the Snowball project.[6]

4.4 POS filtering

Another source of possible noise is the translations of Japanese function words. As all the Lexeed definition sentences are POS tagged, it is a relatively simple process to filter out all Japanese function words, focusing on prefixes, suffixes and particles.

4.5 Lemmatisation, stemming and normalisation

In its basic form, our vector space model treats distinct word as a unique term, including ignoring the obvious similarity between inflectional variants of the same word, such as *dragon* and *dragons*. To remove such inflectional variation, we experiment with lemmatising all words found in both the Lexeed and WordNet vectors, using morph (Minnen et al., 2001). For similar reasons, we also experiment with the Porter stemmer, noting that stemming will further reduce the set of terms but potential introduce spurious matches.

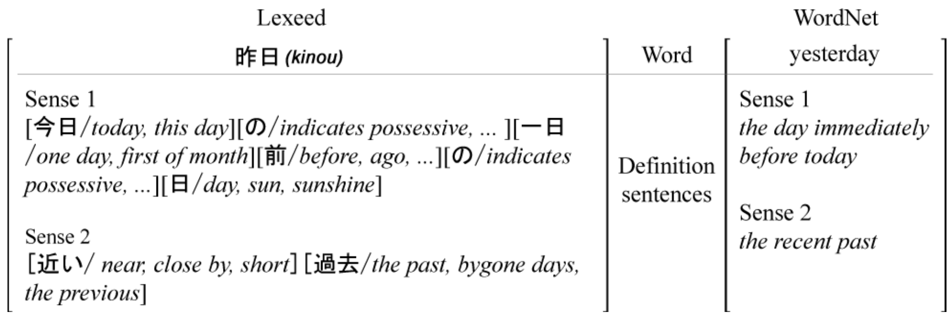
As part of this process (with both lemmatisation and stemming), we remove all punctuation from the definition sentences.

4.6 Lexical relations

Both the Lexeed and WordNet sense inventories are described in the form of hierarchies, making it possible to complement the sense definitions with those from neighbouring senses. The intuition behind this is that the sense granularity in the two sense inventories can vary greatly, such that a single sense in Lexeed is split across multiple WordNet synsets, which we can readily uncover by considering each sense as not a single point in WordNet but a semantic neighbourhood. For example, the second sense of the word *kinou* in Figure 5, which literally means "near past", should be aligned with the second sense of *yesterday*, which is defined as "the recent past". This alignment is more self-evident, however, when we observe that the hypernym of each of the two senses is defined as "past".

In our current experiments, we only look at the utility of hypernymy. For a given sense Lexeed–

[6] `http://snowball.tartarus.org/`



Lexeed	Word	WordNet
昨日 *(kinou)*	Word	yesterday
Sense 1 [今日/*today, this day*][の/*indicates possessive, ...*][一日/*one day, first of month*][前/*before, ago, ...*][の/*indicates possessive, ...*][日/*day, sun, sunshine*] Sense 2 [近い/ *near, close by, short*] [過去/*the past, bygone days, the previous*]	Definition sentences	Sense 1 *the day immediately before today* Sense 2 *the recent past*

Figure 5: The output of word-translating Japanese definition sentences to English

WordNet sense pairing, we extract the hypernyms of the respective senses and expand the definition sentences with the definition sentences from the hypernyms. The term vectors are then based on this expanded term set, similar to query expansion in information retrieval.

5 Experimental Setup

5.1 Gold-standard data

To evaluate the performance of our system, we randomly selected 100 words from Lexeed, extracted out the Lexeed–WordNet sense pairings as described above, and manually selected the gold-standard alignments from amongst them. The 100 words were associated with a total of 268 Lexeed senses and 772 WordNet senses, creating a total of 206,896 possible alignment pairs. Of these, 259 alignments were selected as our gold-standard.

We encountered a number of partial matches that were caused by the Japanese word being more specific than its English counterparts (as identified by our WordNet matching method). For example, *kakkazan* is translated as "active volcano". Since WordNet does not have any entry for *active volcano*, the longest right word substring that matches in WordNet is simply *volcano*. The definition sentences returned by Lexeed describe *kakkazan* as "a volcano which still can erupt" and "a volcano that will soon erupt", while *volcano* is described as "a fissure in the earth's crust (or in the surface of some other planet) through which molten lava and gases erupt" and "a mountain formed by volcanic material". Although there is some similarity between these definitions (namely key words such as *erupt* and *volcano*), we do not include this pairing in our gold-standard alignment data.

5.2 Baseline

As a baseline, we take the most-frequent sense of each of the 100 random words from Lexeed, and match it with the synset with the highest SemCor frequency count out of all the candidate synsets.

5.3 Thresholding

All our calculations are based on cosine similarity, which returns a similarity between 0 and 1, with 1 being an exact match. In its simplest form, we would identify the unique WordNet sense with highest similarity to each Lexeed sense, irrespective of the magnitude of the similarity. This has the dual disadvantage of allowing only one WordNet sense for each Lexeed sense, and potentially forcing alignments to be made on low similarity values. A more reasonable approach is to apply a threshold x, and treat all WordNet senses with similarity greater than x as being aligned with the Lexeed sense. Thresholding also gives us more flexibility in terms of tuning the performance of our method: at higher threshold values, we can hope to increase precision at the expense of recall, and at lower threshold values, we can hope to increase recall at the expense of precision.

5.4 Evaluation metrics

To evaluate the performance of our system, we use precision, recall and F-score. In an alignment context, precision is defined as the proportion of correct alignments to all alignments returned by the system, and recall is defined as the proportion of the correct alignments returned by our system to all the align-

	Basic	Stopping	POS filtering	Lemmatisation	Stemming	Normalisation
Basic model	0.228	0.334	0.256	0.240	0.243	0.221
Basic+TF-IDF	0.292	0.335	0.295	0.344	0.330	0.288

Table 1: Best system F-score of combination of all features using the basic model vs. the basic model with TF-IDF weighting

ments in our gold-standard. F-score is the harmonic mean of precision and recall, and provides a single figure-of-merit rating of the balance between these two factors. We evaluate our system using unbiased ($\beta = 1$) F-score.

6 Results

Throughout our experimentation, we evaluate relative to the 100 manually sense-aligned Japanese words.

Our baseline method predicts 100 alignments (as it is guaranteed to produce a unique alignment per source-language word), of which 60 are correct. Hence, the precision is $\frac{60}{100} = 0.600$, the recall is $\frac{60}{259} = 0.231$, and the F-score is 0.334.

With the basic alignment model, the highest F-score achieved with thresholding is 0.228 at a threshold value of 0.19, well below the baseline F-score. The recall and precision value at this threshold are 0.263 and 0.202, respectively.

The basic model with TF-IDF weighting performed considerably better, scoring the highest F-score of 0.292 (recall = 0.382 and precision = 0.236) at a threshold value of 0.04, but is still well below the baseline F-score. To confirm that TF-IDF term weighting is always beneficial to overall alignment performance, we took the unweighted model combined with each of the proposed extensions, and compared it with the same extension but with the inclusion of TF-IDF (without lexical relations at this point). The result of these experiments can be found in Table 1. As we can see, TF-IDF weighting constantly improves alignment performance. Also note that, with the exception of simple (punctuation) normalisation, all extensions improve over the basic model both with and without TF-IDF weighting.

We extended our experiments by considering all possible combinations of 2 or more proposed extensions (excluding lexical relations for the time being) with TF-IDF weighting. The purpose of this experiment is to investigate whether the proposed extensions are complementary in improving alignment performance. The 5 top-performing combinations are presented in Table 2.

Method	**Precision**	**Recall**	**F-score**
Baseline	0.600	0.231	0.334
WS+PF+L+N	0.298	0.494	0.372
WS+PF+L	0.326	0.428	0.370
WS+L	0.305	0.455	0.365
WS+PF+L+S+N	0.275	0.540	0.364
WS+L+S	0.301	0.455	0.363

Table 2: Top-5 combinations of extensions, excluding lexical relations (WS = Word stopping, PF = POS filtering, L = Lemmatisation, S = Stemming, N = Normalisation)

The best result is achieved by combining all the proposed extensions, at an F-score of 0.364, which is significantly above baseline. It is also interesting to see that not all methods are fully complementary. By excluding stemming, e.g., the system actually performs better, producing a higher F-score of 0.372.

We then experimented with the addition of lexical relations to the different combinations of extensions explored above. The 5 top-performing combinations are presented in Table 3. The best F-score of 0.408 is achieved with the combination of all the extensions proposed. When lexical relations are used exclusively or combined with less than three of the proposed extensions, the performance tends to decline.

In our best performing combination, we outperformed the baseline F-score by 22%. 349 alignments were returned for this F-score, of which 124 matched the gold-standard. The precision and recall scores are 0.355 and 0.478, respectively.

We carried out more detailed analysis of the precision–recall trade-off. While we expect the pre-

Method	Precision	Recall	F-score
Baseline	0.600	0.231	0.334
WS+PF+L+S+N+H	0.355	0.478	0.408
WS+PF+L+S+H	0.344	0.490	0.404
WS+PF+N+S+H	0.342	0.478	0.399
WS+L+S+H	0.361	0.440	0.396
WS+PF+S+H	0.317	0.525	0.396

Table 3: Top-5 performing combinations of extensions, including lexical relations (WS = stopping, PF = POS filtering, L = Lemmatisation, S = Stemming, N = Normalisation, H = Hypernym)

cision to go up to 1 as we increase our threshold, we found out that it is in fact not the case. The precision peaks at 0.625 at a threshold level of 0.265. At this level, there are 10 correct alignments out of 16 alignments returned. Upon investigating the six non-matching entries, we found that they all contain similar words but that the literal meaning of the senses are very different. Below, we present two of the six non-matching entries.

The first one relates to a sense of the Japanese word *shanpuu* "shampoo". The definition sentences for this sense found in Lexeed are directly translated as "shampoo medicine, drug, or dose; detergent or washing material that is used to wash hair or fur". The corresponding match in WordNet is "the act of washing your hair with shampoo". We can see that there are similar terms in the two vectors, such as *shampoo, washing* and *hair*, but that the literal meaning of the two senses is quite different.

The second example is very similar to the *kakkazan* example presented in Section 5. One sense of *sengetsu* ("last month") is defined as "the previous month", and is aligned to the WordNet synset of *month* (WordNet does not have an entry for *last month*). It does not help that the hypernym of *sengetsu* is *tsuki* which translates to "month", boosting the similarity of this alignment.

7 Discussion

In terms of F-score, the best-performing combination of extensions performed better than the baseline. However, the recall seems to be the dominant factor in the F-score calculations for the proposed method. This is in sharp contrast to what we have in our baseline, where precision dominates the F-score calculation. There are several reasons for the baseline scores. First, there are 259 alignments in our gold-standard for 100 random words, corresponding to approximately 2.6 alignment per word. Given how we created our baseline, with one alignment per word, the maximum recall that the baseline can achieve is $\frac{100}{259} = 0.386$.

On the other hand, the first-sense basis of the baseline method leads to high precision, largely due to the design process for ontologies and dictionaries. Namely, there is usually good coverage of frequent word senses in ontologies and dictionaries, and additionally, the translations for a given word are generally selected to be highly biased towards common senses (i.e. even if a polysemous word is chosen as a translation, its predominant sense is almost always that which corresponds to the source language word, for obvious accessibility/usability reasons). For this reason, there is a very high probability that these frequent senses for each of the two languages align with each other.

In this paper, we proposed a cross-lingual sense-to-sense alignment method, based on similarity of definition sentences as calculated via a bilingual dictionary. We explored various extensions to a simple lexical overlap method, and achieved promising results in preliminary experiments.

In future work, we plan to exploit more lexical relations, such as synonymy and hyponymy. We also plan to experiment with weighting up alignments where both the sense pairing and the hypernym pairing match well.

Nichols et al. (2005) linked Lexeed senses to WordNet in their evaluation on ontology induction. Comparison with their method would be very interesting and is an area for future research.

Acknowledgements

We thank Francis Bond for his insights and suggestions for our experiment and Sanae Fujita for her help with the data. We are also grateful to anonymous reviewers for their helpful comments. This research was supported by NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation.

References

Naoki Asanoma. 2001. Alignment of ontologies: WordNet and Goi-Taikei. In *Proc. of the NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 89–94, Pittsburgh, USA.

Timothy Baldwin, Su Nam Kim, Francis Bond, Sanae Fujita, David Martinez, and Takaaki Tanaka. to appear. MRD-based word sense disambiguation: Further$_{\#2}$ extending$_{\#1}$ Lesk. In *Proc. of the Third International Joint Conference on Natural Language Prcoessing (IJCNLP 2008)*, Hyderabad, India.

Jim Breen. 1995. Building an electronic Japanese-English dictionary. In *Japanese Studies Association of Australia Conference*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Alfio Massimiliano Gliozzo, Marcello Ranieri, and Carlo Strapparava. 2005. Crossing parallel corpora and multilingual lexical databases for WSD. In *Proc. of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2005)*, pages 242–5, Mexico City, Mexico.

Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kasunagi, and Shigeaki Amano. 2004. Construction of a Japanese semantic lexicon: Lexeed. In *Proc. of SIG NLC-159, IPSJ*, Tokyo, Japan.

Kevin Knight and Steve K. Luk. 1994. Building a large-scale knowledge base for machine translation. In *Proc. of the 12th Annual Conference on Artificial Intelligence (AAAI-94)*, pages 773–8, Seattle, USA.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the Tenth Machine Translation Summit (MT Summit X)*, Phuket, Thailand.

Shari Landes, Claudia Leacock, and Randes Tengi. 1998. Building semantic concordances. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

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

Grace Ngai, Marine Carpuat, and Pascale Fung. 2002. Identifying concepts across languages: A first step towards a corpus-based approach to automatic ontology alignment. In *Proc. of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan.

Eric Nichols, Francis Bond, and Daniel Flickinger. 2005. Robust ontology acquisition from machine-readable dictionaries. In *Proc. of the 19th International Join Conference on Artificial Intelligence (IJCAI-2005)*, pages 1111–6, Edinburgh, UK.

Kyonghee Paik, Francis Bond, and Shirai Satoshi. 2001. Using multiple pivots to align Korean and Japanese lexical resources. In *Proc. of the NLPRS-2001 Workshop on Language Resources in Asia*, pages 63–70, Tokyo.

Gerald Salton. 1971. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall.

Satoshi Shirai, Kazuhide Yamamoto, and Kyonghee Paik. 2001. Overlapping constraints of two step selection to generate a transfer dictionary. In *ICSP-2001*, pages 731–736.

Sofia Stamou, Kemal Oflazer, Karel Pala, Dimitris Christoudoulakis, Dan Cristea, Dan Tufiş, Svetla Koeva, George Totkov, Dominique Dutoit, and Maria Grigoriadou. 2002. BALKANET: A multilingual semantic network for the Balkan languages. In *Proc. of the International Wordnet Conference*, pages 12–4, Mysore, India.

Takaaki Tanaka, Francis Bond, and Sanae Fujita. 2006. The hinoki sensebank — a large-scale word sense tagged corpus of Japanese —. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*, pages 62–9, Sydney, Australia.

Piek Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Proceedings of the Australasian Language Technology Workshop 2007, pages 134-142

Statistical Machine Translation of Australian Aboriginal Languages: Morphological Analysis with Languages of Differing Morphological Richness

Simon Zwarts
Centre for Language Technology
Macquarie University
Sydney, Australia
szwarts@ics.mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, Australia
madras@ics.mq.edu.au

Abstract

Morphological analysis is often used during preprocessing in Statistical Machine Translation. Existing work suggests that the benefit would be greater for more highly inflected languages, although to our knowledge this has not been systematically tested on languages with comparable morphology. In this paper, two comparable languages with different amounts of inflection are tested, to see if the benefits of morphology used during the translation process, depends on the morphological richness of the language. For this work we use indigenous Australian languages: most Australian Aboriginal languages are highly inflected, where words can take a considerable number of postfixes when compared to Indo-European languages, and for languages in the same (Pama Nyungan) family, the morphological system works similarly. We show in this preliminary work that morphological analysis clearly benefits the richer of the two languages investigated, but is more equivocal in the case of the other.

1 Introduction

The majority of research in the field of Machine Translation (MT) nowadays takes a statistical approach. Morphologically rich languages have some characteristics which make MT hard, particularly in the statistical MT (SMT) context. In one common language group we want to investigate the effect of applying special morphological treatment within SMT for languages with varying degree of morphological richness. Without any morphological preprocessing, individual word counts can be quite low in highly inflected languages, causing more data sparseness than necessary, and ignoring some information which might be useful in Natural Language Processing.

Preprocessing before SMT has been used as a way of improving results. This ranges from basic tokenisation (e.g. separating possessive *'s* on English before training) to extensive syntax-based reordering (e.g. Collins et al. (2005)). Often, the choice of preprocessing proceeds without consideration of the type of language; consider for example recent work on Arabic (Sadat and Habash, 2006), where the various combinations of different preprocessing strategies are systematically worked through, with no particular attention to the characteristics of Arabic.

In most work, there is an intuitive notion that there is a connection between morphological richness of a language and the usefulness of morphological preprocessing. This is suggested in its use in parsing for Korean (Han and Sarkar, 2002) and Turkish (Eryiğit and Oflazer, 2006), and MT for Czech (Al-Onaizan et al., 1999). But in this body of work, as well as the body of work mentioned in section 3.1, only analysis of one language is performed. Moreover there is no specific measure of richness of morphology; it is not obvious how to compare the morphology of different languages such as English, Arabic, Turkish or Korean with their different combinations of prefixing, suffixing and infixing. In this paper, to examine this idea, we look at two Australian Aboriginal languages sharing a similar morphological system, but with different levels of morphological richness. Australian Aboriginal languages are quite different from most others used in Natural Language Processing. Although indigenous Australian languages individually are quite distinct, some features are shared among many of them. In particular, many indigenous Australian languages are morphologically very rich. As for most languages around the world, heavier inflection usually goes together with a freer word order. The inflection of the different words conveys information which languages like English encode in word order, for example to distinguish subjects from objects. Most indigenous Australian languages are very heavily inflected, where it is not uncommon to have three or more postfixes on the same word. In some of these languages the boundaries between postfixes and words are quite imprecise. The form of a word reflects this, and morphology might be explicitly marked on words, where roots and postfixes are separated by special characters.

This morphologically rich nature of indigenous Australian languages becomes even clearer when set against European languages. In indigenous Australian languages suffixes attached to one word can carry a meaning which in Indo-European languages has to be expressed by separate individual words as opposed to suffixes. The boundary between these suffixes and individual words is starting to become vague as the suffixes do not just add some information to the root word, but can introduce complete new meaning elements.

Our work focuses on the languages Warlpiri (an indigenous language of central Australia) and Wik Mungkan (northern Cape York, Queensland, Australia). To the best of our knowledge, no machine translation on indigenous Australian languages has been attempted before, even though these languages share some quite interesting characteristics which are unique in the world. The major part of work in MT focuses on Indo-European and Asian languages. Applying MT to indigenous Australian languages therefore presents us with a new set of challenges.

The paper is structured as follows: in section 2 we provide the background on two Australian Aboriginal languages, and we describe the available data in these languages. Section 3 starts with some work related to our method, gives some background on the data characteristics of our domain, then describes our approach, experiment and the method for evaluation. Section 4 contains the results from this evaluation and has discusses interpret these results; this leads to a conclusion in section 5.

2 Languages and Data

It is difficult to research the effect of morphological analysis between languages with a different amount of morphological richness. It is very hard to compare different languages from completely different languages families, such as comparing English with Arabic or Czech. In trying to answer the question if morphological treatment is more beneficial for more morphological rich languages, we picked two highly inflected languages from the same language family. The Pama-Nyungan languages are the most widespread family of Australian Aboriginal languages and have in common a morphological system based entirely on suffixation (Austin, 2006). By using two languages from the same family, we can make more valid comparisons between them. Another reason for using Australian Aboriginal languages, is that some of them come with some 'free' morphological analysis: morphology is indicated to a certain degree in the writing system itself. As this is a human analysis, it is therefore more reliable than automatically acquired morphology.

We will first describe the two languages and show how they differ in morphological richness.

2.1 Warlpiri

Warlpiri is an interesting language to investigate because it is often considered the prototypical free word order language, and has a number of unusual characteristics. Morphosyntactic analyses have been proposed that describe these: extensive use of case-marking morphology, syntactic ergativity, PRO-drop (null pronominals), clitic-doubling, free word order (but with tight restrictions on the location of the auxiliary), discontinuous constituents, lack of a copula verb, a grammatical category of preverbs, and so on. In terms of linguistic analysis, there is extensive coverage of the grammar in Laughren and Hoogenraad (1996). Further, it is one of the major Aboriginal languages in Australia: it is spoken natively by roughly 3000 people, with at least another 1000 speaking it as a second language; it is one of the few where children are still learning to speak the language as their first language; and it has a deal of cul-

1 Nyampu yimi, ngulaju kamparru-warnu-juku nyiya-kanti-kantiki. Kamparruju nyurru-wiyiji, ngulaju God-ju nyinajalpa yangarlu-wiyi nyanungu-mipa, yalkiri manu walyaku lawa-juku. Ngula-jangkaju, ngurrju-manu yalkiri manu walya-wiyi.
2 Yalkiri kapu walya kuja ngurrju-manu, ngula-julpa lawa-juku walyaju ngunaja kirlka-juku. Ngulaju nyiya-kanti-kanti-wangu-juku. God-rlu kuja yalkiri manu walya ngurrju-manu, ngula-jangkaju, mangkurdurlulku wuuly-kujurnu, ngulalpa parra-wangu-juku karrija murnma-juku. God-rlu-julpa Pirlirrparlu warru warra-warra-kangu mangkurdu-wanarlu.

Table 1: Warlpiri sample extract, Genesis 1:1, 2

1 Ngay John=ang, ngay wik inanganiy umpang niiyant. Ngay wik inangan Jesus Christ=antam waa'-waa'ang niiyant aak ngeen nathan yaam ke'anaman wampow. Nil piip God=angan waa' nungant Jesus Christ=ant puth than pam wanch yotamang nunangan monkan-wakantan than mee'miy ngul yipam iiyayn. Nil puth Jesus=anganiy-a, ngaantiyongkan kuch nunang nil yipam meenathow ngathar ke' pithang yimanangan, ngay puth piip God nunang monkan-wak-wakang a' puth work nungant iiy-iiyang.
2 Ngay puth latang ump-umpang niiyant ngay pithangan thath-thathanga, wik God=antam anangana niiyant ngul waa' ang, wik anangan kan-kanam nil Jesus Christ=angan waa'-waa' nil God=angan meenath nungant.

Table 2: Wik Mungkan sample extract, Revelations 1:1, 2

tural support, for example through Warlpiri Media[1] and through bilingual teaching at the Northern Territory's Community Education Centres such as Yuendumu. Table 1 gives an impression of what Warlpiri looks like.

Because Warlpiri is a heavily agglutinative language, words can have many suffixes. The result can be very long words. To not confuse speakers, suffixes longer than one syllable are usually explicitly marked with a hyphen. This is an important feature we want to exploit later. Other inflections are not marked with hyphen:[2] *Nyangkajulu* which translates as *Look at me* is built from the blocks *(nyangka, look at) (+ju, me)* and *(+lu, you)*.

Suffixes can indicate many things, like tense, case, prepositions, location and more. Some examples are: *-wangu* which translates as *not, without*; *-pala* which indicates two speakers; *-kari* which means *another*; and *-nawu* which indicates it is that specific one. An extensive lexically based analysis of Warlpiri morphosyntax is given by Simpson (1991).

To have a first indication of which part of the written language consist of explicitly marked suffixes we counted how many hyphens the average word in Warlpiri has in our corpus (section 2.3). In table 3 we can see that over half the words carry at least one suffix, with many words carrying more.

[1]http://www.warlpiri.com.au

[2]We follow the notation convention which is common for Warlpiri to use a + for suffixes which 'glue' to the word without a hyphen and a - for suffixes where the hyphen remains when attached.

2.2 Wik Mungkan

To investigate the effects of morphological analysis we also look at another Australian Aboriginal language. We chose Wik Mungkan (Gordon, 2005), because of data availability and because it belongs to the same Pama-Nyungan language family as Warlpiri, and shares the highly agglutinative characteristics of Warlpiri. Wik Mungkan is a language which originates in northern Cape York, Queensland, Australia. The language nowadays is spoken by far fewer people (600 speakers, 400 native) and fewer resources are available for this language.

Table 2 gives an example of written Wik Mungkan. Wik Mungkan has less extensively marked morphology than Warlpiri, as can be concluded from table 4. Whereas Warlpiri has 0.615 postfixes on average per token, in Wik Mungkan we only have 0.257.

There are different writing conventions for Wik Mungkan as compared to Warlpiri. While in Warlpiri we only split on the hyphen token (−), in

Postfixes	count	percentage
0	36389	49.32%
1	30248	41.00%
2	6373	8.63%
3	704	0.95%
4	54	0.07%
5	3	0.00%

Average Postfixes per word: 0.615

Table 3: Warlpiri words carrying postfixes

Postfixes	count	percentage
0	211563	77.80%
1	51505	18.94%
2	8226	3.02%
3	647	0.24%
4	7	0.00%

Average Postfixes per word: 0.257

Table 4: Wik Mungkan words carrying postfixes

Wik Mungkan we split on the at-sign (@), the equal sign (=), the hyphen (−), the tilde (˜) and the apostrophe (′). A token like *Jesus=anganiy-a* is split into 3 individual tokens.

2.3 Bible Corpus

Bilingual data comprising English and an indigenous Australian language is extremely scarce. SMT models usually are data hungry, with performance increasing with availability of training data. Languages like Warlpiri have more texts available, but are either not translated, or do not have a close English translation. In our experiments we used parts of the Bible. Warlpiri and other indigenous Australian languages have Bible translations, which obviously are also available in English. We used a couple of books of the Bible which are translated into Warlpiri and the complete New Testament for Wik Mungkan.[3] We verse-aligned the texts in the Aboriginal language with an English Bible translation, the World English Bible (WEB) version. In English we had the opportunity to pick between several translations. We chose for the WEB translation because of the literalness of translations and, because the language is reasonably modern English, unlike the even more literal King James version.

Overall our corpus is very small for SMT models, and we are trying to obtain more data. For the moment we are interested in relative machine translation quality, and hope that translation quality will improve when provided with more bilingual data.

3 Method

3.1 Related approaches

To treat morphologically rich indigenous languages we want to do morphological analyses before translating. We do this as a preprocessing step in Phrase Based SMT (PSMT), leaving all the other PSMT steps untouched.

Preprocessing before applying PSMT has shown to be able to improve overall MT quality. As examples, Xia and McCord (2004), Collins et al. (2005) and Zwarts and Dras (2006) present an PSMT approach with word reordering as a preprocessing step, and demonstrate improved results in translation quality.

Work in Czech, done during the 1999 Summer Workshop at John Hopkins University (Al-Onaizan et al., 1999), describes an approach where Czech was turned into ‘Czech-prime’ as a preprocessing step. For Indo-European languages, Czech is highly inflected and has a relatively free word order. In their approach they first completely discarded inflective information like number, tense and gender. Later they used this information to artificially enhance their statistical model, by enriching the vocabulary of their statistical look-up table by adding new tokens based on seen roots of words with known morphology. Note that this work was not done in the PSMT paradigm, but using the original IBM statistical models (Brown et al., 1993) for MT.

An example of a fairly comprehensive analysis of the use of morphological analysis as a preprocessing step has been done on Arabic (Sadat and Habash, 2006). An Arabic morphological analyser was used to obtain an analysis of the build-up of

[3]These texts were made available to us by the Aboriginal Studies Electronic Data Archive (ASEDA).

Arabic words. Several models were presented which preprocessed the Arabic text. The key idea was to split off word parts based on specific analysis of the word. For example, pronominal clitics are split into several words. However, Arabic morphology is not as extensive as in languages like Warlpiri. Riesa et al. (2006) is another example where the use of morphological information boosts SMT quality. In this approach the tokens are separated from prefixes and postfixes based on a predefined list, derived from a grammar book. Lee (2004) similarly works on Arabic to English translation and separates prefixes and suffixes from the word stem. In contrast with our data, where we do not need to differentiate between different affixes. We only have postfixes, although stacked on each other and playing different roles, so we treat all morphology uniformly.

3.2 Data characteristics

We want to apply morphological preprocessing to Aboriginal languages to investigate its effect on morphologically rich languages as opposed to morphologically poorer ones. In Warlpiri it is possible to explicitly mark suffixes. We separate the suffixes from the main word and treat them as individual tokens. If we have the example sentence *Pina wangkaya yimi-kari* (*Say it again another way*) where we observe *yimi-kari* with *yimi* is *word, sentence* and *-kari* is *another*; we thus separate this to *Pina wangkaya yimi -kari*. Now the SMT models can pick up the individual meaning for *yimi* and *-kari* where this previously could not have been done. In situations where we find *-kari* without the original root word, we assume the SMT model can still translate it.

As a first step to see if our intuition is right we have done a word count for both the original tokens as for the tokens when split on hyphen, to get an idea of the frequency distribution. Some words which are not frequent when counted by string match become frequent if split on suffixes. Table 5 gives an overview of this distribution.

The most frequent word when split on suffixes appears less than ten times only by itself without splitting. Also, some suffixes suddenly appear very high in the frequency list when counting them as separate tokens, while it is impossible for them to feature in the top when we do not apply splitting. The third most common token after splitting, for example, is already a suffix, beating normal root words. In the top 100, we observe 46 suffixes.

rank	count	normal	count	normalised
1	2204	manu	3018	ngula
2	1330	ngulaju	2342	manu
3	934	yangka	2192	-jana
4	773	kuja	1748	God
5	538	ngula-gankaj	1656	-kari
6	529	Jesus	1619	kuja
7	453	wankaja	1557	-juku
8	438	nyina	1508	ngulaju
9	421	nyinaja	1479	-kurra
10	420	junga-juku	1314	yangka
17	226	God-rlu	859	-nyangu
18	256	God-kurlangu	807	-nyayirni
19	255	God-ku	804	-wangu

Table 5: Warlpiri word count and postfix normalised token count

Furthermore if we look at the positions 17, 18 and 19 in the top 100 we see the same root word. If we treat these tokens literally for the PSMT machinery they are three completely separate tokens, but surely they share some meaning. If we split them on hyphen, this partially reduces the data sparseness problem.

Phrase-based translation still allows to treat the split words with morphemes together and even map them to a single English token. Because both the root token and the separated suffixes are still in the same phrasal window, as far as the PSMT machinery is concerned it can still handle them together as if they are one token.

In that case on the Warlpiri side the phrase has several tokens. The difference is that it is now up to the phrasal model to decide how to treat them, individually or as a root suffix combination. Also the individual components have been observed more often in training, so the statistical accuracy for them individually should be higher. The model can choose to use the phrase or the individual components.

3.3 Experiment

For our baseline, we use the original corpus; we compare this against the corpus where the words are split on morphology. We verse-align them, because

1	B	I am most God jaru-kari so nyurrurla-kari-piya-wangu therefore concerned the marnkurrpaku-mipa working because christ jungangku out fruit
	S	I say this to be with you as in the three other wangkamirra because Christ speech give you for an prayed with you
	R	I thank my God I speak with other languages more than you all
2	B	but pina-yanta samaria ngajuku-palangu-kurlangu-kurra does
	S	but back -yanta from my father's of to right away
	R	but you shall go to my father's house
3	B	he ngula-warnurluju-jana Peter-rluju met all the Cornelius-kurlu and to all
	S	thus in all the Peter he told all the Cornelius life and of his life
	R	but Peter began and explained to them in order saying

Table 6: Warlpiri improvement: example translation set: (B)aseline, (S)plit, (R)eference

1	B	when he said to the house will and the assembly jews-antamakan hades
	S	so when he was lost and to enter into the synagogue
	R	he departed there and went into their synagogue
2	B	but he began again to the uuyamam he said to assuredly I tell this as I would like to know by inaniu I uuyaminga
	S	but when Peter to uuama he said truly I head a uuyaminga man was not know no again I is speaks
	R	Again he denied it with an oath I don't know the man

Table 7: Wik Mungkan improvement: example translation set: (B)aseline, (S)plit, (R)eference

both corpora come with verse information. Aligning them on a sentence level within verses was found to be extremely hard, especially since the same information was probably distributed over different sentence in a way problematic for the statistical machinery.

We use the normal tools to for PSMT: GIZA++ (Och and Ney, 2003) to statistically derive a sentence alignment on token level; and the decoder Pharaoh (Koehn, 2004), a beam search decoder for PSMT. Phrases are extracted by our own Phrase Builder, which extracts phrases based on the GIZA++ alignment as described in the Pharaoh manual. We used a trigram model with interpolated Kneser-Ney discounting as a language model. The language model was built using Biblical text and was enriched with extracts from the European Parliament in order to reduce data sparseness. The SRILM (Stolcke, 2002) toolkit was used to build this language model.

Our system still suffers from quite some considerable noise. This is not uncommon for a statistical approach, but particularly hits the system hard in data-poor environments. In an abundant data scenario, noise tends to get averaged out. Some of the noise we experienced in our domain was due to poor verse alignment. There is strong indication in the test set that the PSMT system is actually translating a different sentence than the reference. Since the test and training data are obtained via the same means we assume this is also the case in the training set.

3.4 Evaluation

Often the BLEU metric is used in MT next to human evaluation, to assess translation quality. We did not perform BLEU evaluation, since our overall translation quality is quite poor. This means that in translation often synonyms are selected which BLEU does not pick up. In data sparse environments this might increase the randomness in BLEU results unfavourably and the test set already is small. More importantly however, it might favour one of the systems unfairly above the others. The PSMT system used leaves unknown words untranslated.

Proper nouns are quite likely to have morphology attached in the baseline system. When untranslated this means the proper noun is not matched against a possible English proper noun. Our system splits on morphology, leaving the proper noun by itself, which is identical to English and can be matched without translation. So although nothing is translated our system would score higher using the BLEU metric. Therefore we decided to do a human evaluation only.

To evaluate our model properly we asked human annotators to evaluate the new model against the baseline model. We used three human annotators to evaluate the Warlpiri set, and two to evaluate the Wik Mungkan set. In a blind evaluation we provided them with two alternatives for a translation and a reference verse. For each verse the ordering of baseline and 'split' version was random. So as not to overload our human annotators we drew 50 sentences from our test set, which was based on translation of unseen verses during the training period for the PSMT system. We asked the annotator to indicate which option was a better translation. They were also allowed to leave sentence-options undecided if they could not distinguish quality or if translation quality was too poor to make a good indication. They were provided with a reference translation in English.

4 Results and Discussion

The results of the human evaluation are presented in table 8. For both languages the sum over different annotates is presented for each time they chose that system. To test for statistical significance we used the non-parametric Sign Test. For the Wik Mungkan language the improvement is not statistically significant at the 5% level. With a probability of 7.5% it is possible that our system was chosen more often by random chance, and not because of improved translation. The largest frequently used threshold for statistical significance is 5%, although occasionally 10% is used, so this gives at best weak support to the rejection of the null hypothesis. For Warlpiri, however, there is overwhelming support to indicate we indeed achieved translation quality. This provides some initial support for the intuition that more highly inflected Aboriginal languages will benefit more from morphological analysis.

In absolute terms the quality of translation is quite poor, because we operate in an extremely data-poor scenario. We give some examples of translations for which the authors thought there was a clear improvement of translation quality. This also gives an indication of overall translation quality and shows the clear need for more training data for PSMT. For Warlpiri the examples can be found in Table 6, and for Wik Mungkan in Table 7.

If we take the first translation we see that the baseline has four times as many untranslated words as our system based on splitting. Furthermore we can recognise some more words, like *language* and *speech* which presumable link to each other. Now many more steps need to be performed to build a decent translation out of it, but at least there is a strong indication for a relative improvement.

Many suffixes are not captured yet. At the moment we only treated the explicitly marked suffixes because here we can be sure they are suffixes. Warlpiri knows many suffixes which are not separated with a hyphen, usually one syllable suffixes. To recognise these suffixes we need a morphological analyser. Since we have shown that splitting words contributes positively towards translation quality this seems like a logical step to extend this project in the future. Further experiments need to be carried out to see if these not explicitly marked suffixes can also improve overall quality when they are separated from their root word.

We assume our model performs better for different reasons. First of all, because we have a PSMT system, we can still pick the word with morphology if the system prefers it (word and morphology still fits the phrase window), removing most of the drawbacks a morphological prepocessing step would have without the ability to group things together in phrases. Also, the system can actually use words in cases where the individual words with that morphology attached have never been encountered before, in cases where we have seen it with different morphology. Secondly, because more words are translated, the language model starts to kick in. When words remain untranslated the language model cannot differentiate; when more words are translated we get a positive feedback. Most of all, many suffixes do not only carry morphosyntactic in-

	Wik Mungkan			Warlpiri			
	Ann. I	Ann. II	**Total**	Ann. I	Ann. II	Ann. III	**Total**
System							
Split	24	30	54	26	35	45	106
Baseline	23	18	41	7	4	3	14
Undecided	3	2	5	17	11	2	30
Sign Test							
Likelihood			$7.5 \cdot 10^{-2}$				$7.5 \cdot 10^{-20}$

Table 8: Assessment of human annotators

formation, but are actual meaning elements. Unlike English where inflection represents only a small amount of information such as tense or number, in Aboriginal languages the morphology is so extensive that to translate this morphology itself, we might need (multiple) separate words in English. By separating them, our model gives the PSMT machinery the option to exploit this.

5 Conclusion

Previous work indicates that preprocessing of Natural Language helps achieving overall quality in different Natural Language applications. Our focus is the Phrasal Statistical Machine Translation paradigm in the highly inflected indigenous Australian languages. We show a clear relative improvement of overall Machine Translation quality by separating explicitly marked suffixes when we preprocess languages like Warlpiri, which is the language with the heavier explicitly marked morphology of the two. In Wik Mungkan we observe only a possible but not statistically significant improvement.

A clear improvement of translation quality is achieved by targeting explicitly marked morphology only. However there is more morphological analysis possible in these languages. In future work we would like to included other morphology by using morphological analysers and measure their impact on machine translation quality: looking at a wider range of languages will let us test more extensively the relationship between morphological richness and the usefulness of morphological preprocessing.

6 Acknowledgement

We would like to acknowledge the translators of the Bible parts, Steve Schwarz for Walpiri and Christine Kilham for Wik Mungkan and the institute which granted us the digital resources and the right to use this material: Aboriginal Studies Electronic Data Archive (ASEDA) and the Australian Institute of Aboriginal and Torres Strait Islander Studies (AIATSIS).

We would also like to thank the individual annotators who were willing to annotate translation quality over the evaluation texts we provided them.

References

Yaser Al-Onaizan, Jan Cuřín, Michael Jahr, Kevin Knight, John Lafferty, Dan Melam ed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. *Statistical Machine Translation, Final Report, JHU Workshop*. URL `http://www.clsp.jhu.edu/ws99/projects/mt/final_report/mt-final-report.ps`.

Peter Austin. 2006. Countries and Language – Australia. In *International Encyclopedia of Language and Linguistics 2nd edition*, Article 1711. Oxford: Elsevier.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics 19(2)*, pages 263–311.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 531–540. Ann Arbor, Michigan. URL `http://www.aclweb.org/anthology/P/P05/P05-1066`.

Gülsen Eryiğit and Kemal Oflazer. 2006. Statistical

Dependency Parsing for Turkish. In *Proceedings of EACL 2006 - The 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 89–96.

Raymond G. Gordon. 2005. *Ethnologue : Languages of the World*, Fifteenth edition. URL http://www.ethnologue.com/show_language.asp?code=wim.

Chung-hye Han and Anoop Sarkar. 2002. Statistical Morphological Tagging and Parsing of Korean with an LTAG Grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Formalisms*.

Philip Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models, Philipp Koehn, Association for Machine Translation in the Americas.

Mary Laughren and Robert Hoogenraad. 1996. *A Learner's Guide to Warlpiri. Tape Course for Beginners. Wangkamirlipa Warlpirilki*. IAD Press, Alice Springs, Australia.

Young-Suk Lee. 2004. Morphological Analysis for Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 57–60.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Jason Riesa, Behrang Mohit, Kevin Knight, and Daniel Marcu. 2006. Building an English-Iraqi Arabic Machine Translation System for Spoken Utterances with Limited Resources. In *Proceedings of the International Conference on Spoken Language Processing (Interspeech'2006)*, pages 17–21.

Fatiha Sadat and Nizar Habash. 2006. Combination of Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 1–8. Association for Computational Linguistics.

Jane Simpson. 1991. *Warlpiri morphosyntax: a lexicalist approach*. Kluwer, Dordrecht.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904.

Fei Xia and Michael McCord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of the 20th International Conference on Computational Linguistics (Coling)*, pages 508–514.

Simon Zwarts and Mark Dras. 2006. This Phrase-Based SMT System is Out of Order: Generalised Word Reordering in Machine Translation. In *Proceedings of the Australasian Language Technology Workshop*, pages 149–156.

Exploring Extensions to Machine Learning-based Gene Normalisation

Benjamin Goudey,[†] **Nicola Stokes**[†‡] and **David Martinez**[†]
† Computer Science and Software Engineering
and
‡ NICTA VRL
University of Melbourne
VIC 3010, Australia
{bgoudey,nstokes,davidm}@csse.unimelb.edu.au

1 Introduction

One of the foundational text-mining tasks in the biomedical domain is the identification of genes and protein names in journal papers. However, the ambiguous nature of gene names means that the performance of information management tasks such as query-based retrieval will suffer if gene name mentions are not explicitly mapped back to a unique identifier in order to resolve issues relating to synonymy (i.e. many different lexical forms representing the same gene) and ambiguity (i.e. many distinct genes sharing the same lexical form). This task is called gene name normalisation, and was recently investigated at the BioCreative Challenge (Hirschman et al., 2004b), a text-mining evaluation forum focusing on core biomedical text processing tasks. In this work, we present a machine learning approach to gene normalisation based on work by Crim et al. (2005). We compare this system with a number of simple dictionary lookup-based methods. We also investigate a number of novel features not used by Crim et al. (2005). Our results show that it is difficult to improve upon the original set of features used by Crim et al. We also show that for some organisims gene name normalisation can be successfully performed using simple dictionary lookup techniques.

2 Data

The experiments described in this paper were performed on the data provided by the first BioCreative workshop for gene normalisation. For each abstract in the test collection the system must create a list of normalised gene names mentioned in the text. Three distinct organism datasets were investigated: *yeast*, *mouse*, *fly*. Systems are provided with a gene synonym list for each organism containing a comprehensive list of gene identifiers and many of their related gene mentions, together with a set of training instances (abstracts with corresponding gene lists) for each organism. The gene lists used as training data were created by filtering down pre-existing manually compiled lists that applied to whole documents. This automatic filtering process added noise to the training data by lowering the recall of gene lists to 86%, 80% and 55% for the yeast, fly and mouse data respectively. More information on the data for this task can be found in (Hirschman et al., 2004a).

3 System Description

As already mentioned, we use a machine learning approach similar to that used by Crim et al. (2005), one of the top performing systems at the BioCreative I Challenge. There are 3 main stages in our system: first, the document is run through a high recall gene identification system; each candidate mention is then used to create a series of instances for each possible gene identifier related to the mention, extracting a variety of contextual features based on the surrounding text; and finally, instances are passed to a maximum entropy classifier. We use the training data to build our model, where each instance in the testing part is classified and a confidence value is returned. Finally, the gene identifiers with the highest confidence value for a particular gene mention are added to the gene list for that abstract.

4 Results and Conclusions

One of the limitations of the system by Crim et al. (2005) is that the use of exact matching and the in-

	Yeast			Mouse			Fly		
	Prec.	Rec.	F-sc.	Prec.	Rec.	F-sc.	Prec.	Rec.	F-sc.
BioTagger - Basic	94.0	59.2	72.6	73.8	70.4	71.8	49.5	39.3	40.5
LU - Basic	89.0	90.9	89.9	1.9	89.7	3.7	2.0	95.3	3.8
LU - Entrez/Filtered	94.0	88.7	91.3	73.7	74.8	**74.3**	45.8	91.6	61.0
LU - Variations and Entrez/Filtered	93.5	89.6	**91.5**	60.4	77.9	68.1	41.8	92.1	57.5
ML - Basic	95.4	77.3	85.4	84.4	56.8	67.9	75.1	72.5	73.8
ML - Entrez/Filtered	95.2	87.4	91.2	82.2	66.2	73.3	74.0	81.6	**77.6**
ML - Variations and Entrez/Filtered	94.7	88.3	91.4	78.7	68.6	73.3	71.8	82.5	76.8

Table 1: BioTagger results and synonym list expansion over our machine learning (ML) and lookup-based (LU) systems (best f-scores shown in bold)

	Yeast			Mouse			Fly		
	Prec.	Rec.	F-sc.	Prec.	Rec.	F-sc.	Prec.	Rec.	F-sc
Crim et al. (2005)	95.6	88.1	91.7	78.7	73.2	75.8	70.4	78.3	74.2
ML - Crim Features	94.9	88.9	91.8	82.2	66.2	73.3	74.0	81.6	77.6
ML - All Features	95.1	88.1	91.4	79.4	71.0	75.0	69.5	82.8	75.5
ML - Optimal Features	94.4	90.0	**92.2**	78.8	73.7	**76.2**	75.6	81.5	**78.5**

Table 2: Comparison of (Crim et al., 2005) and our ML system with different feature sets (best f-scores shown in bold)

completeness of the synonym list limits the ability of system to achieve high recall. To address this issue, we experiment with a variety of synonym list expansion and filtering methods including:

- **Lexical Variations** - the creation of gene name variations with different hyphenation and spacing patterns.
- **Entrez Gene** - the expansion of the original synonym list with information from Entrez Gene (Maglott et al., 2005).
- **Conditional Probability** - a conditional probability filter which was used by (Crim et al., 2005) in their pattern matching system.

We tested two different approaches to this task: the first performs no explicit disambiguation, but adds all possible gene identifiers to the gene list for each gene mention; the second by using the maximum entropy classifier as outlined in Section 3. We also compared our results to those of the BioTagger (McDonald and Pereira, 2005), a well-known "out of the box" gene identification system.

For our two main systems (lookup and machine learning), we ran different combinations of the synonym list expansion, with the top two performing results and the baseline shown in Table 1. We can see that the recall of the BioTagger is very low, which suggests that we would need some tuning to apply it to this specific dataset. These results illustrate that the yeast data needs almost no explicit disambiguation, i.e. the the lookup-based system performs best. While the fly data, which contains very ambiguous gene mentions, needs a machine-learning approach to identify the correct identifier. Surprisingly, the mouse, which also has a reasonable degree of ambiguity, performs at its best with a lookup based system. It must be noted, that the noise of the training data may have contributed to the poor performance of the classifier, yet this is still an interesting result.

In the next experiment, our aim was to improve classifier performance by increasing the original feature set (from 5 to 21) with different features derived from linguistic information (POS tags) and information from external resources (e.g. whether the target word is defined in WordNet). Using these features we compare our gene normalisation system to that of (Crim et al., 2005) using all new features as well as an optimal subset of these. The results are shown in Table 2. While the results of using the entire extended feature set tend to degrade performance compared to the basic set (except for mouse), using a subset of features unique to each organism does lead to some performance improvements. This implies that a one-classifier fits all approach is not suitable for gene normalisation, and that individual classifiers must be created for each organism.

References

J. Crim, R. McDonald, and F. Pereira. 2005. Automatically annotating documents with normalised gene lists. *BMC Bioinformatics*, 6 (Supplement I)(13).

L. Hirschman, M. Colosimo, A. Morgan, and A. Yeh. 2004a. Overview of biocreative task 1b: Normalized gene lists. *Journal of Bioimedical Informatics*, 37.

L. Hirschman, A. Yeh, C. Blaschke, and A. Valencia. 2004b. Overview of biocreative: Critical assessment of information extraction for biology. *Journal of Bioimedical Informatics*, 37.

Donna Maglott, Jim Ostell, Kim D. Pruitt, and Tatiana Tatusova. 2005. Entrez gene: Gene-centered information at ncbi. *Nucleic Acids Resarch*, 33 (Database Issue).

R. McDonald and F. Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *Journal of Bioimedical Informatics*, 37.

Proceedings of the Australasian Language Technology Workshop 2007, pages 146-148

Distributional Similarity of Multi-Word Expressions

Laura Ingram and **James R. Curran**
School of Information Technologies
The University of Sydney
NSW 2006, Australia
{ling6188, james}@it.usyd.edu.au

Abstract

Most existing systems for automatically extracting lexical-semantic resources neglect multi-word expressions (MWEs), even though approximately 30% of gold-standard thesauri entries are MWEs.

We present a distributional similarity system that identifies synonyms for MWEs. We extend Grefenstette's SEXTANT shallow parser to first identify bigram MWEs using collocation statistics from the Google WEB1T corpus. We extract contexts from WEB1T to increase coverage on the sparser bigrams.

1 Introduction

Lexical-semantic resources, such as WordNet (Fellbaum, 1998), are used in many applications in Natural Language Processing (NLP). Unfortunately, they are expensive and time-consuming to produce and are prone to bias and limited coverage. Automatically extracting these resources is crucial to overcoming the knowledge bottleneck in NLP.

Existing distributional approaches to semantic similarity focus on unigrams, with very little work on extracting synonyms for multi-word expressions (MWEs). In this work, we extend an existing system to support MWEs by identifying bigram MWEs using collocation statistics (Manning and Schütze, 1999). These are calculated using n-gram counts from the Google WEB1T corpus (Brants and Franz, 2006).

We evaluate against several gold-standard thesauri and observe a slight decrease in overall performance when the bigram MWEs were included. This is unsurprising since the larger vocabulary and sparser contextual information for bigrams makes the task significantly harder. We also experimented with contexts extracted from WEB1T in an attempt to overcome the data sparseness problem. Inspection of the results for individual headwords revealed many cases where the synonyms returned were significantly better when bigram data was included.

2 Background

Distributional similarity relies on the *distributional hypothesis* that similar terms appear in similar contexts (Harris, 1954). Here we extend the SEXTANT parser (Grefenstette, 1994) to include multi-word *terms* and syntactic *contexts*.

Curran (2004) experiments with different parsers for extracting contextual information, including SEXTANT, MINIPAR (Lin, 1994), RASP (Briscoe and Carroll, 2002), and CASS (Abney, 1996). Lin (1998) used MINIPAR and Weeds (2003) used RASP for distributional similarity calculations. MINIPAR is the only parser to identify a range of MWEs that has been used for distributional similarity. Weeds (2003) and Curran (2004) evaluate measures for calculating distributional similarity. We follow (Curran, 2004) in using the weighted Jaccard measure with truncated t-test relation weighting for our experiments.

3 Detecting MWEs

The initial step in creating a thesaurus for MWEs is to identify potential MWE headwords using collocation statistics. We used various statistical tests, e.g. the t-test and the log-likelihood test (Manning and Schütze, 1999), calculated over the Google WEB1T unigram and bigram counts. These counts, calculated over 1 trillion words of web text, gave the most reliable counts. However, highly ranked terms, e.g.

Contact Us and Site Map, demonstrate bias towards web-related terminology. This list of selected bigrams is used to detect bigrams within the BNC using a modified version of the Viterbi algorithm.

4 Context Extraction

Grefenstette's (2004) (SEXTANT) parser was extended to extract contextual information for the list of selected bigrams extracted above. Adding these bigrams does not result in a substantial increase in the number of relations which implies that there is very little contextual information available about the bigram data. This has a significant impact on the difficulty of the task.

Experiments were also conducted whereby the contextual information was extracted from the WEB1T 3, 4 and 5-gram data for a list of known bigrams from the gold-standard thesauri. This data lacks the syntactic information provided by SEXTANT but the counts are estimated over 10,000 times as much data. This should reduce the sparseness problem for the bigram headwords.

5 Synonym Extraction

Following Curran (2004), the extracted synonyms are compared directly against multiple gold-standard thesauri. We extend this evaluation to include multi-word headwords and synonyms. We randomly selected 300 unigram and 300 bigram headwords from the MAQCUARIE (Bernard, 1990), MOBY (Ward, 1996), and ROGET'S (1911) thesauri, and WORDNET (Fellbaum, 1998).

We calculated the number of direct matches against the gold standard (DIRECT) and the inverse rank (INVR), the sum of the reciprocal ranks of matches. The results for the unigram headword experiments are summarised in Table 1.

Both INVR and DIRECT demonstrate that performance decreases when MWEs are included. However, performance did increase significantly for some terms when MWEs were added. For example, tool improved from 0.270 to 0.568 INVR. The results for rate, shown in Table 2, also improved.

The next set of experiments extracted synonyms for 300 bigram headwords drawn from the MACQUARIE thesaurus. The best results for bigram headwords was achieved when unigram and bigram data was extracted from WEB1T and the VPC resource (Baldwin and Villavicencio, 2002) was included. Table 3 shows the top 5 synonyms (as ranked by the Jaccard measure) for atomic bomb and dining table.

		DIRECT	INVR
BNC t-test	UNI	22.6	1.717
	UNI+ BI	22.2	1.650
	UNI+ BI+ VPC	22.2	1.659
WEB1T t-test	3UNI	16.9	1.182
	4UNI	19.3	1.454
	3UNI+ 4BI	15.6	1.004
	4UNI+ 5BI	19.8	1.344
	3UNI+ 4BI+ 4VPC	15.6	1.001
	4UNI+ 5BI+ 5VPC	19.8	1.346
WEB1T THES	3UNI+ 4BI	17.5	1.185
	4UNI+ 5BI	17.5	1.194
	3UNI+ 4BI+ 4VPC	17.5	1.187
	4UNI+ 5BI+ 5VPC	21.2	1.491

Table 1: Results for unigram headwords

UNI	UNI+ BI	UNI+ BI+ VPC
level	level	level
price	price	price
cost	amount	cost
income	cost	amount
growth	speed	average

Table 2: Sample synonyms for rate

ATOMIC BOMB	DINING TABLE
nuclear bomb	coffee table
atom bomb	dining room
nuclear explosion	cocktail table
atomic explosion	dining chair
nuclear weapon	bedroom furniture

Table 3: Sample bigram synonyms

6 Conclusion

We have integrated the identification of simple multi-word expressions (MWEs) with a state-of-the-art distributional similarity system. We evaluated extracted synonyms for both unigram and bigram headwords against a gold standard consisting of the union of multiple thesauri.

The main difficulties are the sparsity of distributional evidence for MWEs and their low coverage in the gold standard. These preliminary experiments show the potential of distributional similarity for extracting lexical-semantic resources for both unigrams and MWEs.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. *Journal of Natural Language Engineering*, 2(4):337–344.

Timothy Baldwin and Aline Villavicencio. 2002. Extracting the unextractable: A case study on verb-particles. In *Proceedings of the Sixth Conference on Computational Natural Language Learning (CoNLL 2002)*, pages 98–104, Taipei, Taiwan.

John R.L. Bernard, editor. 1990. *The Macquarie Encyclopedic Thesaurus*. The Macquarie Library, Sydney, Australia.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1. Technical report, Google Inc.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1499–1504, Las Palmas de Gran Canaria, 29-31 May.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

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

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, USA.

Zellig Harris. 1954. Distributional structure. *Word*, 10(2/3):146–162.

Dekang Lin. 1994. Principar – an efficient, broad-coverage, principle-based parser. In *Proceedings of COLING-94*, pages 482–488, Kyoto, Japan.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 2, pages 768–774, Montreal, Quebec, Canada.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Language Processing*. MIT Press, Cambridge, MA USA.

Peter Mark Roget. 1911. *Thesaurus of English Words and Phrases*. Longmans, Green and Company, London, UK.

Grady Ward. 1996. Moby thesaurus. http://etext.icewire.com/moby/.

Julie E. Weeds. 2003. *Measures and Applications of Lexical Distributional Similarity*. Ph.D. thesis, University of Sussex.

Proceedings of the Australasian Language Technology Workshop 2007, pages 149-151

Extending CCGbank with quotes and multi-modal CCG

Daniel Tse and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dtse6695, james}@it.usyd.edu.au

Abstract

CCGbank is an automatic conversion of the Penn Treebank to Combinatory Categorial Grammar (CCG). We present two extensions to CCGbank which involve manipulating its derivation and category structure. We discuss approaches for the automatic re-insertion of removed quote symbols and evaluate their impact on the performance of the C&C CCG parser. We also analyse CCGbank to extract a multi-modal CCG lexicon, which will allow the removal of hard-coded language-specific constraints from the C&C parser, granting benefits to parsing speed and accuracy.

1 Introduction

Combinatory Categorial Grammar (Steedman, 2000) is a powerful lexicalised grammar formalism. In CCG, the combination of *categories* using a small set of *combinatory rules* allows a parser to simultaneously build up syntax and semantics.

CCGbank, a corpus automatically converted from the Wall Street Journal section of the Penn Treebank (Marcus et al., 1994), forms the cornerstone of research in wide-coverage CCG parsing. Its coverage enables the construction of practical, efficient and robust CCG parsers (Clark and Curran, 2007).

We describe corpus transformations on CCGbank which improve its linguistic fidelity and discriminative power. We restore the quote symbols to CCGbank derivations removed by the CCGbank derivation procedure (Hockenmaier, 2003). Quotes yield useful local information for many corpus applications, such as speaker or topic segmentation, and the *supertagging* phase in a CCG parser (the assignment of categories to lexical items).

Multi-modal Combinatory Categorial Grammar (Baldridge, 2002) is an extension to CCG for finer-grained control over the applicability of combinatory rules. We develop a method for the semi-automatic extraction of a MMCCG corpus from CCGbank. These corpus transformations increase the fidelity and precision of CCGbank and hence its usefulness in a range of applications.

2 Multi-modal CCG

In CCG, lexical items are mapped to categories, which are combined through a set of combinatory rules. Typically, permitting all of the combinators to be considered by the parser leads to the undesired acceptance of ungrammatical examples, known as *overgeneration*. To prevent this in pure CCG, restrictions on the set of applicable rules must be specified as hard-coded parser constraints.

$$X/_{\star}Y \quad Y \rightarrow X \qquad\qquad Y/_{\circ}Z \quad X\backslash_{\circ}Y \rightarrow X/_{\circ}Z$$
$$Y \quad X\backslash_{\star}Y \rightarrow X$$

$$X/_{\circ}Y \quad Y/_{\circ}Z \rightarrow X/_{\circ}Z \qquad\qquad X \rightarrow T/_{i}(T\backslash_{i}X)$$
$$Y\backslash_{\circ}Z \quad X\backslash_{\circ}Y \rightarrow X\backslash_{\circ}Z \qquad\qquad X \rightarrow T\backslash_{i}(T/_{i}X)$$

To restore lexicality, Baldridge (2002) devised *multi-modal* CCG, in which each slash of a category encodes an indication (or *mode*) defining the rules in which it may participate. Modes simultaneously yield efficiency benefits and a reduction in derivational ambiguity by limiting the set of combinatory rules a MMCCG parser has to consider.

3 Re-quoting CCGbank

We restore quotes to CCGbank by consulting the Penn Treebank to determine the original location of the opening and/or closing quote. Having found the corresponding leaf in the CCGbank derivation, we ascend to its parent as long as the leaves of its subtree contain strictly a sequence of tokens which were originally between quotes in the text. We splice in the quote below the node at which this condition is no longer true, so that the re-inserted quote leaf dominates the text that it quotes.

We correctly reinstate quotes in 8477 of the 8677 derivations (97.7%) originally containing quotes. We evaluate the re-quoted CCGbank on the C&C CCG parser (Clark and Curran, 2007).

Corpus	Labelled F	Supertagger acc
C&C orig	85.12%	93.05%
Re-quoted	85.03%	93.18%

Figure 1: C&C evaluation

As per our hypothesis, the extra local information provided by the presence of quotes slightly increases supertagging accuracy. However, there is a small tradeoff against parser accuracy, due to the additional complexity the re-added quotes entail.

4 Multi-modal CCG

A simple approach to mode annotation is to examine each slash of each category occurring in CCGbank. If a given slash is *consumed* by a given rule with a proportion α of total cases, then we assign that slash a mode ($\star, \bowtie, \circ$) compatible with that rule. The *null mode* ($\bowtie$) permits no rules, allowing us to mark a category such as $S[pt]\backslash NP$, whose slash is never directly consumed.

The problem with this automated approach is sparseness: rarely attested rules can nevertheless contribute to legitimate analyses. We address this by performing manual annotation on those slashes consumed often by the more powerful composition rules, while relying on our frequency cutoff criterion to assign the application-only and null modes.

There are two outcomes in annotating a given CCG category with modes: for each slash, we either assign the least permissive mode that preserves the vast majority of CCGbank derivations, or else we discover that a given CCG category corresponds to two or more MMCCG categories differing by mode.

We give an example of the additional fine-grained control provided by MMCCG, allowing us to make a grammaticality distinction previously impossible to specify lexically in CCG. The adverbs *freely* and *evidently* belong to the classes *VP adverb* and *sentential adverb*, respectively. The former are characterised by their ability to undergo a degree of shifting unavailable to the latter.

(1) Adverbs permute within their phrase *freely*.
(2) Adverbs permute *freely* within their phrase.
(3) He knows some judo *evidently*.
(4) *He knows *evidently* some judo.

This distinction cannot be made in CCG, because both *freely* and *evidently* share the structural category $(S\backslash NP)\backslash(S\backslash NP)$. However, the additional derivational control of MMCCG allows us to partition the VP and sentential adverbs. In particular, the VP adverbs would carry a category $(S\backslash NP)\backslash_{\circ}(S\backslash NP)$, the mode $\circ$ permitting the use of the combinatory rules of composition which enable a CCG analysis of movement, while the sentential adverbs would receive the category $(S\backslash NP)\backslash_{\star}(S\backslash NP)$, the mode $\star$ only permitting the non-associative and non-permutative rules of application. A further benefit of moving these distinctions into the lexicon is that we can make these grammaticality distinctions with the granularity of lexical items.

5 Conclusion

We have described two transformations on CCGbank, which enhance and extend its utility as a CCG corpus. We have produced a CCGbank of greater fidelity through an algorithm for re-instating quote symbols removed during its corpus conversion process, demonstrated the role of multi-modal CCG in addressing overgeneration inherent in pure CCG, and described a strategy for the generation of a MMCCG corpus. We have considered automatic and manual strategies for the annotation of a MMCCG corpus, and justify our chosen solution of a compromise between them. The focus of our work is now to refine the corpus obtained, and implement a full MMCCG parser.

Through the generation of a multi-modal version of CCGbank, we have the potential for more accurate, and at the same time more efficient wide-coverage CCG parsing.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Statistical Parsing with CCG and Log-Linear Models. *To appear in* Computational Linguistics 2007. *Unpublished draft manuscript*.

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

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Author Index