Australasian Language Technology Workshop 2005

Proceedings of the Workshop

Workshop Chairs:
Timothy Baldwin
James Curran
Menno van Zaanen

10-11 December 2005
University of Sydney
Sydney, Australia

Proceedings of the Australasian Language Technology Workshop 2005

URL: http://www.alta.asn.au/events/altw2005/

Sponsors:

Australian Government
Department of Defence
Defence Science and
Technology Organisation

The University of Sydney

ISBN: 0-9751687-2-X

To order copies of this and other ALTA proceedings, contact workshop@alta.asn.au

Introduction

This volume contains the papers accepted for presentation at the Australasian Language Technology Workshop (ALTW) 2005, held at the University of Sydney, Sydney, Australia, on 10-11th of December, 2005. This is the third 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.

One innovation in this year's Australasian Language Technology Workshop was the introduction of poster presentations in addition to the regular talks. Our intention here was to optimise the presentation medium for each submission, and reviewers were accordingly instructed to independently rate the acceptability of each submitted paper first as a regular paper and second as a poster. Specifically, consideration was given to: (a) which mode of delivery was most appropriate for a given submission (e.g. papers which were felt to benefit from a more interactive presentation were preferred as posters), (b) what was the technical merit of the submission (e.g. highly technical papers which relied on detailed explanation of a series of equations were preferred as regular papers), and (c) was the submission of general interest (e.g. papers describing general results with ramifications for a range of fields were preferred as regular papers). It is important to note that research quality and technical rigour were not taken into consideration in determining whether to accept a paper as a regular paper or poster. As such, regular papers and posters are of identical academic status.

Of the 45 papers submitted, 30 papers were selected by the programme committee for publication and appear in these proceedings. Of these, 14 are regular papers and 16 are posters. 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 F1 conference publications.

We would like to thank all the authors who submitted papers, as well as the members of the program committee for the time and effort they contributed in reviewing the papers, and Dan Flickinger, Kathy McKeown and Virach Sornlertlamvanich for providing the ideal complement to the workshop with their invited talks. Our thanks go also to members of the ALTA executive, and particularly Steven Bird and Cécile Paris for encouragement and support in organising the workshop. Finally, we would like to thank the sponsors (DSTO, NICTA, CSIRO and Appen) for their generous help in supporting the workshop, and Dominique Estival as sponsorship chair.

Timothy Baldwin, James Curran, Menno van Zaanen

Organizers:

Timothy Baldwin (University of Melbourne)
James Curran (University of Sydney)
Menno van Zannen (Macquarie University)

Program Committee:

Ash Asudeh (University of Canterbury)
Eric Atwell (Leeds University)
Timothy Baldwin (University of Melbourne)
Steven Bird (University of Melbourne)
Lawrence Cavedon (NICTA Victoria)
Trevor Cohen (University of Melbourne)
James Curran (University of Sydney)
Walter Daelemans (University of Antwerp)
Robert Dale (Macquarie University)
Dominique Estival (Defence Science and Technology Organisation)
Dan Flickinger (Oslo, Saarland and Stanford Universities)
Tanja Gaustad (Appen)
Graeme Hirst (University of Toronto)
Ben Hutchinson (University of Edinburgh)
Jong-bok Kim (Kyung Hee University)
Alistair Knott (University of Otago)
Valia Kordoni (University of Saarland)
Mirella Lapata (University of Edinburgh)
Hang Li (Microsoft Research)
Diana McCarthy (University of Sussex)
Daniel Midgley (University of Western Australia)
Diego Molla (Macquarie University)
Kyonghee Paik (KLI Language and Translation)
Ajeet Parhar (Telstra Research Laboratories)
Cécile Paris (CSIRO ICT Centre)
Jon Patrick (University of Sydney)
David Powers (Flinders University)
Tony Smith (Waikato University)
Harold Somers (University of Manchester)
Nicola Stokes (NICTA Victoria)
Takaaki Tanaka (NTT Communication Science Laboratories)
Aline Villavicencio (University of Essex)
Menno van Zaanen (Macquarie University)
Simon Zwarts (Macquarie University)

Invited Speakers:

Dan Flickinger (Oslo, Saarland and Stanford Universities)
Kathy McKeown (Columbia University)
Virach Sornlertlamvanich (NICT)

Table of Contents

Workshop Programme

DAY 1 — 10 DECEMBER, 2005

09:25-09:30 Opening Remarks

09:30-10:00 *Disambiguating Conjunctions in Named Entities*
Pawel Mazur and Robert Dale

10:00-10:30 *Learning of Graph Rules for Question Answering*
Diego Molla and Menno van Zaanen

10:30-11:00 *A Statistical Approach towards Unknown Word Type Prediction for Deep Grammars*
Yi Zhang and Valia Kordoni

11:00-11:30 Coffee Break

11:30-12:00 *Tagging Unknown Words with Raw Text Features*
David Vadas and James R. Curran

12:00-12:30 *POS Tagging with a More Informative Tagset*
Andrew MacKinlay and Timothy Baldwin

12:30-13:00 *Augmenting Approximate Similarity Searching with Lexical Information*
James Gorman and James R. Curran

13:00-14:00 Lunch

14:00-15:00 *Dimensions of Deep Grammar Validation*
Invited Speaker – Dan Flickinger

15:00-15:15 Coffee Break

15:15-16:30 POSTER SESSION 1

Word Prediction in a Running Text: A Statistical Language Modeling for the Persian Language
Masood Ghayoomi and Seyyed Mostafa Assi

Using Diverse Information Sources to Retrieve Samples of Low Density Languages
Andrew MacKinlay

Faking it: Synthetic Text-to-speech Synthesis for Under-resourced Languages – Experimental Design
Harold Somers

Dual-Type Automatic Speech Recogniser Designs for Spoken Dialogue Systems
Jason Littlefield and Michael Broughton

Efficient Knowledge Acquisition for Extracting Temporal Relations
Son Bao Pham and Achim Hoffmann

Formal Grammars for Linguistic Treebank Queries
Mark Dras and Steve Cassidy

Extracting Exact Answers using a Meta Question Answering System
Luiz Augusto Pizzato and Diego Molla

Multimedia Presentation of Grammatical Description: Design Issues
Simon Musgrave

16:30-17:00 *Structuring Documents Efficiently*
Robert Marshall, Steven Bird and Peter Stuckey

17:00-17:30 *Round-trip Translation: What Is It Good For?*
Harold Somers

17:30-18:00 *Evaluating the Utility of Appraisal Hierarchies as a Method for Sentiment Classification*
Jeremy Fletcher and Jon Patrick

DAY 2 — 11 DECEMBER, 2005

09:30-10:00 *Efficient Grapheme-phoneme Alignment for Japanese*
Lars Yencken and Timothy Baldwin

10:00-10:30 *Statistical Interpretation of Compound Nominalisations*
Jeremy Nicholson and Timothy Baldwin

10:30-11:00 *Paraphrase Identification by Text Canonicalization*
Yitao Zhang and Jon Patrick

11:00-11:30 Coffee Break

11:30-12:30 *Text Summarization: News and Beyond*
Invited Speaker – Kathy McKeown

12:30-14:00 Lunch

14:00-15:15 POSTER SESSION 2

Words and Word Usage: Newspaper Text versus the Web
Vinci Liu and James R. Curran

Automatic Induction of a POS Tagset for Italian
Raffaella Bernardi, Andrea Bolognesi, Corrado Seidenari and Fabio Tamburini

A Dual-Iterative Method for Concept-Word Acquisition from Large-Scale Chinese Corpora
Guogang Tian and Cungen Cao

Programming With Unrestricted Natural Language
David Vadas and James R. Curran

Identifying FrameNet Frames for Verbs from a Real-Text Corpus
Matthew Honnibal and Tobias Hawker

A Distributed Architecture for Interactive Parse Annotation
Baden Hughes, James Haggerty, Joel Nothman, Saritha Manickam and James R. Curran

Multi-document Summarisation and the PASCAL Textual Entailment Challenge
Nicola Stokes and Eamonn Newman

Design and Development of a Speech-driven Control for a In-car Personal Navigation System
Ying Su, Tao Bai and Catherine I. Watson

15:15-15:45 *Combining Confidence Scores with Contextual Features for Robust Multi-Device Dialogue*
Lawrence Cavedon, Matthew Purver and Florin Ratiu

15:45-16:15 *Automatic Utterance Segmentation in Instant Messaging Dialogue*
Edward Ivanovic

16:15-16:45 Coffee Break

16:45-17:45 *From Non-segmenting Language Processing to Web Language Engineering*
Invited Speaker – Virach Sornlertlamvanich

17:45-18:00 Award Ceremony and Closing Remarks

Dimensions of Deep Grammar Validation

Dan Flickinger
Oslo, Saarland, and Stanford Universities
danf@csli.stanford.edu

Abstract

In order to arrive at a more disciplined approach to the sustained development of linguistically rich grammars, I present a methodology for grammar validation, identifying principal dimensions of the task, and illustrating the application of the method for one release cycle of the open-source English Resource Grammar.

1 Introduction

Broad-coverage grammars for natural language processing which encode rich linguistic description are resources which develop relatively slowly, through many iterations of modification and testing, and ideally through exposure to the demands of a variety of NLP tasks. As with most large software components, such grammars are designed to exhibit a complexity of behavior which presents serious challenges for quality assurance from one release of a grammar to the next release. There is by now a substantial literature on many aspects of the evaluation of NLP software, including grammars, much of it focused around 'black box' or functional evaluation within some specific task domain(Hirschman and Thompson, 1998), (Sparck Jones and Galliers, 1998). Methods and tools have also been developed for 'glass box' or structural evaluation (EAGLES, 1996), with one line of work analyzing the internal formal coherence of such deep grammars, for example to flag inconsistencies or identify unused rules or constraints in the grammar code (Broeker, 2000), (Barr and Siefring, 2004), and another line of work using paraphrase generation to illuminate properties of a bi-directional grammar not easily detected when parsing (Dymetman and Isabelle, 1988). And finally, there has been work on developing a methodology of sustained grammar development, whereby these labor-intensive, long-lived resources undergo periodic modification intended to improve corpus coverage, or processing efficiency, or linguistic precision, or more typically all three at once (Oepen and Flickinger, 1998). Such a methodology must incorporate a set of disciplined procedures for validation of the resulting grammar, ensuring that it meets the expectations of its engineers, and explicitly communicates the interface specifications for its use in applications.

In this talk I describe an instantiation of a method for natural language grammar validation (cf. (Barr and Klavans, 2001)) to identify and motivate the multiple dimensions of the procedure, and to illustrate how many of the the tools and techniques proposed in the NLP evaluation literature are being exploited by engineers of large deep grammars. For concreteness, I present the method used in maintaining and extending the English Resource Grammar (ERG (Flickinger, 2000), a semantically precise, broad-coverage Head-driven Phrase Structure Grammar (HPSG) implementation used for both parsing and generation in several NLP applications, being developed within the Deep Linguistic Processing with HPSG Initiative (DELPH-IN: www.delph-in.net).

At the heart of this method is the use of a growing set of test suites of two kinds, together with a sophisticated grammar profiling tool, [incr tsdb()], which collects and preserves competence and performance data on these test suites. Some of the test suites consist of sets of hand-built example sentences and ungrammatical strings illustrating core linguistic phenomena, while the other class of test suites contain naturally occurring text items drawn from corpora, intended to be representative of the language data expected for a particular domain or task. For each of these test suites, human annotators have identified the intended analysis for each item out of the exhaustive set of analyses that the grammar supplied (up to the limits of available computational resources), thus providing 'gold standard' treebanks against which a modified version of the grammar can be mea-

Proceedings of the Australasian Language Technology Workshop 2005, pages 1–3,
Sydney, Australia, December 2005.

sured. Employing test suites from a variety of domains and tasks not only provides a basis for 'snapshot' profiles that enable a historical view of the grammar's development, but more importantly protects against overfitting of the grammar to just the current domain's data.

The principal dimensions in this grammar validation stand in some interesting tension with each other, since in each incarnation the grammar seeks to maximize (1) **robustness** in the range of phenomena and sheer quantity of data it processes; (2) **precision** linguistically in its mappings between strings and semantic representations; (3) **efficiency** in its consumption of CPU and memory resources; and (4) **stability** in the mappings it previously assigned to the items in those test suites, to minimize adaptation costs for its customer base, and to minimize the cost of updating the gold standard treebanks. For each of these four dimensions, I will present tools and techniques used to evaluate the relevant properties of the grammar, and illustrate the tensions among these dimensions with examples from the existing inventory of test suites.

Other dimensions that enter into this validation method arise from the heterogeneity of contexts in which the grammar can be used, including (5) multiple NLP **processing systems** (including at least the LKB (Copestake, 2002) and PET (Callmeier, 2000)); (6) **bi-directionality** of processing with attendant demands on each of the principal dimensions above for both parsing and generation; (7) multiple **configurations** of the grammar, including variants of preprocessing, unknown-word handling, root (start symbol) constraints, chart packing (for either parsing or generation, or both), and storage of the lexicon as text file vs relational database; (8) stochastic **parse/realization selection** or disambiguation (Oepen et al., 2004), particularly for highly ambiguous items where resource limitations become a crucial factor; and of course (9) the demands of **multiple applications**, currently including generation for Norwegian-to-English machine translation in the LOGON project (Lønning et al., 2004), extraction of ontological relationships by parsing dictionary definitions, and robust interpretation of transcribed conversations via enriched annotations of rhetorical relations.

Finally, for the grammar to be of use to application developers without an overly intimate knowledge of it, each release must be accompanied by (10) external **interface specifications** whose currency must be validated. The content of these specifications is in part automatically generated from the implemented representations of lexical entries, lexical types and grammar rules, and in part manually maintained. Principal among these specifications is the SEM-I (semantic interface) (Flickinger et al., 2005), an exhaustive listing of each lexical semantic predicate and its salient properties, which should precisely determine the observed variation in the elementary predications within the Minimal Recursion Semantics (MRS (Copestake et al., 2006)) representations that the grammar produces or accepts as input. A second essential specification provides the set of available lexical entry types, particularly those for open-class words which will inevitably need to be added for each new application, whether automatically guessed via part-of-speech tagging or entered into the lexicon manually.

In the talk, I will provide a detailed step-by-step tour of this method of validation as applied to the ERG for one typical grammar update, noting along the way how each of the dimensions identified above comes into play, often more than once in the process, and illustrating the interactions among several pairs of these dimensions in arriving at what must always be a compromise resolution of the tensions inherent in grammar engineering.

2 Acknowledgements

This work draws from research on the ERG that has been supported by several funding sources to the LinGO project at CSLI, Stanford University, over many years, including two grants from the German BMBF as part of the Verbmobil project, two National Science Foundation grants (IRI-9612682 and BCS-0094638), two grants from NTT Communication Science Laboratories, and one grant from Scottish Enterprise through the Stanford- Edinburgh Link. In addition I am grateful for current support from the LOGON project at Oslo University, and from the Computational Linguistics department at Saarland University. For the issues discussed in this talk, I am especially grateful for my recent collaborations with Stephan Oepen, Ann Copestake, Francis Bond, and Tim Baldwin.

References

Valerie Barr and Judith Klavans. 2001. Verification and validation of language processing systems: Is it evaluation? *Proceedings of the Workshop on Evaluation Methodologies for Language and Dialogue Systems, ACL2001.*

Valerie Barr and Ellen Siefring. 2004. Verification of lexicalized tree adjoining grammars. *Online Proceedings of the Seventh International Workshop on TAG and Related Formalisms.*

Norbert Broeker. 2000. The use of instrumentation in grammar engineering. *Proceedings of COLING 2000.*

Ulrich Callmeier. 2000. PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):99 – 108.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2006. Minimal Recursion Semantics. An introduction. *Research on Language and Computation.* (to appear).

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars.* CSLI Publications, Stanford, CA.

Marc Dymetman and Pierre Isabelle. 1988. Reversible logic grammars for machine translation. *Proceedings of the 2nd International Conference on Theoretical and Methodological Issues in the Machine Translation of Natural Languages.*

EAGLES. 1996. *Evaluation of Natural Language Processing Systems. Final Report EAG-EWG-PR.2.* EU Report.

Dan Flickinger, Jan Tore Lønning, Helge Dyvik, Stephan Oepen, and Francis Bond. 2005. SEM-I rational MT: Enriching deep grammars with a semantic interface for scalable machine translation. In *Machine Translation Summit X*, Phuket. (to appear).

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1):15 – 28.

Lynette Hirschman and Henry Thompson. 1998. Overview of evaluation in speech and natural language processing. In G. Varile and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, New York.

Jan Tore Lønning, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, Victoria Rosén, and Erik Velldal. 2004. LOGON. A Norwegian MT effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden.

Stephan Oepen and Dan Flickinger. 1998. Towards systematic grammar profiling. Test suite technology ten years after. *Journal of Computer Speech and Language*, 12 (4):411 – 436.

Stephan Oepen, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Language and Computation.*

Karen Sparck Jones and Julia Galliers. 1998. *Evaluating Natural Language Processing Systems.* Springer-Verlag, Berlin.

Text Summarization: News and Beyond

Kathleen McKeown
Department of Computer Science
Columbia University

Redundancy in large text collections, such as the web, creates both problems and opportunities for natural language systems. On the one hand, the presence of numerous sources conveying the same information causes difficulties for end users of search engines and news providers; they must read the same information over and over again. On the other hand, redundancy can be exploited to identify important and accurate information for applications such as summarization and question answering.

Columbia's Newsblaster system for online news summarization exploits online redundancy to generate a summary, at the same time creating a concise synopsis of recent events for end users. Newsblaster crawls the web nightly for news articles, clusters news on the same event and generates a summary of each event. In this talk, I will present the current capabilities of Newsblaster, with some focus on its ability to generate and edit text. I will then turn to our ongoing work which goes beyond summarization of English news. Our research on summarization of multilinguual news requires us to deal with noisy input; we rely on state of the art machine translation systems and use information that is available at the time of summarization to improve the fluency of the summary. We are also moving to summarization of other media, including email and meetings. Both of these media also require the ability to handle noisy input, but add an additional challenge to handle features of dialog.

Proceedings of the Australasian Language Technology Workshop 2005, pages 4–4,
Sydney, Australia, December 2005.

From Non-Segmenting Language Processing to Web Language Engineering

Virach Sornlertlamvanich
Thai Computational Linguistics Laboratory (TCL), NICT, Thailand
virach@tcllab.org

It is interesting to look at the statistics of the online languages reported by the Global Reach (www.global-reacg.biz). In September 2004, it was reported that the top six online language populations were English 35.2%, Chinese 13.7%, Spanish 9.0%, Japanese 8.4%, German 6.9%, and French 4.2% while the web contents were English 68.4%, Japanese 5.9%, German 5.8%, Chinese 3.9%, French 3.0% ,and Spanish 2.4%. There are some changes in ranking between the online language populations and the existing of the web contents. However, English is still the majority language used in the online community. Many efforts have been making to prevent the fall-off in using of other languages, especially the less computerized languages. It is said that there are about 7,000 languages using in all over the world. At the same time the less computerized languages are disappearing. The Rosetta Project (http://64.81.54.21:8080/live/) is a global collaboration to build an online archive of all documented human languages. The Language Observatory Project (www.language-observatory.org) initiated by Nagaoka University of Technology to search for the disappearing languages.

To deal with languages as many as we can find online, it is much more efficient to consider the language independent approaches. The big difference between segmenting languages (i.e. English and other European languages) and non-segmenting languages (i.e. Thai, Lao, Khmer, Japanese, Chinese and a lot of Asian languages) in the existing of word boundary marker causes the change in language processing. Most of the current approaches are based on the assumption that words are already identified disregarding the existing of the word boundary markers. The research on word boundary is separately conducted under the topic of word segmentation. On contrary, we proposed some algorithms to handle the non-segmenting languages (Virach 2005a, Virach 2005b) to establish a language independent approach.

In our recent research, we proposed a language interpretation model to deal with an input text as a byte sequence rather than a sequence of words. It is an approach to unify the language processing model to cope with the ambiguities in word determination problem. The approach takes an input text in the early stage of language processing when the exhaustive recognition of total word identity is not necessary. In our research, we present the achievements in language identification, indexing for full text retrieval, and word candidate extraction based on the unified input byte sequence. Our experiments show comparable results with the existing word-based approaches.

In our statistical-based word extraction research (Virach et al. 2000), it was reported to yield about 30% of the total word candidates being the unregistered words of a published dictionary, when the recall threshold was set to 56%. Character-based mutual information and entropy provided significant information to C4.5 algorithm for selecting appropriate candidates for words. The approach greatly supported the process of developing a dictionary, and later was extended to fulfill a dictionary-less search engine (Virach et al. 2003). The search engine had introduced a word score as a heuristic value to determine the word likelihood of a string. The word score was a normalized value of a mutual information value. The minimum score of the left and right hand side of a string in question was assigned as the word score of the string. Based on the proposed approach, we successfully implemented a multi-lingual search engine with minimum modification.

Language identification (Canasai et al. 2005) is yet another challenging task when it is done without any parsing knowledge. Byte sequence is the only magic key in our approach to determine the language of the input text. We introduced string kernel for this language identification task. We conducted experiments using 2 kernel classifiers i.e. centroid-based and support vector machine (SVM) methods. The accuracy of identification was acceptable for both methods. The accuracies reached 95 percent with only 10 training sets (2 KB per set). It was also found that the simple centroid-based classifier is comparable to the SVM classifier based on the string kernel.

Our approaches had been proven effective under the Thai language and the multi-lingual environment of 16 European and 4 Asian languages including Thai, Chinese, Japanese, Korean, English and many other

Proceedings of the Australasian Language Technology Workshop 2005, pages 5–6,
Sydney, Australia, December 2005.

European languages. We are expanding our corpus for conducting our experiments under the environment of a large number of languages.

Based on the successful results of word extraction, language identification and language independent indexing for search engine, we are conducting an experiment of the collaborative crawler on the high speed link (45 mbps) between Thailand and Japan. This collaborative work will provide an infrastructure for collecting web contents to study about the web language. The language together with its encoding of every webpage will be automatically identified and indexed to make the archive. Collaborative search engine will then go through all archives in all registered sites to present the ranked search results for any particular requests in any languages. The reports on the web languages from any perspectives can also be constructed by the proposed web language engineering.

Reference:

Virach Sornlertlamvanich. Implementations that Unify the Language Processing, Proceedings of the 9th NCSEC, University of Thai Chamber of Commerce, Bangkok, Thailand, pp. 1053-1062, 27-28 October, 2005.

Virach Sornlertlamvanich. Statistical-Based Approaches for Non-Segmenting Languages, Proceedings of Pacific Association for Computational Linguistics (PACLING), Meisei University, Tokyo, Japan, pp. 75-84, 24-27 August 2005.

Virach Sornlertlamvanich, Tanapong Potipiti and Thatsanee Charoenporn. Automatic Corpus-based Thai Word Extraction with the C4.5 Learning Algorithm. Proceedings of the 18th International Conference on Computational Linguistics (COLING2000).

Virach Sornlertlamvanich, Pongtai Tarsaku, Prapass Srichaivattana, Thatsanee Charoenporn and Hitoshi Isahara. Dictionary-less Search Engine for the Collaborative Database, Proceedings of The Third International Symposium on Communications and Information Technologies (ISCIT-2003), Songkhla, Thailand, 3-5 September 2003.

Canasai Kruengkrai, Prapass Srichaivattana, Virach Sornlertlamvanich, and Hitoshi Isahara. Language Identification Based on String Kernels, Proceedings of the 5th International Symposium on Communications and Information Technologies (ISCIT-2005), Beijing, China, October 12-14, 2005.

Disambiguating Conjunctions in Named Entities

Paweł MAZUR
Institute of Applied Informatics
Wrocław University of Technology
Wyb. Wyspiańskiego 27
50-370 Wrocław,
Poland
Pawel.Mazur@pwr.wroc.pl

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

Abstract

The recognition of named entities is now a well-developed area, with a range of symbolic and machine learning techniques that deliver high accuracy identification and categorisation of a variety of entity types. However, there are still some named entity phenomena that present problems for existing techniques; in particular, relatively little work has explored the disambiguation of conjunctions appearing in candidate named entity strings. We demonstrate that there are in fact four distinct uses of conjunctions in the context of named entities; we present the results of some experiments using machine-learned classifiers to disambiguate the different uses of the conjunction, with 81.73% of test examples being correctly classified. We provide some discussion and analysis of the problem of conjunction in named entities, and we show that there are some cases which are ambiguous even for humans.

1 Introduction

Initially developed as a component task in information extraction (see, for example, (Grishman and Sundheim, 1996)), named entity recognition, whereby entities such as people, organizations and geographic locations are identified and tracked in texts, has become an important part of other natural language processing applications such as question answering, text summarisation and machine translation.

Named entity recognition consists of both identifying those strings in a text that correspond to named entities, and then classifying each such named entity string as being of a specific type, with typical categories being Company, Person and Location. Sometimes an additional step is introduced for coreference resolution, which establishes whether two named entities in a given document refer to the same (either physical or abstract) object; so, for example, we might determine that the named entity *International Business Machines Corporation* has the same real world referent as both *IBM* and *Big Blue*. We can go further and determine whether named entities in separate documents (hence, cross-document coreference resolution) refer to the same entities (see (Bagga, 2004)), although this is a much less explored area.

All of the above assumes that identifying individual named entities in text is a relatively straightforward and well-defined task. However, although there are reported high performance figures for named entity identification and classification in general, there are some categories of named entities that remain problematic. We will refer to those strings in a text that may correspond to named entities as **candidate named entity strings**; one type of candidate named entity string that is problematic, from a surface linguistic perspective, is the category that contains conjunctions. Consider the string *Australia and New Zealand Banking Group Limited*: in the absence of an appropriate domain lexicon, an occurrence of this string within a document could be interpreted as either being the name of one company, or as being a conjunction of a location and a company name.[1] Determining the correct interpretation is clearly important for any application which relies on named entity extraction.

We have been working with a data set from the Australian Stock Exchange (ASX). This data set consists of a large set of company announcements: for a variety of regulatory reasons, listed companies provide around 100000 documents to the ASX each year, and the ASX subsequently makes these available to users via the web. Our goal is to take this data set (and similar data sets) and to add value to the documents by making use of language technologies.

[1] Such a conjunction may appear pragmatically odd, but, as we argue below, rejecting possibilities on such grounds requires a complicated set of rules.

Proceedings of the Australasian Language Technology Workshop 2005, pages 7–14,
Sydney, Australia, December 2005.

The overall approach taken in this work is described in (Dale et al., 2004).

The significance of the kinds of ambiguities we introduced above depends, of course, on the extent to which the phenomenon of conjunctions in named entities is widespread. Our current work focuses on a subcorpus of 13000 ASX documents. From this subcorpus, we selected 45 documents at random; in these documents, there were a total of 545 candidate named entity strings, of which 31 contained conjunctions. This informal sampling suggests that conjunctions appear, on average, in around 5.7% of candidate named entity strings;[2] however, in some documents in our sample, the frequency is as high as 23%. These frequencies are sufficient to suggest that the seeking of an appropriate means of handling conjunctions is a worthwhile and important pursuit.

For present purposes, we define a candidate named entity string as any sequence of words beginning with initial capitals, with the possible inclusion of a single instance of the word *and* or the form *&* internal to the string.[3] An examination of the candidate named entity strings appearing in our corpus reveals four distinct uses of the conjunction, as exemplified in the following:

1. Oil and Gas Ltd
2. Agfa and Fuji
3. John and Mary Smith
4. Company Secretary Resignation and Appointment

In example (1), we have a single named entity that happens to contain an internal conjunction; in example (2), we have a conjunction of two distinct named entities; and in examples (3) and (4), we have conjunctions that, from a linguistic perspective, contain a form of ellipsis, so that one conjunct is incomplete on its own, but can be completed using information provided in the other conjunct.

That conjunctions are problematic has been noted before, in particular by Mikheev et al. (1998), who suggested the strategy of examining the preceding document context to identify candidate conjuncts that should be considered as separate named entities. Mikheev et al. mention this approach being part of their system used in the MUC-7 competition, but no data is reported on the accuracy of this kind of heuristic; in our experience, there are many cases where there are no antecedent mentions that can be used in this way. Furthermore, in the MUC-7 data, strings like *John and Mary Smith* were considered as one named entity, whereas we take the view that, for many information extraction applications, it is important to recognize that this string represents two distinct entities.

2 Problem definition

Following from the above, we distinguish four[4] categories of candidate named entity strings containing conjunctions.

A: Name Internal Conjunction: This category covers those cases where the candidate named entity string contains one named entity, where the conjunction is part of the name. Some examples from our corpus:[5] *Copper Mines and Metals Limited*, *Herbert P Cooper & Son*, *Ernst & Young*, *Acceptance and Transfer Form*, and *Fixing and Planning Phase*.

B: Name External Conjunction: This category covers those cases where the conjunction serves to separate two distinct named entities. Some examples from our corpus: *Proxy Form and Explanatory Memorandum*, *Hardware & Operating Systems*, *John Travolta and Robin Wright Penn*, and *EchoStar and News Corporation*.

C: Right-Copy Separator: This category of conjunction separates two named entities, where the first is incomplete in itself but can be completed by copying information from the right-hand conjunct. This is perhaps most common in conjunctions of

[2] For comparison, a check on the MUC-7 evaluation data shows that, in that corpus, the proportion of candidate named entity strings containing conjunctions is 4.5%, so our corpus appears not particularly unusual in this regard.

[3] This is clearly an overly restrictive definition, but it appears to account for a large proportion of the cases we are interested in.

[4] Conceptually, we might view the last two categories as subtypes of the more general category **Copying Separator**; however, it makes sense to keep the two categories separate, since the process of reconstructing the unelided conjuncts is different in each case.

[5] In what follows, we treat the ampersand (*&*) and the full lexical item *and* as being equivalent; however, as we discuss later, there are cases where it may be useful to distinguish the two forms.

proper names, as in *William and Alma Ford*, but appears in other contexts as well. Some examples from our corpus: *Connell and Bent Streets*, *Eastern and Western Australia*, and *Melbourne and Harvard Universities*.

D: Left-Copy Separator: This is similar to the previous category, but instead of copying information from the right-hand conjunct, in order to complete the constituent named entities we need to copy information from the left conjunct. Examples in our corpus: *Hospital Equipment & Systems*, *J H Blair Company Secretary & Corporate Counsel*.

In the data we have analysed, most examples are either Name Internal or Name External, and Left-Copy Separators are the rarest. It should be noted that the Copy Separator categories have been explored within linguistic treatments of conjunction, particularly as found in Categorial Grammar (see, for example, (Steedman, 1985)), although linguistic analyses tend to focus on conjunctions involving common nouns rather than proper names.

We could attempt to distinguish the different uses of the conjunction by means of some heuristics. For example, if a candidate named entity string matches the pattern ⟨GivenName and GivenName FamilyName⟩, the conjunction is probably a Right-Copy Separator; and if it matches the pattern ⟨CompanyName and CompanyName⟩, the conjunction is most likely a Name External Conjunction. However, analysis of a reasonably large sample makes it clear that there are many different cases to be considered, and the heuristics required are difficult to derive by hand; a significant reason for this is that the names of people, companies, and locations, as well as other less common named entity types, may occur in many different combinations.

Consequently, we decided to view the problem as one of classification: given a particular instance of the conjunction and its left and right conjuncts, we want to determine, via machine learning, which category the conjunction belongs to.

3 Experimental Setup

We carried out an experiment to determine how machine learning algorithms cope with the problem of conjunction classification in named entities. In the work described in this paper, we limited the experiment to candidate named entity strings containing a single occurrence of the conjunction *&* or *and*.

In our approach, before we attempt to disambiguate the conjunctions we first tag the constituents of each candidate named entity string with their types. This step also recognizes multi-word elements where there is no ambiguity (for example, in the case of unambiguous person and company names); for example, *Australia and New Zealand Banking Group Limited* should be recognized as a sequence of tokens whose types are marked as Loc and Loc Org CompDesig, where the second Loc tag corresponds to the pair of tokens *New Zealand*.

Table 1 lists the 21 tags we use to annotate the tokens. Some of these, such as Loc, Org, GivenName, AlphaNum, Dir, and PersDesig, are the same as those used by many other named entity recognizers; there are also two tags that come from part-of-speech tagging (Noun and Adj). In our corpus, we note that there are a number of frequently occurring key terms which can be thought of as closed class items that are highly indicative of named entity type, and so we also use a number of tags to annotate specific words that perform a key role in distinguishing the categories (Form, Son, Project, System, and Phase). In our corpus there are 160 occurrences of *Form(s)* tokens, 102 of *Son(s)*, 75 of *Project(s)*, 56 of *System(s)* and 4 occurrences of *Phase(s)*.

Some additional comments are appropriate by way of explanation of some of the tags:

- Fac (Facility) is a general-purpose category intended to cover 'domain objects': names of buildings, meeting places, and worksites. Examples are *Ashmore Tavern*, *Imperial Hotel*, *Parkway Plaza*, and *Solano Mall*.
- CompDesig (Company Designator) is used for those tokens that unambiguously mark the occurrence of a company name, such as *Ltd*, *Limited*, *Pty Ltd*, *GmbH*, and *plc*; we also use this tag for much longer and not so obvious multi-word sequences like *Investments Pty Ltd*, *Management Pty Ltd*, *Corporate Pty Ltd*, *Associates Pty Ltd*, *Family Trust*, *Co Limited*, *Partners*, *Partners Limited*, *Capital Limited*, and *Capital Pty Ltd*.

By assigning tags to tokens we obtain a **pattern** which represents the named entity candidate string. For example, for the string *Herbert P Cooper and Son Ltd* the

No	Tag	Meaning
1	Loc	The name of a location
2	Org	The name of an organization
3	GivenName	A person's given name
4	FamilyName	A person's family name
5	Fac	A facility
6	Initial	An initial in the range A-Z
7	CompPos	A position within a company
8	Abbrev	Abbreviation
9	PersDesig	A person designator
10	CompDesig	A company designator
11	Son	*Son(s)*
12	Dir	A compass direction
13	AlphaNum	An alphanumeric expression
14	Day	The name of a day
15	Month	The name of a month
16	Adj	An adjective
17	Noun	A noun
18	Project	*Project(s)*
19	System	*System(s)*
20	Phase	*Phase(s)*
21	Form	*Form(s)*

Table 1: The tags used for text annotation.

pattern is ⟨GivenName Initial FamilyName & Son CompDesig⟩.

For the purposes of machine learning, we then encode each pattern in the following way. We create an attribute for each of the 21 tag types for each of the left and right sides of a conjunction, for a total of 42 attributes. The attributes are of integer type with values $\{0, 1\}$, thus signaling either the presence or absence of a token of that type anywhere within either conjunct. We will refer to this encoding as the **Basic Encoding**.

On the basis of the results we obtained using the Basic Encoding (see Section 5.1), we created an additional five attributes for each conjunct: GivenNameCount, FamilyNameCount, InitialCount, NounCount and AdjCount. They serve as counts of the number of occurrences of the relevant token types, and have non-negative integer values. This extended set of 52 features will be referred to as the **Extended Encoding**.

With each data instance there is associated a ConjType attribute with the values {A,B,C,D}; this is used to encode the category of the conjunction in the training or test example.

The corpus used for our research consisted of a 13460 document sub-corpus drawn from a larger corpus of company announcements from the Australian Stock Exchange. The documents range in length from 8 to 1000 lines of text.

Choosing training and test examples was carried out in a number of steps. First, candidate named entity strings containing sequences of words with initial capitals, and an embedded conjunction, were extracted using a Perl script. This provided over 10560 candidate named entity string instances, corresponding to 6645 unique forms. For this experiment we did not collect candidate named entity strings containing lowercased prepositions and determiners such as *of*, *in*, *a*, and *the*, although clearly many relevant named entities will contain these elements.

From this set we randomly selected examples for our training and test data sets. Each random selection was followed by hand elimination of examples that turned out to be wrongly identified as candidate named entity strings in the text.[6]

The experiment consisted of two test runs, one using the basic encoding and the second using the extended encoding, as described above. In each case we used the same set of 348 training instances, and the same set of test data with 197 previously unseen examples.

Table 2 presents for each data set the distribution of examples across the four categories of conjunction.[7]

Data Set	A	B	C	D	Sum
Training	135	160	35	18	348
Test	55	119	15	8	197

Table 2: Data sets sizes and example distributions in categories.

4 The Algorithms

The experiment was conducted using the WEKA toolkit (Witten and Frank, 2005). This provides implementations of several machine learning algorithms, along with the data structures and code needed to perform data input

[6]Wrong identification occurred due to typographic features such as ASCII formatted tables, paragraphs in all upper case, sentences or titles where every word contained an initial capital, and punctuation errors in the source texts.

[7]By way of comparison, the corpus used for the MUC-7 final evaluation contains 53 strings corresponding to our category A and 124 strings for category B; the data from the training phase of the competition contains 28 category A strings and 81 category B strings. Recall from above that the MUC data does not recognize our categories C and D.

and output, data filtering and results evaluations and presentation.

After some initial exploration using a variety of algorithms for supervised machine learning available in WEKA, we chose six which gave the best results: the Multilayer Perceptron, two lazy algorithms (IBk and K*), and three tree algorithms: Random Tree, Logistic Model Trees and J4.8. We also include here the results for Naïve Bayes, given the popularity of this method in the field.

5 Results

We observe (see Table 2) that conjunctions of category B (Name External Conjunction) are the most frequent in our annotated data set. This gives us a simple baseline for comparison: by choosing the most frequent category by default, we would achieve a correct classification rate of 60.41%.

Tables 3 and 4 present detailed results of the two test runs using the classifiers trained on our feature set; recall that the extended encoding takes account of the number of instances of specific token types, rather than just a binary distinction between presence and absence. We show the number of correctly classified examples for both the training and test data sets.

Algorithm	Training	348	Test	197
Naïve Bayes	71.84%	250	68.53%	135
Mult. Perc.	91.95%	320	80.20%	158
IBk	92.24%	321	74.62%	147
K*	92.24%	321	74.62%	147
Random Tree	92.24%	321	77.16%	152
LMT	92.24%	321	81.22%	160
J4.8	87.36%	304	77.67%	153

Table 3: Results for basic encoding.

Algorithm	Training	348	Test	197
Naïve Bayes	70.11%	244	64.47%	127
Mult. Perc.	93.97%	327	77.67%	153
IBk	93.97%	327	75.13%	148
K*	93.97%	327	75.13%	148
Random Tree	93.97%	327	72.08%	142
LMT	93.68%	326	81.73%	161
J4.8	87.36%	304	77.67%	153

Table 4: Results for extended encoding.

5.1 Basic Encoding

When looking at the results from the classifiers, it turns out that patterns like ⟨Noun & Noun⟩ or ⟨Noun & Noun Noun⟩ made the biggest contribution to misclassification; also patterns built from combinations of several Nouns and some other tags such as Adj or Org had a significant impact on making the results worse. An interesting subgroup for us are patterns with tags Project, System and Form. It turns out that these domain specific tags were not sufficient to categorize test instances correctly.

The third large group of difficult examples are those that are represented by long patterns that consist of several kinds of tags. They also usually contain up to three Noun tags; examples of these patterns are as ⟨Noun GivenName Org & Noun Noun Noun⟩ and
⟨Abbrev CompDesig & Adj Noun CompDesig⟩.

The fourth group of problematic cases are patterns based on the FamilyName tag:
⟨FamilyName & FamilyName⟩,
⟨FamilyName & FamilyName Loc⟩,
⟨FamilyName Loc & FamilyName Loc⟩,
⟨FamilyName & FamilyName Noun⟩,
⟨FamilyName Noun & FamilyName Noun⟩,
⟨Initial & Initial FamilyName⟩.

5.2 Extended Encoding

An observation we made as a consequence of the results obtained for the basic encoding was that complexity and length of conjuncts appeared to play a role in misclassification; accordingly, for the extended encoding we introduced ten new attributes, in the form of five counters for each side of a conjunction. It turned out that new information was only slightly helpful for K* and LMT (in both cases the gain in performance was one example); for two algorithms there was no impact; and the Multilayer Perceptron and Random Tree algorithms encountered a significant drop in performance, by, respectively, five and ten examples (see Table 4). For both the Perceptron and Random Tree algorithms there were some cases where classification improved, and some for which it got worse.

For the Random Tree algorithm, nine additional examples were classified correctly; as expected, these were cases involving long patterns of up to 9 tags. However, 19 examples that were correctly classified using the basic encoding were misclassified with the extended encoding; most of these consisted of tags related to people or company names.

For the Multilayered Perceptron algorithm, the extended encoding improved performance on short patterns but made things worse on long

patterns.

5.3 General remarks

All the algorithms presented here performed far above the baseline in each test run. The results for the training data from each test (Tables 3 and 4) show that, in the training data, there are about 7% of examples which could be perceived as genuinely ambiguous. On investigation, the vast majority of these examples belong to several (usually two) categories and the most ambiguous are of ⟨Noun & Noun⟩ and ⟨FamilyName & FamilyName⟩ patterns, which we discuss later. The instances of these patterns appeared ambiguous to a human annotator.

The best result, 81.7259% of examples classified correctly, was achieved with the LMT algorithm and the extended set of attributes. The precision, recall and F-measure for this case are presented in Table 5. Table 6 provides a confusion matrix with the desired and actual classification of examples.

The LMT algorithm was also the best algorithm in the test run using the basic encoding of 43 features. The second best algorithm was the Multilayer Perceptron, which scored 80.2% and 78.2% with the basic and extended encodings respectively. However, the difference between the results for the two algorithms is not statistically significant.

Category	Precision	Recall	F-Measure
A	0.658	0.909	0.763
B	0.969	0.790	0.870
C	0.667	0.667	0.667
D	0.778	0.875	0.824
weighted mean	0.851	0.817	0.823

Table 5: Detailed accuracy by category of conjunction for best result (LMT, Extended Encoding).

A	B	C	D	→ classified as ↓
50	22	3	1	A
1	94	2	0	B
4	1	10	0	C
0	2	0	7	D

Table 6: Confusion matrix for best result (LMT, Extended Encoding).

6 Analysis

6.1 Data preparation

It is important to stress that our experiment was conducted in the specific domain of company announcements from the Australian Stock Exchange; these documents have some features that are not necessarily typical for others. In particular, texts in this domain frequently have some of the characteristics of legal documents, where many sometimes apparently arbitrary elements are given initial capitals: typical examples from our corpus would be expressions like *Primary and Secondary*, *Profit & Loss*, *Receivers and Managers* or *Resource and Reserve*. Many of these are high frequency terms, and so could be filtered out in a separate preprocessing stage; however, a complicating factor here is that casing is not used consistently by some authors. A more general problem is that of titles, for example titles of books, documents and document elements such as tables and headings; the unrestricted productivity of these kinds of terms means that they are not easily characterisable by patterns of the kind explored here, and would be more appropriately handled by a more syntactically driven model.

The results of our conjunction disambiguation process are very dependent on the tags assigned in the preprocessing stage. It can make a big difference, for example, whether a substring is recognized as a sequence of Nouns or as one Location name. Extensive gazetteers can play a role here, but some cases are ambiguous even for humans. For example, *Trustees Executors* is a company name;[8] but if this is not detected during tagging, it is tagged as ⟨N N⟩, which is much more ambiguous and impacts on performance. Similarly, a string like *Boyer and Haro Fields* can be difficult to analyze correctly without recourse to extensive world knowledge.

6.2 Error analysis

6.2.1 One Pattern, Many Categories

When investigating the misclassified strings we found that there are many cases where a given pattern belongs to more than one category. For example, we have the following in the training data:
String: *Ernst and Young Consulting*
Pattern: ⟨FamilyName & FamilyName Noun, A⟩
String: *National Parks and Wildlife Service*
Pattern: ⟨Adj Noun & Noun Noun, A⟩

[8] *Trustees Executors* was the first trustee company in New Zealand, established in 1881.

but in the test data we have:
String: *Boyer and Haro Fields*
Pattern: ⟨FamilyName & FamilyName Noun, C⟩
String: *General Meeting and Proxy Votes*
Pattern: ⟨Adj Noun & Noun Noun, B⟩

The ⟨Noun & Noun⟩ and ⟨FamilyName & FamilyName⟩ patterns are particularly prone to this ambiguity; we discuss these cases further below.

These examples suggest that our feature set is not capturing enough distinctions to enable correct classification; providing richer information about the candidate string would help, drawing from morphological, syntactic, semantic and even pragmatic features, should these be available.[9]

In different models created on the basis of training data by individual classifiers, a different category is assigned to these ambiguous patterns. Since the number of examples of a given pattern is not the same for particular categories, the number of misclassified examples differs for particular models. For example, there are ten instances of ⟨Noun & Noun⟩ pattern in the test data. Four of them are of category A, and six of them are of category B. The LMT algorithm assigns for this pattern probabilities $p_A = 0.455$, $p_B = 0.409$, $p_C = 0.136$ and $p_D = 0$. When a ⟨Noun & Noun⟩ example from the test data is to be classified, the classifier chooses category A as the most likely. On the other hand, the Multilayered Perceptron algorithm derives the following probabilities: $p_A = 0.414$, $p_B = 0.476$, $p_C = 0.110$ and $p_D = 0$. Consequently, it classifies all strings matching the ⟨Noun & Noun⟩ pattern as being of category B.

6.2.2 ⟨Noun & Noun⟩

As noted in Section 5, some of the most ambiguous candidate strings are those whose patterns are based on nouns and adjectives. This is not surprising, since these are very general tags capturing everything which is not recognized as a genuine proper name, an alphanumeric sequence of characters, or any of the specific distinguished terms captured by our other tags. In these cases the correct categorisation of the conjunction usually depends on the context, which sometimes can be even the whole text. General knowledge about the world is sometimes also essential; consider, for example, the strings *Gummy & Kipper* or *Showtime and Encore*, whose nature is quite unclear without recourse to other knowledge sources. This problem concerns mainly categories A and B, since initcapped general terms can appear equally freely in both cases.

In the test with the extended encoding, six out of 36 incorrect classifications involved the pattern ⟨Noun & Noun⟩; in each case, these should have been categorised as instances of category B (Name External Conjunction), but were classified as category A (Name Internal Conjunction). There were also another three examples of slightly more complex patterns containing more nouns:

String: *Tourism and Hotel Management*
Pattern: ⟨Noun & Noun Noun⟩
String: *Placement Shares and Options*
Pattern: ⟨Noun Noun & Noun⟩
String: *Country Comfort and Chifley*
Pattern: ⟨Noun Noun & Noun⟩

6.2.3 ⟨FamilyName & FamilyName⟩

The ⟨FamilyName & FamilyName⟩ pattern also causes problems in classification. Because many company names are created as a conjunction of two surnames, it happens that our training data contains more examples of this pattern for category A (Name Internal) than for category B (Name External), and so the model built by LMT considers category A to have probability of 0.704. However, there are still some cases when ⟨FamilyName & FamilyName⟩ does not denote an organization. So, regardless of the quality of the training data, it is worth checking whether somewhere in the text there exists a corresponding string with the pattern ⟨FamilyName & FamilyName CompDesig⟩.

6.2.4 Domain Dependent Substrings

As for any domain, it may be possible to identify a set of specific recurring strings or patterns that could be recognized in a preprocessing step, so that the learning algorithm does not need to deal with these cases. In our data, these turn out to belong to the set of strings that cause a range of problems. Two such examples are *Explanatory Notes and Proxy Form*, *Information Memorandum and Proxy Form*. In our corpus of candidate named entities, the string *Proxy Form* occurs 35 times. A reasonable strategy would be to assume that any string of the form *... and Proxy Form* involves the use of a Name External conjunction.

[9] In the example shown here, *Ernst and Young* could also be detected during preprocessing on the basis of a company name gazetteer, but such lists will never be complete.

7 Conclusions and Future Work

We have analyzed the problem of conjunctions in candidate named entity strings; we distinguished four categories of conjunction that appear in these strings, noted that the appropriate disambiguation of these is a problem that requires attention, and defined the problem as one of classification. We then conducted an experiment whose aim was to determine whether the problem could be solved by means of machine learning algorithms.

We have shown that there are instances of conjunction which are difficult even for humans to classify correctly. Very often the decision requires extensive analysis of the content, and the use of general world knowledge. Given the existence of such cases, the results demonstrated here with machine-learned classifiers are very encouraging. There are several regards in which the work reported here can be improved further.

1. We have restricted ourselves to candidate named entity strings which contain a single conjunction; however, there are of course cases where multiple conjunctions appear. One category consists of examples like *Audited Balance Sheet and Profit and Loss Account*, where again the kinds of syntactic ambiguity involved would suggest a more syntactically-driven approach would be worth consideration. Another category consists of candidate named entity strings that contain commas as well as lexicalised conjunctions.
2. In the work described here, we did not distinguish between the two variants of the lexicalised conjunction (i.e., *&* and *and*). Obviously, these two forms are not used in text completely interchangeably: for example, it is relatively unusual to separate two person names (as in *Alex and Bill Smith*) using an ampersand.
3. Candidate named entity strings can contain other closed class items, such as *of* and *the*; extension of the treatment here to this larger class of candidate strings will introduce new complexities, but at the same time these terms may provide useful disambiguating features.

An issue that may be restricted to corpora that have a broadly legal character is the frequent appearance of candidate named entities that are made up of common nouns. In contrast to a 'substring recognition' approach that relies on information from gazetteers, many of these cases might be amenable to a more syntactically sophisticated analysis, and this is one place where work on conjunction from a linguistic perspective might provide some leverage; however, there then remains the issue of determining which approach to use in a given case.
Named entities containing conjunctions, and named entities separated by conjunctions, constitute a form of ambiguity that needs to be handled for high accuracy named entity extraction. We have shown that machine learning can achieve good results in resolving these ambiguities.

8 Acknowledgements

The work reported here was carried out while the first author was a visiting scholar at the Centre for Language Technology at Macquarie University.

References

Amit Bagga. 2004. Cross-Document Coreference: Methodologies, Evaluations, and Applications. In António Branco, Tony McEnery, and Ruslan Mitkov, editors, *Proceedings of 5th Discourse Anaphora and Anaphor Resolution Colloquium, Portugal, 23-24 Sep., 2004*.

R. Dale, R. Calvo, and M. Tilbrook. 2004. Key Element Summarisation: Extracting Information from Company Announcements. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, 7th-10th December 2004, Cairns, Queensland, Australia*.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference-6: A Brief History. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, Los Altos, Ca. Morgan Kaufmann.

A. Mikheev, C. Grover, and M. Moens. 1998. Description of the LTG System Used for MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proc. of a Conf. held in Fairfax, Virginia, 29 April-1 May, 1998*.

M. Steedman. 1985. Dependency and Coordination in the Grammar of Dutch and English. *Language*, 61:523–568.

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

Learning of Graph Rules for Question Answering

Diego MOLLA and **Menno VAN ZAANEN**
Centre for Language Technology, Macquarie University
Sydney,
Australia,
{diego,menno}@ics.mq.edu.au

Abstract

AnswerFinder is a framework for the development of question-answering systems. AnswerFinder is currently being used to test the applicability of graph representations for the detection and extraction of answers. In this paper we briefly describe AnswerFinder and introduce our method to learn graph patterns that link questions with their corresponding answers in arbitrary sentences. The method is based on the translation of the logical forms of questions and answer sentences into graphs, and the application of operations based on graph overlaps and the construction of paths within graphs. The method is general and can be applied to any graph-based representation of the contents of questions and answers.

1 Introduction

Text-based question answering (henceforth QA) is the process whereby an answer to an arbitrary question formulated in plain English is found by searching through unedited text documents and returned to the user. The current availability of increasingly large volumes of text for human consumption has prompted an intensive research in QA. A well-known forum for the evaluation of QA systems is the question-answering track of the Text REtrieval Conference[1] (Voorhees, 2001), where systems developed by some of the most active researchers in the area are compared within the context of a common task. In addition, QA technology is being deployed in practical applications. For example, several Web-based question-answering systems are currently available (e.g. START[2], AnswerBus[3]), and recently popular Web search engines have started incorporating automated question-answering techniques (e.g. Google[4], as for September 2005).

The development of successful QA technology requires solid foundations both in the areas of software engineering and natural language processing. The nature of text-based question answering requires the use of a wide range of techniques, some of which are described in (Hirschman and Gaizauskas, 2001; Voorhees, 2001). For example, traditional document retrieval techniques are typically used to preselect the documents or document fragments that may contain the answer to the question. In addition, information extraction techniques are commonly used to extract all the named entities in the question and the preselected text, on the ground that fact-based questions typically expect one of these named entities as an answer. To analyse the questions, techniques range from the use of regular expressions to the use of machine-learning techniques that classify the questions according to the type of the expected answer. Finally, to find the answer, techniques may vary from a bag of words comparison of keywords used in the question and the answer sentence, to the use of full parsers and logical proof tools such as OTTER[5]. Additional resources are typically used, notably the WordNet lexical resource.[6] Consequently, the most successful QA systems are complex pieces of engineering that require frequent development and testing, such as (Moldovan et al., 2003). An unwelcome side-effect of this is that much of the effort spent in developing a QA system is spent, not in the developing of QA methodologies, but in defining the optimal parameters of a system.

On the other hand, QA presents challenging theoretical issues. One of the most salient theoretical challenges is related to the problem of

[1] http://trec.nist.gov
[2] http://www.ai.mit.edu/projects/infolab/
[3] http://www.answerbus.com/index.shtml
[4] http://www.google.com
[5] http://www-unix.mcs.anl.gov/AR/otter/
[6] http://wordnet.princeton.edu/

Proceedings of the Australasian Language Technology Workshop 2005, pages 15–23,
Sydney, Australia, December 2005.

paraphrasing. There are many ways of expressing the same piece of information. For example, the simple question *Where was Peter born?* can be similarly asked as:

1. *In what city was Peter born?*
2. *What is Peter's birthplace?*
3. *What is the birthplace of Peter?*
4. *Name Peter's birthplace*

Whereas it may not be difficult to manually devise rules that account for the most popular ways of rephrase a question, variations in the sentences containing the answer are much less predictable. A human would not have any problem to find the answer to the above questions in the following examples:

1. *Peter was born in Paris.*
2. *Paris is Peter's birthplace.*
3. *Paris, Peter's birthplace, is located in France.*
4. *Mrs Smith gave birth to Peter in Paris.*

However, a machine would need to have access to lexical, syntactic, and world knowledge information if it is to find the answer.

The above are simple constructed examples. Real text with much more complex examples abounds, but the examples above suffice to illustrate the problem encountered by any text-based question-answering system. For further details about the problem of paraphrasing within the context of QA, see (Rinaldi et al., 2003).

Some systems have attempted to systematically build rules that link questions with answer sentences. For example, (Soubbotin, 2001) used a complex hierarchy of rules on surface strings. Other systems, such as (Echihabi et al., 2004), use a method for the automatic learning of surface-level rules. Other systems, such as (Bouma et al., 2005), use hand-crafted rules based on syntactic information.

Our hypothesis is that the accuracy of question-answering systems would improve if these rules are based on linguistic features located at a deeper level. Furthermore, to handle the problem of paraphrasing, the rules must be automatically learnt based on a representative corpus of questions and answers. In this paper we present our current work for developing and testing this hypothesis. Our work is being integrated in the AnswerFinder QA system, which is briefly described in Section 2. Section 3 describes the Logical Graph notation that we use to represent the logical contents of questions and answer sentences. Section 4 presents the rules based on Logical Graphs, and how they are automatically learnt from a corpus of question/answer pairs. Section 5 shows the use of these rules to find the exact answer to a question, and Section 6 shows the results of our evaluations. Sections 7 and 8 point to related research and give the final conclusions, respectively.

2 AnswerFinder, a Framework for Question Answering

Our solution to the need to use software engineering techniques for the development of practical QA systems is AnswerFinder. Initially designed as a simple QA prototype, AnswerFinder is currently being redesigned to allow the rapid development and test of QA techniques. Its primary application is the TREC Question Answering track,[7] but we also envisage its use, directly or indirectly, in other evaluation frameworks such as CLEF,[8] DUC,[9] and the PASCAL Recognising Text Entailment challenge.[10] For this reason, AnswerFinder's architecture is flexible and configurable, allowing to plug and test various modules easily.

The design of the system is functional and object-oriented. Focusing on function (instead of data) makes it easier to replace functions of the system with others. The system is implemented in C++. C++ was selected because it is a high-level language on the one hand, but can also be used in a more low-level way. It interfaces well with C, which allows for easy integration of many external systems. Furthermore, the resulting executable is relatively fast.

AnswerFinder consists of two main components, the client and the server. The client can get information from the server about the algorithms and files/document collections it provides to clients. The client can also send information to the server requesting question(s) to be processed using specific algorithms and data collections. The server can be fully configured via XML. For example, if the client calls

[7] http://trec.nist.gov
[8] http://www.clef-campaign.org/
[9] http://www-nlpir.nist.gov/projects/duc/
[10] http://www.pascal-network.org/Challenges/RTE/

the server without any configuration information, the server replies with an XML document listing all the available services. The client can then call the server with an XML file containing all the configuration information.

The request the server receives from the client contains all information needed to process the question(s). It specifies the document collection and the algorithms that should be used. The server then runs the required services by creating algorithm objects. An algorithm defines the full question-answering process, and it may use sub-algorithms for specific phases (such as question classification, document preselection, etc). The sub-algorithms are designed so that they can be called by any algorithm. Thus, different ways of trying QA techniques can be easily implemented by defining new algorithms that call the specific sub-algorithms with specific parameters.

The following sub-algorithms are currently defined in AnswerFinder. They are classified by the question-answering phase in which they are used:

Document Selection. Sub-algorithms in this phase are used to preselect the candidate documents.

- **NIST Doc Selection:** This sub-algorithm returns the documents provided by NIST for the TREC QA task.

Question Classification. Sub-algorithms in this phase are used to analyse and classify the question.

- **Regexp Q Classification:** This is a set of regular expressions that determines the type of the expected answer according to a simple hierarchy of types.

Sentence Selection. Sub-algorithms in this phase are used to determine what sentences are likely to contain an answer. These sub-algorithms can be cascaded to provide the final ranking of sentences.

- **Word Overlap:** Count the number of words in common between the question and the answer sentence. This sub-algorithm allows the use of a list of stop words that are not considered in the overlap.
- **Grammatical Relations Overlap:** Count the number of grammatical relations in common. We used a subset of the grammatical relations defined by (Carroll et al., 1998).
- **Logical Form Rules:** Count the number of logical form predicates in common, after applying a set of logical form rules. The process is explained in (Mollá and Gardiner, 2004).
- **Logical Graph Rules:** Count the graph overlap between the question and the answer after applying graph transformation rules. This process is explained in the remainder of this paper.

Named Entity. sub-algorithms in this phase are used to detect all named entities in the text (person and organisation names, locations, etc).

- **LingPipe:** This sub-algorithm uses the Alias-i LingPipe named entity recogniser.[11]

3 Logical Graphs

We are developing a graph notation for the expression of the logical contents of questions and answer sentences. Our Logical Graphs are inspired on Conceptual Graphs (Sowa, 1979), though our graphs do not attempt to encode the full semantics of a sentence. Instead, the focus of our Logical Graphs is on robustness and practicability.

Robustness. It should be possible to automatically produce the Logical Graph of any sentence, even of those sentences that are not fully grammatical. The importance of this feature becomes obvious once one looks at the quality of the English used in typical corpora used for QA.

Practicability. The Logical Graphs should be automatically constructed in relatively short run time. The operations with the graphs should be computable within relatively short time.

Like Sowa's Conceptual Graphs, our Logical Graphs are directed, bipartite graphs with two types of vertices, concepts and relations:

Concepts. Examples of concepts are objects *dog*, *table*, events and states *run*, *love*, and properties *red*, *quick*. Concepts may be arranged in a network of word relations (such as ontologies), though our method does not yet exploit this possibility in full.

[11] http://alias-i.com/lingpipe/

Relations. Relations act as links between concepts. Traditional examples of relations are grammatical roles and prepositions. However, to facilitate the production of the Logical Graphs we have decided to use a labelling of relations which is relatively close to the syntactic level of linguistic information. For example, instead of using the usual thematic roles *agent*, *patient*, and so forth, we use syntactic roles *subject*, *object*, etc. For convenience, and to avoid entering into a debate about the possible names of the syntactic roles, we have decided to use numbers. Thus, the relation *1* indicates the link to the first argument of a verb (that is, what is usually a subject). The relation *2* indicates the link to the second argument of a verb (usually the direct object), and so forth.

Figure 1 shows various examples of Logical Graphs. The first example shows the use of a relation *1* to express the subject of the *go* event, and two relations, *to* and *by*, that represent two prepositions. The second example shows the use of lattice structures to represent complex entities (such as the ones formed when a conjunction is used). This use of lattices is inspired from the treatment of plurals and complex events (Link, 1983; Mollá, 1997). Finally, the third example shows the expression of clauses and control verbs. These examples only cover a few of the linguistic features but we hope they will suffice to show the expressive power of our Logical Graphs.

The Logical Graphs are constructed automatically from the output of the Conexor dependency-based parser (Tapanainen and Järvinen, 1997). The choice of the parser was arbitrary, and it would be easy to produce the same or similar graphs from the output of any dependency-based parser. It would be also possible to use the output of a constituency-based parser by applying well-known methods to convert from constituency structures to dependency structures like those described by Schneider (1998), or practical methods like the one described by Harabagiu et al. (2000).

4 Logical Graph Rules

The Logical Graph rules used by AnswerFinder are based on the concepts of *graph overlap* and *path* between two subgraphs in a graph.

The *graph overlap* between two sentences is the overlap of the Logical Graphs of the two sentences. A naïve definition of the overlap between two graphs would be the graph consisting of all the common concepts and relations. The actual definition of an overlap, however, must be made more complicated on the light of the existence of repeated vertex labels. The third example of Figure 1, for example, shows that the relations named *1* and *2* appear twice in the same graph. Concept labels can also be repeated in a graph if the sentence uses the same word to express two different concepts. For example, the sentence *John bought a book and Mary bought a magazine* describes two distinct events of buying.

Graph overlaps must therefore be defined on the basis of a *correspondence relation* so that each vertex (edge) of a graph correlates with one and only one vertex (edge) in the other graph (Montes-y-Gómez et al., 2001). Thus, there is a projection from the graph overlap to a subgraph of each of the original graphs, such that there is a correspondence from every vertex (or edge) of the graph overlap to a vertex (or edge) of the projected subgraphs. Figure 2 shows an example of two graph overlaps and their projections to subgraphs in the original graphs.

There could be several overlaps between two graphs. Of these, the most useful ones are the *maximal overlaps*, that is, the overlaps that are not subgraphs of any other overlaps. There could still exist several maximal overlaps between two graphs. For example, Figure 2 shows two different maximal overlaps between the Logical Graphs of two sentences.

A *path* between two subgraphs in a graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ that connects the two subgraphs. As is the case with graph overlaps, there may be several paths between two subgraphs, especially when the graphs have a high density of edges.

Each rule r will contain three components. For the sake of completion the components are listed here but their use will be described in detail in Section 5.

r_o An overlap between a question and its answer sentence. This overlap is used to determine when the rule should trigger.

r_p A path between the overlap and the actual answer in the answer sentence. This path is used to find the location of the exact answer.

r_a A graph representation of the exact answer.

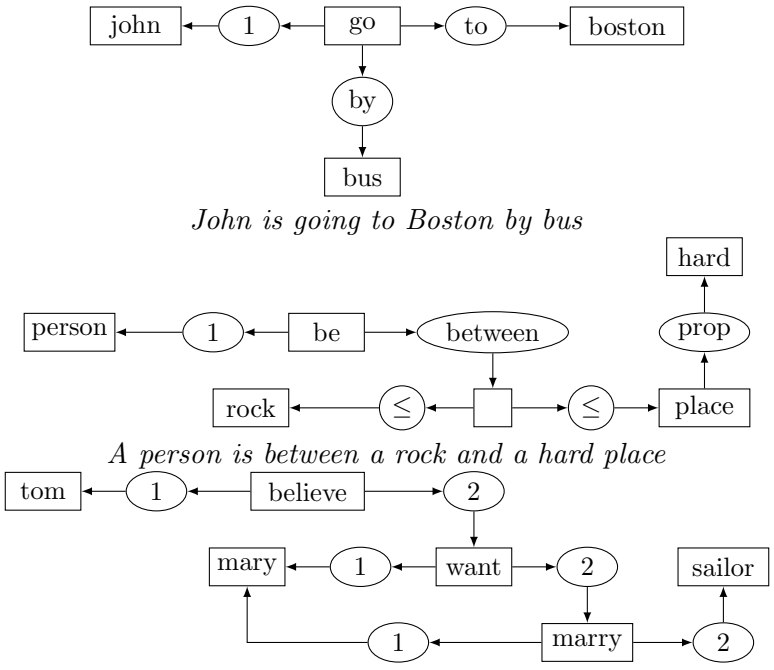


Figure 1: Examples of logical graphs

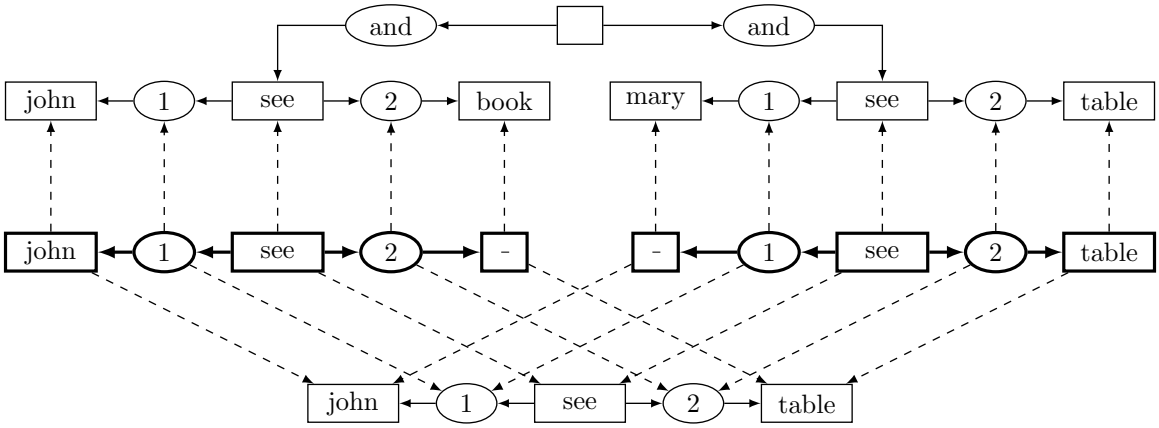


Figure 2: Graph overlaps of sentences *John saw a book and Mary saw a table* and *John saw a table*. The two overlaps are shown in thick lines. The dashed lines show the correspondence relation from the graph vertices of each overlap and the projected subgraphs in the original graphs (the correspondence relation from the edges is not shown to improve readability).

4.1 Learning of Logical Graph Rules

With the help of a training set of questions and sentences containing the answers, a set of Logical Graph rules can be learnt. Figure 3 shows an example of a rule learnt between two sentences. The graph notation has been simplified by replacing the relation vertices with labelled edges.

The algorithm for learning rules is fairly straightforward and is shown in Figure 4. Rules learnt with this algorithm are very specific to the question/answer pair. For example, the

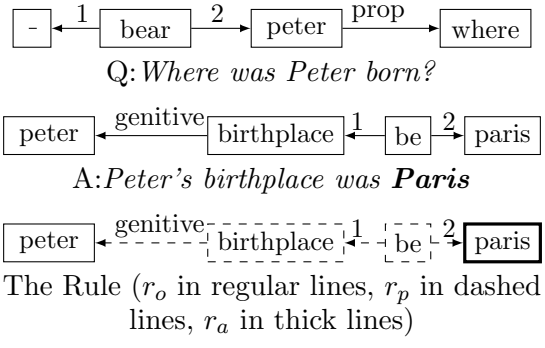


Figure 3: A logical graph rule

FOR every question/answerSentence pair
  $\mathcal{G}_q$ = the graph of the question
  $\mathcal{G}_s$ = the graph of the answer sentence
  $\mathcal{G}_a$ = the graph of the exact answer
  FOR every overlap $\mathcal{O}$ between $\mathcal{G}_q$ and $\mathcal{G}_s$
    FOR every path $\mathcal{P}$ between $\mathcal{O}$ and $\mathcal{G}_a$
      Build a rule $\mathcal{R}$ of the form
        $\mathcal{R}_o = \mathcal{O}$
        $\mathcal{R}_p = \mathcal{P}$
        $\mathcal{R}_a = \mathcal{G}_a$

Figure 4: Learning of graph rules

rule in Figure 3 would only trigger for questions about Peter and it would not trigger, say, for the question *Where was Mary born?*. The rule needs to be generalised. Our generalisation method is very simple: relations do not generalise (relations express syntactic or semantic relations and it is not advisable to over-generalise them), and concepts generalise to "_" (that is, concepts that would unify with anything). The generalisation process applies to every concept except those that belong to a specific list of "stop concepts" (in analogy to the idea of stop words in Information Retrieval). The current list of stop concepts is:

and, *or*, *not*, *nor*, *if*, *otherwise*, *have*, *be*, *become*, *do*, *make*

The resulting generalised rules may then overgeneralise and therefore they must be weighted according to their ability to detect the correct answer in the training corpus. The weight $\mathcal{W}(r)$ of a rule r is computed following the formula:

$$\mathcal{W}(r) = \frac{\text{\# correct answers found}}{\text{\# answers found}}$$

5 Graph-based Question Answering

Given a question q with graph Q and a sentence s with graph S, the process to find the answer iterates over all the rules. A rule r triggers if the overlap component of the rule r_o is a subgraph of Q (which can be easily determined by checking that $ovl(r_o, Q) = r_o$). When that happens, the graph of the question is expanded with the rule path r_p, producing a new graph Q_{r_p}. The resulting graph is more likely to produce a large overlap with an answer sentence similar to the one that generated the rule and, most importantly, the graph contains an indication of where the answer is located.

Once the graph of the question has been expanded with the path, one only needs to compute the overlap between this expanded graph and that of the answer sentence $ovl(Q_{r_p}, S)$. If the overlap retains part of the exact answer that was marked up by the graph rule, then we have found a possible answer.

The above method will cover simple cases, but it needs to be extended to cover two special cases that arise from the fact that the question/sentence pairs that generated the rule are likely to be different from the actual question and sentence being tested. First of all, a question may trigger several rules, and each rule may extract a different answer from the answer sentence. And second, it is possible that the overlap between the expanded graph and the sentence does not contain the complete answer but part of it. We will explain how to handle these two cases below.

To identify the correct answer among a set of possible answers it is necessary to establish a measure of "answerhood" so that the correct answer has a higher score than the score of other candidates. The rule weight gives an indication of the quality of the answer extracted, but we also need to keep in account the similarity (or otherwise) between the text originating the rule and the text being tested. Given that the graph of the question has been expanded with the path linking the question and the exact answer determined in the training corpus, then the size of the overlap between the expanded graph of the test question and the graph of the test answer can be used as an estimation. Thus, the measure of answerhood $\mathcal{A}(pa)$ of a possible answer pa is the product of the weight of the rule used r, and the size of the best overlap between the graph of the question sentence expanded with the rule path and the graph S of the answer sentence:

$$\mathcal{A}(a) = \mathcal{W}(r) \times size(ovl(Q_{r_p}, S))$$

The size of a graph is computed as the weighted sum of all concepts and relations in the path. The formula to determine the weight of each concept and relation is inspired on the use of the Inverse Document Frequency (IDF) measure used in Document Retrieval. The actual formula that we use is:

$$\mathcal{W}_i = \frac{1}{\log N} \log \frac{N}{n}$$

```
35.1
When did Jack Welch become chairman of General Electric?
Jack Welch took over GE in <answ>1981</answ>.
Welch became GE's chief executive in April <answ>1981</answ>.
Welch was named chief executive in <answ>1981</answ>.

35.4
How many people did he fire from GE?
He sold off underperforming divisions and fired about <answ>100,000</answ> people.
More than <answ>100,000</answ> GE jobs have been axed under Welch.
```

Figure 5: Extract of the training corpus

n = total number of sentences using the concept (or relation) i

N = total number of sentences

The formula includes the constant factor $1/\log N$ to ensure that the values range between 0 and 1.

6 Evaluation and Results

We have conducted an initial evaluation of the use of these rules within the task of question answering. For this evaluation we created a training and testing corpus based on the first 111 questions of the question-answering track of TREC 2004 (Voorhees, 2004). For each question, we applied Ken Litkowsky's patterns[12] to automatically extract sentences in the AQUAINT corpus containing possible answers. These sentences were checked manually, and only sentences containing the answer and a justification were selected. As a result we obtained about 560 question/answer pairs. The exact answers in the answer sentences of the training corpus were manually marked up to ensure a corpus without wrong answers. Figure 5 shows an extract of the training corpus.

The question/answer training corpus was split in 5 sets, and a 5-fold cross-validation was performed. Table 1 shows the results. In this table, the accuracy indicates the percentage of questions that are answered correctly. The MRR measure is as used in the TREC evaluations (Voorhees, 2001), and it measures the mean reciprocal rank for ranks from 1 to 5. For example, if the correct answer was ranked 3 (i.e. the system ranked two wrong answers higher than the correct answer), then the reciprocal rank is 1/3. If the correct answer was ranked beyond 5, then the reciprocal rank is 0. The MRR is the mean of the reciprocal ranks across all questions. Given that the results indicate an MRR with value higher than the accuracy, we can deduce that the system sometimes finds the correct answer but it does not assign it the top rank.

	Accuracy	*MRR*
Test 1	25.76%	28.91%
Test 2	26.92%	31.09%
Test 3	9.21%	16.56%
Test 4	26.47%	29.78%
Test 5	18.82%	23.53%
Average	*21.44%*	*25.97%*

Table 1: Graph-based question answering results. The accuracy is the percentage of questions that are correctly answered. The MRR measure is as used in the TREC evaluations (see text).

Figure 6 shows the distribution of weights among the rules learnt (this is the sum of all the rules produced in all the five runs). To avoid computational overhead of handling rules with low weight we decided to set a threshold of 0.5. Any rules with weight below 0.5 were discarded. The figure indicates two clear regions, one with rules of weight lower than 0.6 and one with rules of weight above 0.9. It is probably desirable to set a threshold of 0.9 for a larger training corpus to ensure that only good quality rules are used. We refrained from doing so with our small corpus to avoid ending up with too few rules.

It is difficult to compare the above results with those of existing evaluations for various reasons. First of all, the system used in our evaluation did not have the usual modules of a full-blown QA system as described in Section 1. Second, the task is a simplification of a real QA

[12]Ken Litkowsky's patterns are available from the TREC website (http://trec.nist.gov).

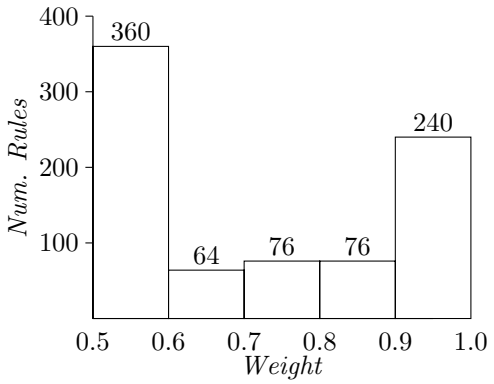


Figure 6: Distribution of weights among the rules learnt

task in that the answer is known to exist in the answer candidate. An area of further work is therefore to integrate this method into the AnswerFinder QA system and to evaluate the system in a task such as the QA track of TREC.

7 Related Work

There have been several attempts to automatically learn the correspondence between a question and an answer. For example, (Echihabi et al., 2004) describes three approaches to use question/answer rules, two of which use machine learning methods. In one approach, the system uses a methodical series of web searches containing a question phrase and the answer to collect a corpus of substrings linking the question phrase with the answer. The other machine-learning method described by Echichabi *et al.* uses techniques based on statistical machine translation to automatically learn the "translation" between a question and an answer.

A system that uses syntactic information in machine learning for QA has been recently published by Shen et al. (2005). This system is based on the extraction of dependency chains connecting a question word with an answer. The information is combined with other statistical features and fed to a Maximum Entropy model that ranks the answer candidates. The use of dependency chains in this system is similar in principle with our use of graph paths in that it provides a way to connect a question with its answer. We have not had time however to study this system in detail.

Another system that uses syntactic information to develop patterns is described by Bouma et al. (2005). Their system uses the output of a dependency parser combined with a set of equivalence rules between sets of dependency relations that paraphrase each other. In contrast with our method, however, the rules were developed manually and there were no indications in the paper about how to develop a method to automatically learn the rules. A method to discover similar sets of dependencies has been described by Lin and Pantel (2001), so in principle it is feasible to learn paraphrase rules and apply them to QA. However, the paraphrase rules described by these two systems do not attempt to connect a question with an answer, as we do.

The only system using logical-form rules that we are aware of is AnswerFinder at the time of participation in TREC 2004 (Mollá and Gardiner, 2004). The rules were based on AnswerFinder's minimal logical forms, and they were built manually. The system presented in our paper is a continuation of this research. Other than AnswerFinder, we are not aware of any QA system that attempts to learn rules based on logical forms.

There is some work on the use of conceptual graphs for information retrieval (Montes-y-Gómez et al., 2000; Mishne, 2004). However, we are not aware of any publication about the use of conceptual graphs (or any other form of graph representation) for question answering other than our own.

8 Conclusions and Further Work

We have introduced a methodology for the learning of graph patterns between questions and answers. Rules are learnt on the basis of two graph concepts: graph overlap, and paths between two subgraphs in a graph.

The techniques presented here use graph representations of the logical contents between questions and answer sentences. These techniques are being tested in AnswerFinder, a framework for the development of question-answering techniques that is easily configurable.

We believe that our method can generalise to any graph representation of questions and answer sentences. Further work will include the use of alternative graph representations, including the output of a dependency-based parser.

Finally, we plan to continue our evaluation of the method by integrating it into the AnswerFinder system and other QA systems to fully assess its potential.

9 Acknowledgements

This research is funded by the Australian Research Council, ARC Discovery Grant no DP0450750.

References

Gosse Bouma, Jori Mur, and Gertjan van Noord. 2005. Reasoning over dependency relations for qa. In *Proc. IJCAI-05 Workshop on Knowledge and Reasoning for Answering Questions*, pages 15–20.

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

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

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. 2000. Falcon: Boosting knowledge for answer engines. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC-9*, number 500-249 in NIST Special Publication, pages 479–488. NIST.

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

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Godehard Link. 1983. The logical analysis of plurals and mass terms: a lattice-theoretical approach. In Rainer Bauerle, Christoph Schwarze, and Arnim von Stechov, editors, *Meaning, Use and Interpretation of Language*, pages 250–209. de Gruyter, Berlin.

Gilad Mishne. 2004. Source code retrieval using conceptual graphs. Master's thesis, University of Amsterdam.

Dan Moldovan, Marius Pas ca, Sanda Harabagiu, and Mihai Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154.

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

Diego Mollá. 1997. *Aspectual Composition and Sentence Interpretation: a formal approach*. Ph.D. thesis, University of Edinburgh.

Manuel Montes-y-Gómez, Aurelio López-López, and Alexander Gelbukh. 2000. Information retrieval with conceptual graph matching. In *Proc. DEXA-2000*, number 1873 in Lecture Notes in Computer Science, pages 312–321. Springer-Verlag.

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

Fabio Rinaldi, James Dowdall, Kaarel Kaljurand, Michael Hess, and Diego Mollá. 2003. Exploiting paraphrases in a question answering system. In *Proc. Workshop in Paraphrasing at ACL2003*, Sapporo, Japan.

Gerold Schneider. 1998. A linguistic comparison of constituency, dependency and link grammar. Master's thesis, University of Zurich. Unpublished.

Dan Shen, Geert-Jan M. Kruijff, and Dietrich Klakow. 2005. Exploring syntactic relation patterns for question answering. In Robert Dale, Kam-Fai Wong, Jian Su, and Oi Yee Kwong, editors, *Natural Language Processing IJCNLP 2005: Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005. Proceedings*. Springer-Verlag.

M. M. Soubbotin. 2001. Patterns of potential answer expression as clues to the right answers. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC 2001*, number 500-250 in NIST Special Publication. NIST.

John F. Sowa. 1979. Semantics of conceptual graphs. In *Proc. ACL 1979*, pages 39–44.

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

Ellen M. Voorhees. 2001. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.

Ellen M. Voorhees. 2004. Overview of the trec 2004 question answering track. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proc. TREC 2004*, number 500-261 in NIST Special Publication. NIST.

A Statistical Approach towards Unknown Word Type Prediction for Deep Grammars

Yi Zhang and **Valia Kordoni**
Department of Computational Linguistics
Saarland University
Saarbrücken, Germany, D-66041

Abstract

This paper presents a statistical approach to unknown word type prediction for a deep HPSG grammar. Our motivation is to enhance robustness in deep processing. With a predictor which predicts lexical types for unknown words according to the context, new lexical entries can be generated on the fly. The predictor is a maximum entropy based classifier trained on a HPSG treebank. By exploring various feature templates and the feedback from parse disambiguation results, the predictor achieves precision over 60%. The models are general enough to be applied to other constraint-based grammar formalisms.

1 Introduction

Deep processing delivers fine-grained syntactic and semantic analyses which are desirable for advanced NLP applications. However, *specificity* and *robustness* are the major difficulties that deep processing has encountered for years.

Unlike shallow methods, which in most cases deliver an expected number of analyses, the number of output from deep processing is usually unpredictable, especially for open texts. The *specificity* problem arises when there are more analyses generated than expected. The analyses might be linguistically sound, but practically uninteresting for real applications. Recently, with more deep processing resources made available (Oepen et al., 2002), the specificity problem is being alleviated with statistical parse selection models (Toutanova et al., 2002).

As to *robustness*, more open questions remain to be investigated. A deep grammar is normally a complicated rule system. Whenever the input varies, even slightly, beyond the grammar developers' expectations, the output becomes unpredictable.

Closer studies of deep grammars have shown that lexicon coverage is one of the major barriers preventing deep grammars from being used for open text processing. Take the LinGO English Resource Grammar (ERG) (Copestake and Flickinger, 2000), for instance. The grammar has been developed for more than 10 years, and currently contains about 22K lexicon entries. A recent test on the BNC corpus reported that only 32% of the strings have full lexical span, of which 57% get at least one parse (Baldwin et al., 2004). About 40% of the parsing failures are due to lexicon missing. Lexicalized deep grammars rely on knowledge-rich lexicon. However, the construction of a lexicon with decent coverage requires a huge amount of human effort and considerable linguistic proficiency.

A widely adopted approach towards robust deep processing is to integrate shallow methods (Callmeier et al., 2004). However, most recent approaches still work on various fall-back strategies. When a deep processing component fails to deliver output, intermediate or shallow components are invoked to provide compatible analyses. Practically valid, this approach does not directly help to enhance the robustness of deep processing itself.

Inspired by the statistical approaches in parse selection (Toutanova et al., 2002), we propose a statistical approach for unknown word type prediction. The experiments are carried out on a broad-coverage linguistically-precise HPSG grammar for English, the LinGO English Resource Grammar (ERG) (Copestake and Flickinger, 2000). However the underlying statistical model is general enough to apply to other deep grammars. Also, by incorporating the parse disambiguation result, we show that the *robustness* is in nature a dual problem to the *specificity*. And they can benefit from each other's improvements.

The remainder of the paper is structured as follows: Section 2 gives the background about the lexicon in HPSG; Section 3 describes our statistical models for unknown word type prediction and the various feature templates we

Proceedings of the Australasian Language Technology Workshop 2005, pages 24–31,
Sydney, Australia, December 2005.

use; Section 4 shows how the parse selection model can be incorporated to enhance the precision of prediction ; Section 5 reports on the experiment results; Section 6 compares our approach to other related work; Section 7 concludes our approach and presents some aspects of our future work.

2 Lexicon Representation and Definitions in HPSG

Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) is a widely adopted constraint-based grammar formalism. Based on *typed feature structure (TFS)* (Carpenter, 1992), HPSG is highly lexicalized, which means there is only a limited number of highly generalized rules (ID Schemata & LP rules). A knowledge-rich lexicon is organized into a complex type hierarchy.

In HPSG, all the linguistic objects are modeled by TFSs. Formally, a TFS is a *directed acyclic graph (DAG)*. Each node in the DAG is labelled with a *sort symbol* (or *type*) corresponding to the category of the linguistic object. All the *sort symbols* are organized into an inheritance system, namely the *type hierarchy*. Two types are compatible if they share at least one common subtype in the hierarchy.

The lexicon is also organized into the type hierarchy. In principle, each lexical entry is a well-formed TFS, which conveys a set of constraints. The constraints include both feature-value appropriateness and type compatibility. For instance, Figure 1 is the TFS for the proper name *"Mary"*.

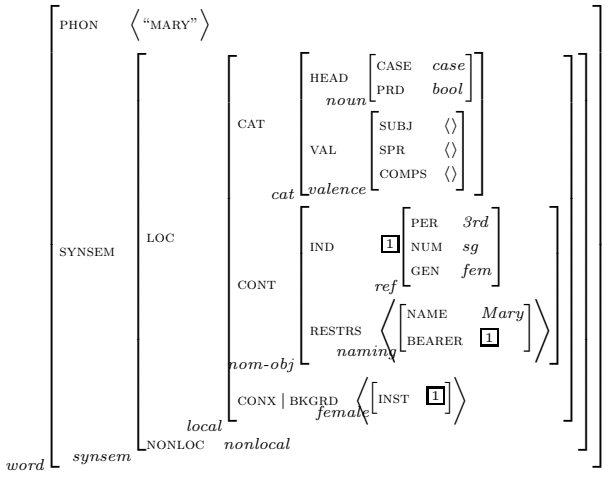


Figure 1: TFS of lexical entry for *"Mary"*

However in implementation, complete description is barely necessary. Most of the constraints will be conveyed via type inheritance. Only entry specific information like the stem and the semantic relation are required.

```
tsunami_n1 := n_intr_le &
  [ STEM < "tsunami" >,
    SYNSEM [ LKEYS.KEYREL.PRED
               "_tsunami_n_rel",
             PHON.ONSET con ] ].

admire_v1 := v_np_trans_le &
 [ STEM < "admire" >,
   SYNSEM [ LKEYS.KEYREL.PRED
              "_admire_v_rel",
            PHON.ONSET voc ] ].
```

Figure 2 gives part of the lexical hierarchy in ERG under the type *basic_noun_word*. Types with the suffix *"le"* are so-called leaf lexical types and should be directly assigned to lexical entries. These types are always mutually separated and incompatible. It is noticeable that each lexical entry takes exactly one leaf lexical type. When a word has more than one syntactic and/or semantic behaviors, different lexical entries will be created separately.

ERG[1] defines in total 741 leaf lexical types, of which 709 types are actually used in its lexicon with 12347 entries. A large number of these lexical types are closed categories whose lexical entries should already exist in the grammar. It is obvious that missing lexical entries, in most cases, should be in open categories. Verb, noun, adjective and adverb are the major open categories. In ERG, the number of leaf lexical types under these general categories are shown in Table 1.

General Cat.	Leaf Lex Types Num.
verb	261
noun	177
adjective	78
adverb	53

Table 1: Number of Leaf Lexical Types under Major Open Categories in ERG

However, even for the open categories, the distribution of existing lexical entries over different lexical types varies significantly. Table

[1]The June 2004 release of ERG was used throughout this paper for experiments and statistics. This was also the version used for building the latest version of Redwood Treebank.

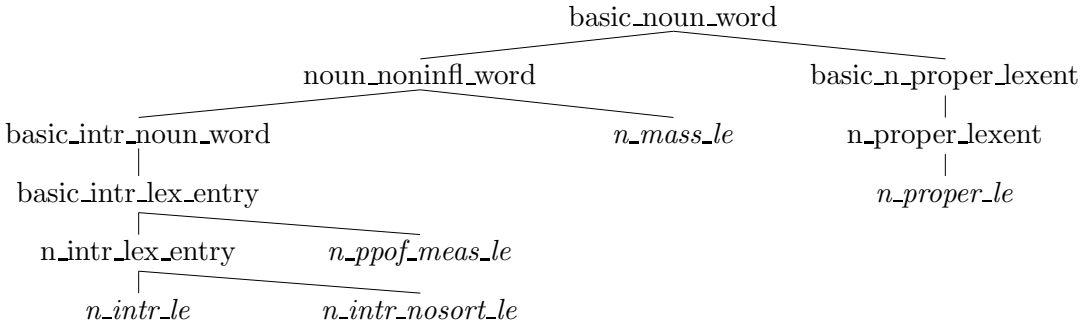


Figure 2: Part of the Lexical Hierarchy in ERG

2 lists the top 10 lexical types with maximum number of entries in the ERG lexicon.

Leaf Lexical Type	Num. of Entries
n_intr_le	1742
n_proper_le	1463
adj_intrans_le	1386
v_np_trans_le	732
n_ppof_le	728
adv_int_vp_le	390
*v_np*_trans_le*	342
n_mass_count_le	292
v_particle_np_le	242
n_mass_le	226

Table 2: Number of Entries for Top-10 Leaf Lexical Types in ERG

The top 10 verbal types count for about 75% of the verbal entries. For nouns the figure is about 95% and 90% for adjectives. Presumably, the automated lexical extension for nouns will be easier. This is plausible because verbal lexical entries normally require more detailed subcategorization information.

3 Statistical Unknown Word Type Prediction Models

For open text processing, a static lexicon inevitably becomes insufficient. A better strategy is to build an unknown word type predictor which can *"guess"* the lexical type from the available context, and generate lexical entries on the fly.

As mentioned in Section 2, the lexicon of an HPSG grammar is organized into a type hierarchy. Each entry bears exactly one leaf lexical type. So the predictor is actually a classifier, which takes various context and morphological forms of the unknown word into consideration, and picks out the most suitable leaf lexical type as output.

Such an unknown word type predictor is essentially very similar to a part-of-speech (POS) tagger. A typical POS tagger assigns a (unique or ambiguous) part-of-speech tag to each token in the input. A large number of current language processing systems use a POS tagger for pre-processing. The difference is that our unknown word type predictor has a very larger tagset. The tagset of a typical POS tagger usually contains tens of different tags. But our predictor needs to handle hundreds of possible types. In addition, an unknown word type predictor only predicts unknown words while a typical POS tagger generates tags for each token on the input sequence. Another point is that our unknown word type predictor can use any context information available at the processing stage. But normally a POS tagger only uses surface context features because these are usually used during pre-processing.

3.1 Maximum Entropy Classifier Based Predition Model

Considering these difference, we have constructed our predictor based on a maximum entropy classifier. The advantages of a Maximum entropy model lie in the general feature representation and in no independence assumptions between features. A maximum entropy model can also easily handle thousands of features and large numbers of possible outputs.

For our prediction model, the probability of a lexical type t given an unknown word and its context c is:

$$p(t,c) = \frac{exp(\sum_i \theta_i f_i(t,c))}{\sum_{t' \in T} exp(\sum_i \theta_i f_i(t',c))} \qquad (1)$$

where feature $f_i(t,c)$ may encode arbitrary

characteristics of the context. The parameters $< \theta_1, \theta_2, \ldots >$ can be evaluated by maximizing the pseudo-likelihood on a training corpus (see (Malouf, 2002)).

The basic feature templates used in our ME-based model include the prefix and suffix of the unknown word, the context words within a window size of 5, and their corresponding lexical types.

3.2 Using Partial Parsing Results as Features

Each lexical type is essentially a set of constraints on linguistic objects. If a word has a specific lexical type, it must conform to all the constraints demanded by the type, and hence it can only appear in some specific linguistic context. The constraints concern various linguistic aspects, among which syntactic constraints are predominant.

One advantage of using a maximum entropy based model is that ME allows the combination of diverse forms of contextual information in a principled manner, and it does not impose any distributional assumptions on the training data. So far, only the surface context features (words and their lexical types) are used. It can be presumed that the precision can be enhanced by adding syntactic context as features into the prediction model.

However, syntactic information is not available in a traditional pipeline processing model, where the syntactic analysis will be the post-processing module to the predictor. Also, when there are unknown words in the input, a full analysis of the sentence is not possible.

So we have modified our strategy by inserting a partial parsing stage before the lexical type predictor if there are unknown words on the input sequence.

The partial parse needs some clarification. A full parse can be represented by a set of edges as shown in Figure 3(a). Each edge is derived from a rule application. There is no more than one edge between each pair of positions. And there is always exactly one full span edge in a full parse.

A partial parse of an input sequence is a set of edges which composes a shortest path from the beginning to the end of the sequence[2]. There might be more than one partial parse for a given input sequence. As shown in Figure 3(b), when the word between position 2 and 3 is unknown, a dummy edge c is created. This dummy edge will prevent further rule application. Both $a - c - d$ and $b - c - d$ are partial parses.

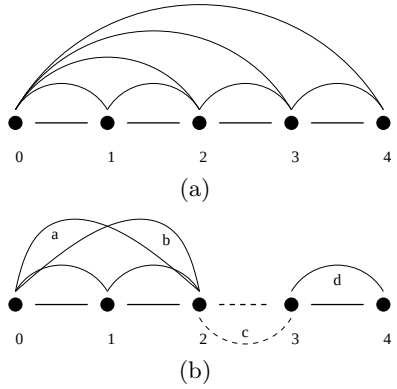


Figure 3: Parsing edges: (a) edges in a full parse; (b) edges in partial parses.

From the partial parses, we collect all edges that are adjacent to the left/right of the unknown word, respectively. Then the rules that generate these edges are counted according to their application (once per edge). The most frequently used rules to create left/right adjacent edges are added as two features conveying syntactic information into the ME-based model. A complete list of all features templates used in our predictor are listed in Table 3.

4 Incorporating Parse Disambiguation Results

As mentioned before, deep lexical types normally encode complicated constraints that only make sense when they work together with the grammar rules. And some subtle differences between lexical types do not show statistical significance in a corpus with limited size. So the feedback from later stages of deep processing is very important for predicting the lexical types for the unknown words.

The partial parsing results break the pipeline model. However, they might help only when the unknown is not the head of the phrase. Otherwise, the full parse crushes into small fragments, and the partial parsing results normally make no sense. An alternative way of breaking the pipeline model is to help the parser to generate full parses in the first place, and let the parsing result tell which lexical entry is good.

[2]Note that the edges on the full parse of the sentence are not necessary in the corresponding partial parses if a word is assumed to be unknown. However, partial parses do reduce the number of candidate edges for consideration.

Features
X is prefix of w_i, $\|X\| \leq 4$
X is suffix of w_i, $\|X\| \leq 4$
$t_{i-1} = X$, $t_{i-2}t_{i-1} = XY$, $t_{i+1} = X$, $t_{i+1}t_{i+2} = XY$
$w_{i-2} = X$, $w_{i-1} = X$, $w_{i+1} = X$, $w_{i+2} = X$
LP is the left adjacent most frequent edge of w_i
RP is the right adjacent most frequent edge of w_i

Table 3: Feature templates used in ME-based prediction model for word w_i (t_j is the lexical type of w_j)

In order to help the parser to generate a full parse of the sentence, we feed the newly generated lexical entries directly into the parser. Instead of generating only one entry for each occurrence of unknown, we pass on top n most likely lexical entries. With these new entries, the sentence will receive one or more parses (assuming the sentence is grammatical and covered by the grammar). From the parsing results, a best parse is selected with the disambiguation model, and the corresponding lexical entry is taken as the final result of lexical extension.

Within this processing model, the incorrect types will be ruled out if they are not compatible with the syntactic context. Also the infrequent readings of the unknown will be dispreferred by the disambiguation model.

5 Experiments

Missing lexical entries can be discovered by lexicon checking. Precision is the only measurement for the lexical type predictor. In this section we will evaluate our models by experiments.

5.1 Resources

Redwoods (Oepen et al., 2002) is a HPSG treebank that records full analyses of sentences with *ERG*. The genre of texts includes email correspondence, travel planning dialogs, etc. The 5th growth of *Redwoods* contains about 16.5K sentences and 122K tokens[3].

In all our experiments, we have done a 10-fold cross validation on the *Redwoods* treebank. For each fold, words that do not occur in the training partition are assumed to be unknown.

A modified version of the efficient HPSG parser *PET* (Callmeier, 2000; Callmeier, 2001) has been used to generate the derivation tree fragments of the partial parses.

We have also modified *LexDB* (Copestake et al., 2004) in order to be able to add temporal lexical entries that are only active for specific sentence.

The parse disambiguation model we have used is a maximum entropy based model that uses non-lexicalized features with 2 levels of grandparnets (see (Toutanova et al., 2002) for detailed discussion about parse disambiguation models for HPSG grammars).

For maximum entropy parameter estimation, we have used (Malouf, 2002)'s *MaxEnt* package.

5.2 Results

For comparison, we have built a baseline system that always assigns a majority type to each unknown according to the POS tag. More specificically, we tag the input sentence with a small Penn Treebank-like POS tagset. Then POS tag is mapped to a most popular lexical type for that POS.[4] Table 4 lists part of the mappings.

POS	Majority Lexical Type
noun	n_intr_le
verb	v_np_trans_le
adj.	adj_intrans_le
adv.	adv_int_vp_le

Table 4: Part of the POS tags to lexical types mapping

Again for comparison, we have built another two simple prediction models with two popular general-purpose POS taggers, *TnT* and *MXPOST*. *TnT* is a HMM-based trigram tagger while *MXPOST* is maximum entropy based. We have trained the tagging models by using all the leaf lexical types as the tagset. The taggers tag the whole sentence. But only the output tags for the unknowns are taken to generate the lexical entries.

[3]Sentences without a full analysis are neither counted here nor used in experiments.

[4]This is similar to the built-in unknown word handling mechanism of the *PET* system.

The maximum entropy based model is tested both with and without using partial parsing results as features. To incorporate disambiguation results, our predictor generates 3 entries for each unknown and store them as temporary entries into the *LexDB*.

Precisions of the different prediction models are shown in Table 5.

Model	Precision
Baseline	30.7%
TnT	40.4%
MXPOST	40.2%
ME(-pp)	50.0%
ME(+pp)	50.5%
ME(-pp)+ disambi. result	61.3%

Table 5: Precision of Unknown Word Type Predictors (+/-pp means w or w/o partial parsing result features)

The baseline model achieves precision around 30%. This means that the task of unknown word type prediction for deep grammars is non-trivial. The general-purpose POS taggers based models perform quite well, outperforming the baseline by 10%. As a confirmation to (Elworthy, 1995)'s claim, a huge tagset does not imply that tagging will be very difficult. Our ME-based model significantly outperforms the taggers-based models by another 10%. This is a strong indication of our model's advantages.

By incorporating simple syntactic information into the ME-based model, we get extra precision gain of less than 1%. It is worth noticing that the syntactic features we used are still naive. Better syntactic features remain to be explored in future work. Also, by applying partial parsing, the computation complexity increases significantly in comparison to our basic ME-based model.

By incorporating the disambiguation results, the precision of the model boosts up for another 10%. The computational overhead is proportional to the number of candidate entries added for each unknown word. However, in most cases, introducing lexical entries with incorrect types will end up to parsing failure and can be efficiently detected by quick checking. In such cases the slowdown is acceptable.

In general, we have achieved up to 60% precision of unknown word type prediction for the ERG in these experiments. Given the complexity of the grammar and the huge number of possible lexical types, these results are satisfying. Also, in real case of grammar adaptation for new domains, a large portion of unknowns are proper names. This means that the precision might get even higher in real applications. A test with some small text collection with real unknown words [5] shows that the precision can easily go above 80% with the basic ME model without partial parsing features.

It should also be mentioned that some of these experiments are also carried out for Dutch Alpino Grammar (Bouma et al., 2001), and similar results are obtained. This shows that our method may be grammar and platform independent.

6 Comparison with Related Work

This work is in essence very similar to the work of deep lexical acquisition (DLA) in (Baldwin, 2005). A minor difference is that our model always generates (at least) one lexical entry for the unknown, so that the deep processing does not halt at the very beginning. A more important difference is that, while (Baldwin, 2005) focuses on generalizing the method of deriving DLA models on various secondary language resources, our work focuses more on how to utilize the deep grammar itself as a source for enhancing robustness. The *Redwoods Treebank* is by nature the output of the deep grammar. And the parsing, as well as the disambiguation models are also part of the grammar that has eventually contributed to the unknown word type prediction.

(Erbach, 1990; Barg and Walther, 1998; Fouvry, 2003) followed a different approach towards unknown words processing for unification based grammars. The basic idea was to use the underspecified lexical entries, namely TFSs with fewer constraints, in order to generate full parses for the sentences, and then extract the sub-TFS from the parses as a new lexical entry. However, lexical entries generated in this way might be both too general and too specific. And underspecified lexical entries with fewer constraints allow more grammar rules to be applied while parsing. It gets even worse when

[5] We used a text set named *rondane* for training and *hike* for testing. *rondane* contains 1424 sentences in formal written English about tourism in the norwegian mountain area, with an average sentence length of 16 words; *hike* contains 320 sentences about outdoor hiking in Norway with an average sentence length of 14.3 words. Both contain a lot of unknowns like location names, transliterations, etc.

two unknown words occur next to each other, which might allow almost any constituent to be constructed. Also, the underspecified lexical entry significantly increases computational complexity. (van Schagen and Knott, 2004) took a similar approach of interactive unknown word acquisition in a dialogue context.

(Thede and Harper, 1997) reported an empirical approach towards unknown lexical analysis using morphological and syntactic information. The approach is similar to ours in spirit. However, the experiments were done for a shallow parser with a very limited number of word classes. The applicability to lexicalist deep grammars with lots of lexical types is unknown.

In (Malouf and van Noord, 2004), the maximum entropy models were used for wide coverage parsing with the *Alpino* Dutch grammar (Bouma et al., 2001). But the focus was on parse selection, not unknown words processing.

Another related work is *supertagging* (Bangalore and Joshi, 1999). In supertagging, the lexical items are assigned with rich descriptions (supertags) that impose complex constraints in a local context. Some statistical techniques of assigning supertags to unknown words have been reported. For example, (Bangalore and Joshi, 1999) used a simple method of combining a probability estimate for unknown words $P(UKN|T_i)$ with a probability estimate based on word features (capitalization, hyphenation, ending of words) by:

$$P(W_i|T_i) = P(UNK|T_i) * P(w\_feat(W_i)|T_i) \quad (2)$$

where UNK is a token associated with each supertag and its count N_{UNK} is estimated by:

$$P(UNK|T_j) = \frac{N_1(T_j)}{N(T_j)+\eta} \quad (3)$$

$$N_{unk}(T_j) = \frac{P(UNK|T_j) * N(T_j)}{1 - P(UNK|T_j)} \quad (4)$$

$N_1(T_j)$ is the number of words that are associated with the supertag T_j that appear in the corpus once. From some aspect, this approach is similar to our work. But our ME-based model allows more general feature representation. Also the lexical types we used are more general in the sense that both local and non-local constraints are encoded.

7 Conclusion and Future Work

Several statistical unknown word type prediction models are implemented and evaluated for deep HPSG grammars. The general-purpose POS taggers based approach delivers satisfying precision. The maximum entropy based predictor allows for more general feature representation. By incorporating parse disambiguation results, the unknown word type predictor achieves precision over 60%.

Although the experiments are carried out with the ERG, the underlying model is general enough to be easily applied on other constraint-based lexicalist grammars, provided the lexical categories can be abstracted by a set of atomic types.

Several aspects of this work need further exploration. More sophisticated syntactic features should be investigated. Besides, the deep grammar also provides semantic analyses which are not available in shallow processing. The general feature representation in our model allows the incorporation of this orthogonal dimension of information to enhance the precision of prediction. Also, larger corpora in more variety of genres are certain to generate better models. The application of the method to more deep grammars is anticipated.

References

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Timothy Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, Michigan, June. Association for Computational Linguistics.

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

Petra Barg and Markus Walther. 1998. Processing unkonwn words in HPSG. In *Proceedings of the 36th Conference of the ACL and the 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of dutch. In *Computational Linguistics in The Netherlands 2000*.

Ulrich Callmeier, Andreas Eisele, Ulrich

Schäfer, and Melanie Siegel. 2004. The deepthought core architecture framework. In *Proceedings of LREC 04*, Lisbon, Portugal.

Ulrich Callmeier. 2000. PET – a platform for experimentation with efficient HPSG processing techniques. *Journal of Natural Language Engineering*, 6(1):99–108.

Ulrich Callmeier. 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, England.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage english grammar using hpsg. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Ann Copestake, Fabre Lambeau, Benjamin Waldron, Francis Bond, Dan Flickinger, and Stephan Oepen. 2004. A lexicon module for a grammar development environment. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal.

David Elworthy. 1995. Tagset design and inflected languages. In *EACL SIGDAT workshop "From Texts to Tags: Issues in Multilingual Language Analysis"*, pages 1–10, Dublin, Ireland, April.

Gregor Erbach. 1990. Syntactic processing of unknown words. IWBS Report 131, IBM, Stuttgart.

Frederik Fouvry. 2003. Lexicon acquisition with a large-coverage unification-based grammar. In *Companion to the 10th of EACL*, pages 87–90, ACL, Budapest, Hungary.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conferencde on Natural Language Learning (CoNLL-2002)*, pages 49–55.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, Taipei.

Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois.

Scott M. Thede and Mary Harper. 1997. Analysis of unknown lexical items using morphological and syntactic information with the timit corpus. In *Proceedings of the Fifth Workshop on Very Large Corpora*, pages 261–272.

Kristina Toutanova, Christoper D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse ranking for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263, Sozopol, Bulgaria.

Maarten van Schagen and Alistair Knott. 2004. Tauira: A tool for acquiring unknown words in a dialogue context. In *Proceedings of the 2004 Australasian Language Technology Workshop (ALTW2004)*, Macquarie University, Australia.

Tagging Unknown Words with Raw Text Features

David Vadas and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dvadas1,james}@it.usyd.edu.au

Abstract

Processing unknown words is disproportionately important because of their high information content. It is crucial in domains with specialist vocabularies where relevant training material is scarce, for example: biological text. Unknown word processing often begins with Part of Speech (POS) tagging, where accuracy is typically 10% worse than on known words.

We demonstrate that features extracted from large raw text corpora can significantly increase accuracy on unknown words. These features supply a large part of what we are missing with unknown words: context information about how the word is used. We describe a Maximum Entropy modelling approach which uses *real-valued features* to represent unannotated contextual information. Our initial experiments with real-valued features have resulted in an increased accuracy from 87.39% to 88.85% on unknown words.

1 Introduction

Part of Speech (POS) tagging involves assigning basic grammatical classes such as verb, noun and adjective to individual words, and is a fundamental step in many Natural Language Processing (NLP) tasks. The tags it assigns are used in other processing tasks such as chunking and parsing, as well as in more complex systems for question answering and automatic summarisation.

All POS taggers suffer a significant decrease in accuracy on *unknown words*, that is, words that have not been previously seen in the annotated training set. A loss of up to 10% is typical for most POS taggers e.g. Brill (1994) and Ratnaparkhi (1996). This decreased accuracy has a flow on effect for the accuracy of both following POS tags and later processes which utilise them.

Unknown words also occur a significant amount of the time, ranging from 2% – 5% (Mikheev, 1997), depending on the training and test corpus. These figures are much higher for domains with large specialist vocabularies, for example biological text.

We improve the performance of a Maximum Entropy POS tagger by implementing features with non-negative real values. Although Maximum Entropy is typically described assuming binary-valued features, they are not in fact required to be binary valued. The only limitations come from the optimisation algorithm. For example, the Generalised Iterative Scaling (Darroch and Ratcliff, 1972) algorithm used in these experiments imposes a non-negativity constraint on feature values.

Real-valued features can encapsulate contextual information extracted from around unknown word occurrences in an unannotated corpus. Using a large corpus is important because this increases the reliability of the real-values. By looking at the surrounding words, we can formulate constraints on what POS tag(s) could be assigned. This can be seen in the sentence below:

(1) *The **frub** house is up on the hill*

Here, **frub** is the unknown word which, as competent speakers of the language, we can surmise is probably a noun or adjective. This is because it sits between a determiner and a noun, which is a position frequently assumed by words with these syntactic categories. Also, if we can find the word **frub** in other places, then we can get an even better, more reliable idea of what its correct tag should be.

It is not necessary to know the correct POS tags for *the* and *house*. We can determine from the words themselves that **frub** is occupying a position similar to adjectives like *big* or nouns like *club*.

The fact that *the* precedes our unknown word tells us a lot by itself, as this is a very common word. Since we see it so often, we know the types of words that follow it quite well. Other words that occur less frequently don't give as strong an indication of what is to follow, simply because the evidence is sparser.

Our aim then, is to take this intuitive reasoning for determining the correct tag for an unknown word, and create features that aid the Maximum Entropy model in doing the same.

Proceedings of the Australasian Language Technology Workshop 2005, pages 32–39,
Sydney, Australia, December 2005.

2 Unknown word processing

POS taggers have reached such a high degree of accuracy that there remain few areas where performance can be improved significantly. Unknown words are one of these areas, with state-of-the-art accuracy in the range 85 – 88%, which is well below the ∼97% accuracy achievable over all words.

The prevalence of unknown words is also problematic, although somewhat dependant on the size and type of corpus being used. We train on sections 0–18 of the Penn Treebank (Marcus et al., 1993), and test on sections 22–24. This test set then contains 2.81% (approximately 4000) unknown words. Also, when applying a POS tagger to a specialised area of text, such as technical papers, the number of unknown words and their frequency would be expected to increase dramatically, due to specific jargon terms being used.

Unknown words are also more likely to carry a greater semantic significance than known words in a sentence. That is, they will often contain a larger amount of the content of the sentence than other words. This is because unknown words are unlikely to be from closed-class categories such as determiners and prepositions, but quite likely to be in open-class categories such as nouns and verbs. It is these classes that generally convey most of the information in a sentence. Further, rarer words often have a more specialised meaning, and thereby classifying them incorrectly will potentially lose a lot of information. For these reasons, it is quite important that unknown words are POS tagged correctly, so that the information carried by them can be extracted properly in future stages of an NLP system.

Previous work on tagging unknown words has focused on morphological features, and using common affixes to better identify the correct tag. This has been done using manually created, common English endings (Weischedel et al., 1993), with Transformation Based Learning (TBL) (Brill, 1994), and by comparing pairs of words in a lexicon for differences in their beginnings and endings (Mikheev, 1997). Our existing tagger (Curran and Clark, 2003) already makes use of such features, while we aim to incorporate additional sources of information from a larger unannotated corpus.

3 Maximum Entropy modelling

A Maximum Entropy model is defined in terms of a number of *constraints* on the expected occurrences of *features* that represent the training data. Once these constraints on the model are met, the model assumes nothing further, giving a uniform distribution to all unknowns, that is, the model with *maximum entropy* (Ratnaparkhi, 1996). In this way, the model makes use of all the information available, but does not favour any further unfounded hypothesis, giving equal chance to all possibilities (Berger et al., 1996).

The empirical expectation of these features, as observed in the training data, is calculated by:

$$\tilde{p}(f) \equiv \sum_{x,y} \tilde{p}(x,y) f(x,y) \tag{1}$$

We attempt to make our model's estimated value:

$$p(f) \equiv \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) \tag{2}$$

an accurate reflection of the training data, so that,

$$p(f) = \tilde{p}(f) \tag{3}$$

and therefore,

$$\sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) = \sum_{x,y} \tilde{p}(x,y) f(x,y) \tag{4}$$

The training algorithm we use to achieve this is Generalised Iterative Scaling (GIS) (Darroch and Ratcliff, 1972). Each iteration of the algorithm involves updating all λ_i as follows:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{1}{C} \log \frac{\tilde{p}(f)}{p^{(t)}(f)} \tag{5}$$

where C is the maximum of the sum of the feature functions over all instances, and $\tilde{p}(f)$ and $p^{(t)}(f)$ are the expectations of the probabilities observed in the training data, and the probabilities in the current model respectively. If $\tilde{p}(f)$ is greater than $p^{(t)}(f)$, then the log of the ratio will be positive and λ_i will be increased. This will in turn increase $p^{(t+1)}(f)$, and move towards convergence and equality between the two probability expectations. Conversely, if $\tilde{p}(f)$ is less than $p^{(t)}(f)$, then λ_i will be decreased, again bringing the two expectations more into line.

We also use a Gaussian prior (Chen and Rosenfeld, 1999) which prefers weights close to a normal distribution. This form of smoothing alters the function we are attempting to maximise, so that no feature receives an inordinately high or low weight. Our code is based on the C&C tagger. (Curran and Clark, 2003).

4 Real-valued features

Maximum Entropy models have always been defined in terms of *binary* features of the form:

$$f(x,y) = \begin{cases} 1 & \text{if } x \text{ and } y = \texttt{class} \\ 0 & \textit{otherwise} \end{cases} \tag{6}$$

The fact that these are binary features, implies certain limitations of the representation, which make them unsuitable for some attributes. For example, the length of the word cannot be represented easily, as this value could range from one to ten or more, rather than being either present or absent.

Discretizing (or binning) the feature value is the easiest way to get around this constraint. For example, one scheme for encoding the length of the word would involve bins of length 1-3, 3-6, 6-9, and 10 or more. Then each word would have a particular feature present, depending on which bin they fitted into. However, it may be hard to find a discretization scheme that performs optimally.

Another problem with binary features that discretization fails to solve, is that they are unable to capture the fact that certain values are related. The values 1 and 50 would seem just as close as 2 and 3. Even worse, the model would be unable to generalise further, to say that 4 is between 2 and 7.

A better representation can be found using *real-valued features*, such as in the example below:

$$f(x, y) = p(\text{punctuation}|x) \text{ and } y = \text{class} \quad (7)$$

Here, $f(x, y)$ can take on any value between 0 and 1, inclusive, allowing a more continuous representation. Such features are commonplace when using other machine learners, but MaxEnt has, in most previous implementations, always been restricted to using binary features. This means that a large number of features are required for even simple pieces of information. For example, rather that having a single feature for the current word, there will instead be one feature for each word, with only the one for the current word turned on. MaxEnt classifiers are certainly capable of working in this manner, but real-valued features are able to do much more.

Implementing real-valued features adds an extra layer of complexity on top of binary features. With the latter, one only needs to know the features that are on for each training instance, since their values will always be one. For real-valued features however, we also need to know the particular value the feature holds in this instance. This is because the same real-valued feature will probably have different values in each instance.

5 Probabilistic contexts

The Associated Press section of the Aquaint corpus (Graff, 2002), containing over 100 million words, was used to calculate the probabilities for the real-valued features described below. Using this corpus gives us over a hundred times more words than the Penn Treebank across a wider range of topics.

FEATURE	UNKNOWN
original	87.39
`pw the`	87.39
`nw comma`	87.59
`pw was`	87.37
`pw determiner`	87.42
`nw punctuation`	87.70

Table 1: Results for Probabilistic Context Features

FEATURE	UNKNOWN	OVERALL
5 buckets, `pw the`	87.20	96.97
5 buckets, `nw comma`	87.34	96.98
10 buckets, `pw the`	86.76	96.94
10 buckets, `nw comma`	87.15	96.95
10 buckets, `pw was`	87.34	96.96

Table 2: Results using binned features

To begin with, we chose a simple feature, that we can intuitively see should help discriminate between classes. This feature is `previous word (pw) the`, that is, a real-valued feature describing the proportion of times in which we see *the* before the unknown word we are trying to find information on, compared to how many times we see the word at all. The idea of this feature is to tell when the unknown word we are looking at is a noun. This is because the determiner-noun pair is very common, and *the* is the most frequent determiner.

More potential features were identified by looking at the most common errors made by the tagger. We found that the most difficult distinction to make is between adjectives and common nouns. It is obvious that a feature with a better chance of increasing performance would solve the most common errors, and from this analysis it should therefore differentiate chiefly between nouns and adjectives. The feature we found that satisfies this property, is `next word (nw) comma`, that is, the proportion of times that the word we are focusing on is followed by a comma. Adjectives are only very infrequently followed by a comma, while nouns are in such a position quite often.

Another feature that was also tried, again in an attempt to discriminate between the most problematic cases, was `pw was`. This feature came from the idea that adjectives will often follow the word *was*, but nouns will not. The results from using each of these three features is shown in Table 1.

As a comparison, we also experimented with using traditional binary features. We used the binning technique mentioned in the previous section, with

five bins (divided as: 0, 0–1/3, 1/3–2/3, 2/3–1 and 1), and also ten equally spaced bins (0–0.1, 0.1–0.2, etc). As can be seen in Table 2, representing the information in this way actually causes a decrease in accuracy. This demonstrates that real-valued features can give an improvement that traditional binary features are unable to reproduce.

5.1 Higher frequency contexts

One reason why the `pw was` feature may not be as effective as `nw comma`, is that the word *was* does not occur as frequently as commas do. Commas (and *the*) occur about five times as often as *was*. The less frequent a word is, the less reliable it will be as a feature in the way we are using it. The reason for this it that it reduces the statistical reliability of the counts we are using as input for the feature. The counts overall will be lower, which increases the chance that one unusual context will be seen more than it should be, compared to the context in which the word is normally seen. Also, there will be less chance of finding counts greater than zero. There may be many adjectives that are used a number of times in the unannotated corpus, but still do not occur after the word *was*.

We next attempted to increase the counts extracted from the unannotated corpus. Seeing as the idea of the `pw the` feature was to identify nouns seen in a determiner-noun pair, one can expand the feature to become: `pw determiner`. Although the probabilities for what tags will follow a determiner are actually very close to those for the word *the*, the number of determiners in the unannotated corpus is significantly larger than just the word *the*.

We can also apply this idea to our `nw comma` feature, by expanding it into a `nw punctuation` feature. The results of experiments run with these new expanded features are also shown in Table 1. One can see that they were indeed more effective than the simpler features they replaced.

5.2 Contexts from the Penn Treebank

In order to find new features, we are looking for words that occur frequently and consistently have a certain tag before or after them. A list of tokens that meet these criteria can easily be found for each tag, by analysing the Penn Treebank. The results of this analysis are shown in Table 3.

This table shows the most frequent tokens following nouns, and the percentage of the time that they follow a noun. The first entry, which contains a feature we have already found to be useful, shows us that this analysis could give us more effective features. Some words however, *million* for example, are overly optimistic due to source of this data. The Wall Street Journal will clearly contain many such words whose frequencies are not representative of what they should be in more general text.

TOKEN	#	TAG %	TOKEN	#	TAG %
,	11541	81	million	1755	99
.	7894	78	on	1368	55
and	4856	70	was	1340	74
of	4171	87	said	1323	69
in	3311	63	has	1211	69
is	2260	70	are	1196	73
's	2173	95	will	1178	74
for	2029	64	from	1177	63

Table 3: Most frequent tokens that follow nouns

TOKEN	#	TAG %	TOKEN	#	TAG %
than	1089	54	venture	82	52
quarter	717	75	ones	61	77
week	433	52	transactions	59	51
income	392	71	term	59	55
period	240	67	factors	52	63
estate	213	93	spring	50	52
German	99	62	subordinated	48	62
thing	91	58	ventures	42	67

Table 4: Most frequent tokens that follow adjectives

FEATURE	UNKNOWN
original	87.39
`nw of`	87.42
`nw for`	87.50
`nw preposition`	87.56
`nw to be verbs`	87.48
`nw modals`	87.42
`nw and`	87.42
`pw possessive pronouns`	87.42
`pw adjectives`	87.45
`pw adverbs`	87.42

Table 5: Results using analysis of the Treebank

We can compare these words to the most common words that follow adjectives, shown in Table 4. The obvious differences visible here mean that the features we apply from this analysis should discriminate between nouns and adjectives well. Using this same technique for different classes, looking at previous word and next, we are able to find a number of features, each of which supplies a minor increase in performance. These are shown in Table 5.

6 Probabilistic lowercase features

Having explored features for better differentiation between nouns and adjectives, we now move onto features for improving some other grammatical cat-

1ST WORD	2 FEATURES	UNKNOWN	OVERALL
No	No	87.83	97.01
No	Yes	88.52	97.05
Yes	No	87.86	97.01
Yes	Yes	88.49	97.04

Table 6: Lower case frequency feature results

egories. One common problem is trying to determine whether a word is a proper noun or a common noun. This becomes particularly difficult when the best way of telling between these two classes: looking at capitalisation of the first letter, is not informative. This can be the case when the word is the first in the sentence or where the whole sentence is in upper case. To solve this problem, we can use the information from an unannotated corpus. If we see a unknown word and it is capitalised, then we can check the unannotated corpus for it, seeing whether it ever occurs uncapitalised. If it does, then there is a good chance that it is not a proper noun at all, but has been capitalised in this case for another reason.

We will represent this information in the model using a real-valued feature. It will be the proportion of times in which the word occurs in the unannotated corpus in lower case, compared to the number of times it occurs at all.

We performed experiments firstly with the lower case frequency of all words, and then using only instances of a word that weren't beginning a sentence. This is because a word beginning a sentence is always capitalised in English, whether it is a proper noun or not, which is precisely the bias we were trying to avoid. We also tried using two features, with one active for the first word in a sentence, while the other was active for all other words. This is because it is words that are first in the sentence that we are specifically trying to correct errors with. With only one feature, it would not get a strong enough weight, since the information it is giving on words that are not first in the sentence is not very useful. Having two features however, frees them both to make better decisions more specifically on the incorrect cases we are trying to fix.

As can be seen from the results in Table 6, the increase in performance was significantly more than any that had been attained previously.

7 Probabilistic plural features

Next we attempt to solve the third most common error that the model makes: incorrectly tagging singular proper nouns as plural proper nouns. The idea we can use to distinguish between them, is that if we find the unknown word in our unannotated corpus with *-s* on the end, then we can assume it is plural in this case, and that it is singular where we are trying to tag it. Although not all plurals are created by the *-s* ending, it is the case in the vast majority of times, and so we believe it will be effective enough. Implementing this as a feature, we attained an increase in unknown word accuracy to 87.61%.

This increase is greater than what most other features have achieved. It is interesting to note that most of the errors of tagging singular proper nouns as plural proper nouns consist of words such as *American Airlines*. Here, *Airlines* should be tagged as singular, but it still has the *-s* ending, so we would not expect our feature to help at all. An analysis of the errors fixed by this new feature shows that it actually helped to distinguish between nouns and adjectives. This can be understood, as if the unknown word in the test corpus appears with an *-s* on the end in our unannotated corpus, it suggests that it is a noun in both cases. Adjectives are not affected by the addition of an *-s* in any normal grammatical way, and so we would expect the chance of seeing such a construction to be much less.

8 Using multiple real-valued features

All the experiments we have described so far have included only one feature at a time (with the exception of lower case frequency, which uses two). Of course, we can use all of them at once, collecting information from all of them.

We also tried combining the features in a different way. Rather than having one feature for each piece of information we are giving the model, we could add up the counts used for each feature, creating one feature with larger, and hopefully more reliable counts than any of those that it is made up of. We then have one feature in the model for all these pieces of information. The value for this feature could exceed one, and so we experimented with limiting the value at one, or simply allowing it to take on whatever value it would.

The features that we used included all those that had individually given a positive result. The `lower case frequency` feature was also used, as was `word exists with -s`, although these two were not summed with the other features, as they are of a different nature. The results achieved, are shown in Table 7.

9 Analysis

Only about 3% of the words in the test set are unknown words, while both the test and training sets are made up of very similar text drawn from the Wall Street Journal. What we would like, is to be

COMBINATION METHOD	UNKNOWN	OVERALL
original	87.39	96.99
all positive	88.82	97.05
all positive, summed	88.85	97.05
all positive, summed with a limit of 1	88.82	97.05

Table 7: Accuracies from combining all positive features

	UNKNOWN WORD %	ORIGINAL		`nw punctuation`		`lower case freq.`		ALL POSITIVE	
		UNK.	ALL	UNK.	ALL	UNK.	ALL	UNK.	ALL
1	2.81	87.39	96.99	87.70	97.00	88.52	97.05	88.82	97.05
1/2	4.15	86.49	96.61	86.63	96.62	87.89	96.68	88.35	96.71
1/4	6.08	86.22	96.28	86.65	96.31	87.42	96.37	87.97	96.44
1/8	8.56	85.67	95.53	85.94	95.56	86.60	95.64	87.34	95.81
1/16	11.99	84.90	94.73	84.95	94.76	86.02	94.91	86.65	95.18

Table 8: Results with reduced training data

able to test our new features on a corpus from a different domain, since this is the kind of application where our features should perform the best. Unfortunately, there is no such corpus that is substantial enough for our purposes, which has the annotations that we would need to measure accuracy. As a result of this limitation, we will instead rely on a different technique to simulate tagging a piece of text with more unknown words.

9.1 Reducing the training dataset

We can reduce the amount of data used for training, thereby increasing the proportion of unknown words. This will also mean the tagger has less idea of the way words are used in general, which is another effect that we would like to mimic. Although the words themselves will still be drawn from the same financial newspaper text, and therefore exhibit all the same grammatical tendencies, we can still get an idea of how effective our new features are in a situation they are actually well-suited for.

Experiments were performed with a half, a quarter, an eighth, and a sixteenth of the training data, with the results shown in Table 8. The precentage of unknown words grows exponentially as the size of the training data decreases. This means that as we move to smaller training sets, unknown words rapidly become a more significant problem.

As would be expected, using a smaller training set reduces the performance of the tagger quite considerably. However, the improvements gained from using the new features remain consistent.

Even more encouraging is the way the overall results begin to improve. This stems from the fact that the unknown words are beginning to take up a more significant proportion of the text. This demonstrates that our new features are indeed able to raise accuracy in the area that they were designed to.

9.2 Cross Validation Experiments

We have also carried out an experiment using 10-fold cross validation. We split the entire Penn Treebank, putting every nth line into the nth fold. This configuration results in a similar percentage of unknown words as in previous experiments. The accuracy for the original system are 97.12% overall and 89.13% on unknown words only. Using the summation of all positive features, as well as the best `lower case frequency` feature and the `word exists with -s` feature, we achieved 97.15% overall and 90.17% on unknown words. This is an accuracy increase of 1.04% on unknown words which confirms the statistical validity of the results we have attained previously.

10 Named Entity Recognition

We have also experimented with the task of Named Entity Recognition (NER) in almost the same manner as we have described for POS tagging. We will use contextual information from the same unannotated corpus, in order to better classify a certain subset of the test corpus. In this case, the named entities themselves, while for POS tagging we looked at the unknown words.

The data we used is from MUC7 (DARPA, 1998), and uses seven entity categories: persons, organisations, locations, dates, money amounts, percentages, and times. 11% of the 84046 word-long corpus is made up of these entities, which is significantly more than the percentage of unknown words

FEATURE	PRECISION	RECALL	F-SCORE
original	86.55	86.78	86.66
nw preposition	86.64	86.87	86.75
pw preposition	86.69	86.69	86.69
pw locative preposition	86.79	86.87	86.83
pw said	86.64	86.87	86.75
pw speech marks	86.55	86.78	86.66

Table 9: NER results

in our POS tagging corpus. We performed an analysis on this dataset, as we did for POS tagging, attempting to find words that could be used as suitable features. Some of those we found are shown in Table 9, as well as the increases in accuracy that come from using them. We also experimented with the features that performed well in POS tagging, and found that they did not work as well at this task.

We can see that use of the `pw locative preposition` feature provided the greatest increase in accuracy. This feature was specifically used to identify locations, as they are often preceeded by words such as *to*, *from*, and *in*.

11 Discussion

What we have seen is that `nw punctuation` performs the best out of the next word/previous word features, while `word exists with -s` is also useful, outperforming many of them. Lower case frequency is quite easily our best feature, as it corrects so many of the errors that occur with proper nouns. Using all our features together, we find that even the minor increases gained by some of the individual features can be translated to an overall gain.

Our second technique for combining multiple features, that of summing the feature values together, has also been shown to give satisfactory results. The effect of having larger, more reliable counts was apparently able to compensate for the conflation of some information upon amalgamation.

All of our experiments and analysis have suggested two measurable quantities to identify a good feature. The first of these is to discriminate well between two possible classes, especially those classes that exhibit a high level of confusion. Here, `pw the` was not able to differentiate between noun and adjectives, while `nw comma` was. The second measure is the frequency of the word we are basing our feature upon, with more common words being better. For example, the `pw was` feature did not perform nearly as well as `nw comma`.

We used these ideas to find a number of productive features. Problem cases with the existing tagger gave us particular areas for improvement, such as the noun-adjective discrepancies. Further analysis of common word/tag combinations in the annotated corpus also gave us a number of worthwhile features. However, it should be noted that we did require the annotations for this analysis, as well as to implement some other features, such as `pw unambiguous adjective`. Seeing as a large annotated corpus, the Penn Treebank, is only available for English, we would have a problem shifting our focus to POS tagging in another language.

One thing that we have achieved is the implementation and use of real-valued features. They were able to outperform traditional MaxEnt features, finding an increase in accuracy where discretized features could not. In particular, one would not be able to represent the features that we are using in the Maximum Entropy model, and still gain an increase in performance, without using real-valued features. One can easily see more possibilities for their use, as they are a much more natural way of representing many attributes in a machine learner.

12 Future Work

The real-valued features we have described are specifically designed at better classify unknown words. We would therefore like to test our approach in a domain where unknown words are more prevalent. We would also like to try using a different, larger unannotated corpus, even when working with the Penn Treebank test corpus.

Using a larger raw corpus would mean that more of the unknown words could be found, and therefore that there would be a chance for correcting errors with these unknown words. Also, the unknown words that are already present in the Aquaint corpus we used would have bigger counts in a larger corpus. We would therefore be able to get more reliable information about them, and a better idea of how they are used.

Another approach to reducing the effect that small counts have is to perform smoothing on them. The problem is that if a word is seen only once, then one cannot be totally sure of its correct tag, no mat-

ter what context it is in. That one instance will make the feature value either 0 or 1, the two most extreme values, which may be tremendously misrepresentative of what the true value should be.

There are many smoothing methods that can help with this problem. The Good-Turing estimation is one approach which calculates estimated values using the observed counts for the next most frequent word (Good, 1953). It is extremely effective for words with low counts, and would therefore be able to solve this problem quite well.

Another smoothing issue comes from the Gaussian prior that is used within the Maximum Entropy training code. This prior has the effect of keeping the weights within a normal distribution, slightly relaxing the constraints upon the model. It has the effect of not allowing any single weight to get too large, which could mean many tagging decisions were made on a single, possibly unreliable, feature. However, the algorithm does not take into account the introduction of real-valued features, and therefore the possibility of non-binary feature values. As a result of this, some real-valued features may be prevented from having as large a weight as they should, and thus having reduced impact. An investigation into what effects this problem has, as well as changing the implementation to calculate the update values more correctly, would be beneficial.

13 Conclusion

We aimed to use the information in an unannotated corpus — in particular, the contexts surrounding unknown words — in order to increase performance on POS tagging unknown words. Although a number of techniques have been applied to this problem in the past, none attempted to draw upon the information that could be found in a larger amount of raw text. The new feature types we have demonstrated are quite different to those previously used, and we have shown the increase in performance that they can give. The advantage of this method is that it can derive contextual information from any unannotated corpus, and so it is easily portable to another domain.

In particular, the use of real-valued features resulted in a much larger improvement than binary or discretized features would have given us. Maximum Entropy features in the past have always been limited in this respect, and seeing the results we attained, one cannot doubt the benefits that real-valued features can bring. The increased flexibility they give us, and their ability to capture relationships between values, make them extremely advantageous. One can see that the kind of information we were trying to represent was a good example case for their usage, but there are many other features that would also be intrinsically suited to them.

The increase of 1.46% on tagging accuracy for unknown words raises the result to a state-of-the-art level, which will translate to benefits when performing other NLP tasks based on POS tagging information. The smaller increase for NER shows that these methods are also feasible for other tagging tasks, and demonstrates once again, the usefulness of real-valued features.

Acknowledgements

We would like to thank members of the Language Technology Research Group and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

E. Brill. 1994. Some advances in rule based part-of-speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722 – 727, Seattle, WA.

S. Chen and R. Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University.

J. R. Curran and S. Clark. 2003. GIS and smoothing for maximum entropy taggers. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 91 – 98, Budapest, Hungary.

DARPA. 1998. Proceedings of the seventh message understanding conference (MUC-7). Fairfax, Virginia. Morgan Kaufmann Publishers, Inc.

J.N. Darroch and D. Ratcliff. 1972. Generalised iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470 – 1480.

D. Graff. 2002. The AQUAINT corpus of English news text. Technical Report LDC2002T31, Linguistic Data Consortium, Philadelphia.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313 – 330.

A. Mikheev. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405 – 423.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the EMNLP*.

R. Weischedel, M. Meteer, R. Schwartz, L. Ramshaw, and J.Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19.

POS Tagging with a More Informative Tagset

Andrew MacKinlay[†] and **Timothy Baldwin**[‡]

[†‡] Dept. of Computer Science and Software Engineering
University of Melbourne
Victoria 3010 Australia

[‡] NICTA Victoria Research Lab
University of Melbourne
Victoria 3010 Australia

{amack,tim}@cs.mu.oz.au

Abstract

We investigate the impact of introducing finer distinctions into the tagset on the accuracy of part-of-speech tagging. This is a tangential approach to most recent research in the field, which has focussed on applying different algorithms using a very similar set of features. We outline the basic approach to tagset refinement and describe preliminary findings.

1 Introduction

Most recent research on corpus-based part-of-speech (POS) tagging has tended to focus on applying new algorithms to an existing task (Brill, 1995; Ratnaparkhi, 1996; Giménez and Màrquez, 2004), or improving the efficiency of an existing algorithm (Ngai and Florian, 2001). While there has been some successful experimentation with modifying the feature sets of particular taggers (Toutanova and Manning, 2000), the various state-of-the-art taggers for the most part use a very similar set of features in determining the tag for a particular token: some subset of the two preceding and two following tokens, and their tags. The different algorithms have tended to plateau to a similar "glass ceiling" in accuracy ($96.9 \pm 0.3\%$ over all tokens for the taggers in this paper) by using these features.

POS tagging is essentially an optimisation process over firstly the tag sequence and secondly the tag–word assignments for a given input. The relative difficulty of this task hinges on the internals of the POS tagset, and the tagging performance over a given dataset can vary greatly depending on the tagset used. In this paper, we seek to enhance tagging performance by adding a third dimension to the optimisation process: the tagset. We explore the possibility of transforming the tagset via reversible (lossless) mappings, to produce a dataset which is more amenable to automatic tagging and thus results in higher performance than the original tagset. We follow the majority of recent mainstream research on English POS tagging in adopting as a baseline the tagset used in the Penn Treebank (Marcus et al., 1993).

We evaluate two different approaches to identifying patterns of syntactic regularity with the existing POSs. Our primary approach is linguistic insight: we investigate a range of linguistically motivated subdivisions which are either designed to assist in a specific problematic instance of disambiguation, or are linguistically sensible enough to be applied for independent reasons. Additionally we compare this with a data-driven approach, where we attempt to identify intra-POS groupings by running a clustering algorithm over the words within a particular class using features derived from syntactic context. We report the most promising results achieved in both cases.

Section 2 outlines some diverse algorithms which have been applied to POS tagging; Section 3 gives some motivation for attempting increases in accuracy; Section 4 describes details of the tagset used in the Penn Treebank; Section 5 outlines our method; in Section 6 we show results for various strategies and in Section 7 we discuss further work.

2 Tagging Algorithms

A large number of algorithms have been applied to POS tagging; a brief treatment of those which are relevant follows.

2.1 Transformation-Based POS tagging

The transformation-based learning (TBL) paradigm as applied to POS tagging was first described in Brill (1995); like all of the taggers described here it is a corpus-based method. In the learning phase, a TBL tagger assigns each word the most-likely unigram tag from the training data, and generates a large set of possible transformational rules which

Proceedings of the Australasian Language Technology Workshop 2005, pages 40–48,
Sydney, Australia, December 2005.

map the unigram tagger assignments onto the gold-standard assignments, conditioned on contextual word and tag features. It iteratively selects from these the rule which minimizes the number of errors, and applies that rule to modify the assigned tags. The output is an ordered list of rules which can then be applied, in combination with the learned unigram tag probabilities, to unseen data.

The TBL implementation used here is fnTBL 1.1 (Ngai and Florian, 2001); it is equivalent in power to Brill's original but runs two orders of magnitude faster due to optimisations which are not relevant here. The reported accuracy in Brill (1995) was 96.6%/81.2% for known/unknown words using 1M words of the Penn Treebank WSJ Corpus as training data and 200K words as test data.

2.2 Maximum Entropy POS tagging

The maximum entropy framework is a probabilistic approach to NLP commonly used for classification tasks including POS tagging. The approach was applied specifically to POS tagging in Ratnaparkhi (1996). The underlying principle is that when choosing between a number of different probabilistic models for a set of data, the most valid model is the one which makes fewest arbitrary assumptions about the nature of the data.

The probabilistic information in this case comes from a set of binary-valued features which in Ratnaparkhi (1996) are dependent solely on local contextual features: the current word and the two words on either side, and the two preceding tags. In Toutanova and Manning (2000) a number of other hand-tuned features derived from a larger context window are added to assist in disambiguation in problematic words, and activated only upon the occurrence of such words. These optimisations bring the accuracy from the baseline for all/unknown words of 96.76%/84.5% (using a subset of the feaures in Ratnaparkhi (1996)) to 96.86%/86.91%.

2.3 SVM POS tagging

Support vector machines (SVMs) have been applied to POS tagging in Giménez and Màrquez (2004), inter alia. The features are parallel to those used in a maximum entropy model: a set of binary features conditioned on the presence of words and tags within a local context window. These features are then used to build an SVM for each part of speech which contains ambiguous lexical items (reportedly 34 for the Penn Treebank WSJ corpus), and in the classification stage, the most confident prediction from all of the SVMs is selected as the tag for the word. The accuracy reported is 97.16%/89.01% for all/unknown words.

3 Motivation

As noted in Garside et al. (1997), the linguistic quality of a tagset is determined by the extent to which each tag denotes a set of words with a unique set of common syntactic properties, while the computational tractability of it is determined by the ease with which the tag for a particular token can be determined, and how much each tag aids in the disambiguation of surrounding words. The extreme cases of tagsets with either one tag per word or one tag for all words, are examples of tagsets which are highly tractable in computational terms, but of very little use linguistically, which perhaps serves to indicate that these requirements sometimes conflict. However, the aim here is to test whether there is always an inverse relationship between the two. A tagset which encodes more subtle distinctions is almost inevitably more useful in linguistic terms unless the additional distinctions are entirely random; here we will test whether the accuracy can be increased by certain carefully selected tagset subdivisions motivated by linguistic intuition.

Indeed, in Klein and Manning (2003) it was demonstrated that a finer-grained set of category labels can markedly improve performance in the related application of parsing, by providing more contextual information upon which to base decisions in cases of ambiguity. This, along with the demonstration by Toutanova and Manning (2000) that there is potential to improve POS tagging performance by adding linguistically motivated features to the tagger suggests that it may be possible to apply an analogous version of Klein and Manning's method to POS tagging. If we alter the tagset to encode more subtle distinctions within the word classes, these new divisions could potentially increase the computational tractability of the tagset and hence improve the performance of the tagger, since subtler distinctions can provide more useful information to disambiguate surrounding words.

It is worth addressing the question here of why it is worth striving for a small performance improvement here. By NLP standards, accu-

racy of ∼97.0% seems astoundingly high, begging the question of whether there is any point in attempting to raise this figure by a few fractions of a percent. However, according to word-by-word evaluation metrics, POS tagging is actually quite a simple task – as noted by Charniak et al. (1993), the unigram-based most-likely tag (MLT) baseline for the task is around 91%.

The problem is POS tagging is generally a pre-processing phase in NLP, which acts as input to a second stage such as sentence-level parsing. If we look at sentence-level accuracy i.e. the proportion of sentences in which all tokens are correctly tagged, the POS tagging task seems harder – with an average sentence length of ∼24 words and assuming errors occur independently we would expect a tagger which gives 97% accuracy over word tokens to achieve 49% at the sentence level, while a tagger performing at 98% should tag 62% of sentences correctly.

4 The Penn Treebank Tagset

The tagset for the Penn Treebank is based on the tagset used for the original Brown corpus (Francis and Kučera, 1979) but at 36 tags (excluding punctuation), it is small in comparison to both the Brown tagset (75 non-compounded tags[1]), and other related tagsets. This was a deliberate design decision, in that Marcus et al. (1993) set out to create "A Simplified POS Tagset for English" to alleviate problems of sparse data in stochastic applications – thus increasing the computational tractability of the tagset. The primary means by which they achieved this simplification was with by applying the notion of 'recoverability': if the distinctions between several tags could be recovered from either syntactic information (available from the parse tree annotations) or lexical information (the character string making up the word), the tags could be conflated.

The avoidance of lexically recoverable distinctions means that classes with just a single lexical item are dispreferred – hence, for example, the abandonment of the explicit POS distinction between auxiliary verbs and content verbs which is made in most other tagsets derived from the Brown tagset (Francis and Kučera, 1979; Garside et al., 1987; Garside et al., 1997). Additionally the presence of syntactic information means that the traditional distinction between prepositions and subordinating conjunctions can be removed as it can be recovered from the phrasal category of the sibling (*SBAR* for a subordinating conjunction and *NP* for a preposition).

However Marcus et al. (1993, p315) stress that all of this information is available to users of the corpus via additional sources:

> ...the lexical and syntactic recoverability inherent in the POS-tagged version of the Penn Treebank corpus allows end users to employ a much richer tagset than the small one described ... if the need arises.

What is interesting here is that the tagset was not designed to differentiate all possible distributional differences when other information is available, but in examples of POS tagging in the literature, the tagset is invariably used in unaltered form despite the tagger having no explicit access to the syntactic information required to recover sub-usages of a given tag.

The lexical information is used, albeit implicitly, by the inclusion of lexicalised features in all of the state-of-the-art taggers mentioned here. Ironically, the Penn Treebank tagset was designed to be coarse to avoid problems of data sparseness, and yet it is this coarseness which contributes to the inevitable inclusion of sparsely-populated lexicalised features to achieve high accuracy. While there have been examples of certain ad hoc modifications, Manning and Schütze (1999) note that a systematic study into the effect of the tagset has not been explored. It seems the possibility of making explicit certain syntactic regularities within the coarse Penn Treebank word classes for the purposes of improving performance in POS tagging is one worth investigating.

5 Method

We wish to investigate here whether we can improve performance by helping the tagger make syntactic generalisations which are not apparent either from the coarse POS tags or from the sparsely populated lexical feature vector. Subdividing the tags in a linguistically sensible way should hopefully provide this information. However the presence of additional POSs clearly has the potential to make the POS tagging task more difficult. Thus, as shown in Figure 1, we will map the tag of each token in the training

[1] For comparison with the Penn Treebank, where the *'s* suffix is split from the host noun, this figure excludes 12 possesive variants of other tags such as *NN$*

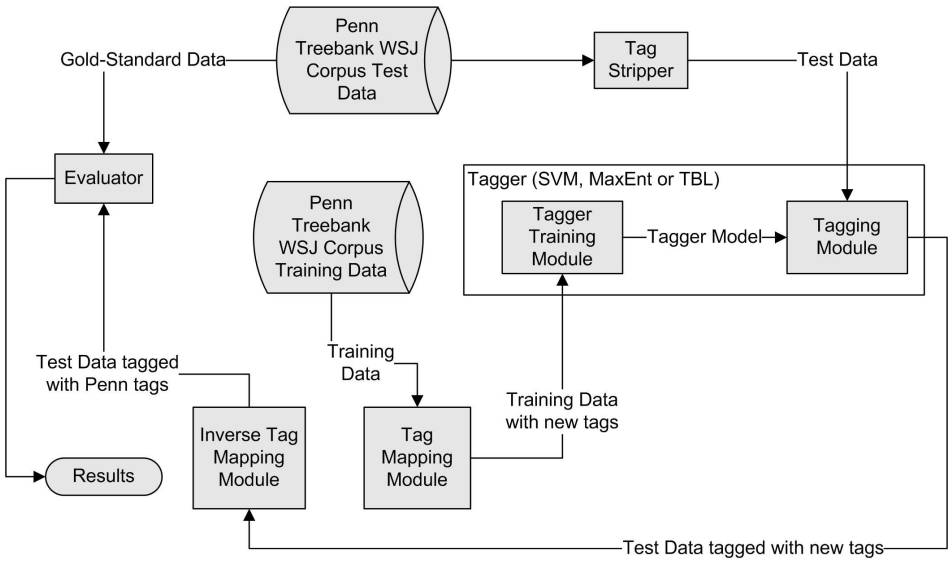


Figure 1: The experimental architecture

data appropriately to a particular new version of the tagset, run the trained tagger over a test corpus, and for purposes of comparison map the finer-grained POS-tags back to the original Penn Treebank tags before evaluating performance. This method means that any increased linguistic utility of the mapped tags is discarded before evaluation, but for the purposes of this experiment the linguistic utility is a means for improving tagger performance rather than an end in itself.

To facilitate the final stage of mapping the tags back to original Penn tags, we place certain restrictions on allowable modifications: the mapping function must be either be injective from the old to the new tags, or any distinctions which are collapsed must be unambiguously recoverable from the wordform so the equivalent tags from the original tagset can be determined reliably.

Our tag-mapping module enables any subset of POS tag assignments to be translated using a conjunctive or disjunctive combination of lexical and syntactic features. Syntactic features include the two surrounding tags, and the phrasal categories of nearby nodes (as defined within the treebank annotation): parent, grandparent, immediately preceding or following siblings, or all preceding or following siblings.

Initial experiments suggested that the marginality of the performance improvements we were aiming for over the data meant that there was a risk of overfitting – even with 200K words of test data, a global change of 0.05% corresponds to only 100 words, or much less over a specific POS; additionally, the inter-annotator discrepancies noted in Ratnaparkhi (1996) are likely to swamp any corpus-wide generalisations. To alleviate this, we used five-fold cross-validation over sections 00-22 of the Penn Treebank WSJ corpus, effectively producing a development set of ∼1M words. Rather than split the data by sections, the data partitions were constructed by placing one sentence from every five in each partition. This tends to inflate performance figures, however this is not a problem here since we are purely looking for improvements relative to the benchmark.

We selected fnTBL (Ngai and Florian, 2001) as our first stage prototyping tool for a set of tagset modifications, as it can complete a five-fold cross-validation test-cycle in under two hours. Any modification which had a large negative impact on performance at this stage was generally not investigated further, since the taggers use a similar set of features, and we were attempting to find universally useful distinctions. The SVM tagger SVMTool 1.2.2 (Giménez and Màrquez, 2004), with a turnaround of under seven hours, was used in subsequent experimentation. Only the Stanford NLP Maximum Entropy tagger (Toutanova and Manning, 2000) had a prohibitive training time, so for practical reasons was used minimally, for benchmarking and later-stage testing.

5.1 Evaluation Metrics

We evaluate the results using several evaluation metrics. First, for comparison with previous work we use the global token-level accuracy metric since it is the most widely-used metric in tagging research. The token-level accuracy over unknown words (i.e. those which did not appear in the training data) is also crucial since this is a major source of tagging errors – in our baseline with an unmodified tagset, just 2.4% of the tokens in the training data were unknown but they contributed 11-13% of errors. Additionally, we show sentence-level accuracy, and precision, recall and F-score over individual POSs.

5.2 Sources of modified tagsets

The primary goal here is to apply linguistic intuition to the task of tagset modification. Potential modifications were drawn from a number of sources, including grammars of English (such as Huddleston and Pullum (2002)) and alternative tagsets, such as CLAWS7 (Garside et al., 1997), and evaluated empirically.

An alternative line of investigation was more data-driven: we investigated whether in a separate stage to training the taggers, we could use machine learning techniques to determine useful subdivisions in the tagsets. To this end, we defined a range of features which could help in determining patterns of syntactic regularity. Some of the features were syntactic, often corresponding to layers of annotation used in Klein and Manning (2003): phrasal categories of the parent, grandparent, left sibling and right sibling, and binary-valued features for whether a given preterminal corresponds to a phrasal head, or whether it is the only element in its phrase. There were also a set of collocational features corresponding more closely to the features available to the tagger, based on the two preceding and two following POSs.

The nominal values of each feature were extracted for each token in the training/development data then conflated by word type and converted into a frequency distribution across the possible feature values for each word type. The value distribution for each feature with n non-zero values was then converted into a set of n real-valued features for the word type using maximum likelihood estimation. This method of combining feature values is not ideal but was the most principled way we could find of capturing a large amount of distributional information manageably.

These feature values were then used as input for the implementation of the EM algorithm in the Weka toolkit (Witten and Frank, 2000). Several different combinations of features were used; broadly, they were syntactic-only, collocational-only and both.

6 Results

6.1 Baseline and Benchmark

The benchmark results from running each of the publicly available taggers with the default or recommended parameter settings are shown in Table 1, with results over specific POSs in Table 2. For a point of comparison, we also applied a suite of naively conceived modifications to illustrate the effects of data sparseness. The idea is borrowed from POS induction, which involves determining word clusters (i.e. POSs) from unannotated data. The task here is similar except that we are looking for patterns of regularity within a particular POS, so the baseline used in Clark (2003) may be informative. To subdivide a part of speech into n subclasses, we assign each of the $(n-1)$ most frequently seen word tokens from the class into $(n-1)$ separate new classes and the remainder to a final subclass. In Table 3, we present the results for $n = 2, 3, 4$ over closed-class POSs of reasonable size, after training and tagging with fnTBL using the same broad indicators shown in Table 1. The best-performing modification from this selection, (i.e. for subdividing *PRP*, with $n = 1$) was additionally tested using SVMTool and the Stanford MaxEnt Tagger; these results are shown in Table 1.

6.2 Linguistically Motivated Modifications

We present here the results for a selection of linguistically motivated modifications which were most successful or most motivated from a theoretical point of view. One obvious candidate modification is reversing the idiosyncratic conflation of prepositions and subordinating conjunctions in the Penn Treebank. This could have been achieved lexically, by extracting a list of lexemes which frequently act as subordinators in the training data, and mapping the tags of the tokens accordingly. However, the most successful and principled approach was using syntactic features for each token and thus deciding on a word-by-word basis. This captures the fact that there are certain words such as *before* that are ambiguous between the two; we let

	TBL			SVM			MaxEnt		
	All	Unk	Sent	All	Unk	Sent	All	Unk	Sent
Benchmark	96.842	81.94	51.77	96.852	84.62	50.72	97.056	87.34	53.72
Freq-based *PRP*:2	96.839	81.81	51.69	96.851	84.60	50.67	97.048	**87.40**	53.51
Freq-based *RB*:3	96.843	81.73	51.72	96.855	84.67	50.71	97.056	87.28	53.72
Clust: IN, All	96.831	81.79	51.52	**96.865**	84.64	**50.90**	**97.065**	87.32	**53.78**
Clust: IN, Coll	**96.850**	**82.00**	**51.82**	96.855	84.61	50.74	97.050	87.32	53.59
Ling: IN—SUB	96.842	81.76	51.63	96.855	84.65	50.77	97.050	87.37	53.51
Ling: RB–DEG	96.818	81.66	51.69	96.847	**84.72**	50.63	–	–	–
Ling: IN—RP	96.832	81.59	51.51	96.851	84.63	50.73	–	–	–

Table 1: Accuracy (%) of the best-performing or most motivated tag modifications for each of the broad methods discussed using five-fold cross-validation over sections 0–22 of the WSJ corpus, with the highest accuracy figure in each column in bold

	TBL		SVM	
	All	Unk	All	Unk
JJ	91.66	76.01	92.22	80.84
JJR	87.55	34.86	88.41	41.44
JJS	93.26	73.87	95.46	70.33
NNPS	65.6	20.78	62.62	19.65
RBR	70.47	–	71.86	–
RBS	78.55	–	86.04	–
VBD	95.06	72.25	95.46	75.25
VBP	92.97	55.46	93.06	44.44

Table 2: Benchmark F-Score (%) over 1,047K words of text, for selected POSs

the tagger resolve this ambiguity as appropriate. Two syntactic features were used to determine if a given *IN* token is a subordinating conjunction (preposition being the default): an *SBAR* parent node or an *S* immediate right sibling.

The results for SVMtool showed very few differences for recall and precision over individual POSs compared to the benchmark: the largest change was a 2% relative increase in F-score over unknown *VBP*s (verb, present tense, non third person singular) and a 3% relative increase for unknown *JJR*s (comparative adjective). The results for fnTBL in comparison were more varied, with a 7% increase in F-score over known members of *RBS* (superlative adverb), and for unknown words, an 8% decrease for *VBP*s and a 5% increase for *JJR*s. The changes in *JJR* are probably due to *than*, which often occurs in their vicinity (e.g. *higher than*) and is usually a preposition by our definition but tends to occur in different contexts to subordinating conjunctions such as *because*. The other differences are harder to account for, and are perhaps unpredictable outcomes due to data sparseness.

Another candidate modification is based on the observation that in the baseline taggers, 5.8-6.4% of tagging errors were due to a gold-standard *JJ* (adjective) being tagged *VBN* (verb past participle) or vice versa, with a further 1.9-2.0% of errors due to the corresponding *JJ*/*VBG* (verb present participle) confusion. This distinction is notoriously difficult to make, but we should be able to assist in discrimination by utilising the linguistic tests distinguishing between the two: adjectives can be modified by degree adverbs such as *very*, while verbs cannot. Thus, the presence of a degree adverb should indicate unequivocally that the head word is an adjective. In reality there is no clear boundary between degree adverbs and the more common verb-modifying adverbs, and empirically the most effective approach, as with the IN–SUB modification, was to allow ambiguity of degree adverb membership and condition the tag mapping on syntactic features for each token: an *RB* with either an *RB* or *JJ* as its right sibling, or an *ADJP* (adjective phrase) as parent was mapped as a degree adverb. This modification is denoted RB–DEG in Table 1.

Compared to the benchmark, the results for SVMtool were again reasonably similar to the baseline, with the only significant differences in F-score being over unknown words: increases of 2% for *JJS* and *VBP*, and 3% for *VBD* which were offset by decreases of 7% for *JJR*. fnTBL over known words gave a 33% relative decrease in F-score for members of *RBS*, and a 5% decrease for *JJS* (superlative adjective), while over unknown words the largest changes in F-score were a 54% increase for *NNPS*, a 3% increase for *VBP*, as well as a 6% decrease for *JJR*. The

POS	IN			DT			PRP		
n	2	3	4	2	3	4	2	3	4
All Tokens	96.823	96.817	96.819	96.806	96.813	96.806	96.839	96.830	96.838
Unknown	81.40	81.78	81.57	81.61	81.78	81.64	81.81	81.95	81.71
Sentences	51.41	51.28	51.55	51.13	51.38	51.28	51.69	51.57	51.60

Table 3: Overall accuracy (%) with naively subdivided POSs using fnTBL

	TBL			SVM			MaxEnt		
	All	Unk	Sent	All	Unk	Sent	All	Unk	Sent
Benchmark	96.68	83.71	49.52	96.75	87.23	49.76	96.99	88.50	52.92
Clust IN, All	96.68	83.59	49.91	**96.78**	**87.38**	**50.04**	96.990	88.47	52.88
IN–SUB	96.70	84.07	**50.00**	96.77	87.32	49.94	96.971	88.29	52.70
VB–INF	**96.73**	**84.10**	49.94	96.75	87.26	49.74	–	–	–

Table 4: Accuracy (%) of selected tag modifications from Table 1 over the held-out 129K-token test set of sections 22 and 23 of the WSJ corpus with sections 0–22 as training data

RBS/*JJS* differences are probably due to confusions between each other for *most* which is often ambiguous when preceding prenominal adjective phrases (e.g. *the most ethical policies*), and *RB-DEG*s which occur in such *ADJP*s lead to spurious generalisations. The differences over unknown *JJR* are probably due to 'degree adverbs' (by our syntactic criteria) such as *much* which modify comparative *ADJP*s and operate quite differently to words such as *very*. Again, we must assume some changes are due to unpredictable data sparseness.

A further round of tests was designed to increase computational tractability with little reference to linguistic motivation. It concerns the ambiguity between *IN* and *RP* (particle). Again, these POSs are notoriously difficult to distinguish between, since many words such as *on* are systematically ambiguous between the two. However, there are many members of *IN* which have no homographs in the *RP* class. If we map the ambiguous members of *IN* to a new class, we are explicitly indicating to the tagger whether or not a word is ambiguous between the two POSs and could improve performance for these particular words. Interestingly, this modification, denoted IN–RP, achieved better performance when applied in conjunction with the IN–SUB modification mentioned (96.832% accuracy over all tokens) above than when it was used alone (96.818% over all tokens).

Various other modifications included retagging verbs based on their likely complements (e.g. if its complement is likely to include a particle, bare infinitive or noun phrase), and several sets of modifications for adverbs, including locative adverbs and those homographic with prepositions. Resultant accuracy using fnTBL ranged from roughly equal to the reported figures to 0.3% below them.

6.3 Intra-POS Clustering Modifications

After running the clustering algorithms with different feature sets as input, we selected a large range of promising sets of POS clusters with a qualitative examination of the clustering output, then starting with fnTBL we successively narrowed down the set of clusters tested with each algorithm by selecting the more successful modifications for the next stage (SVMTool), until finally testing the best-performing modification with the Stanford Maximum Entropy Tagger.

The best performing modification came from using all of the syntactic and collocational features mentioned and resulted in splitting *IN* into four subclasses, corresponding roughly to transitive prepositions (this however included some types such as *before* which can be used as subordinators), rare prepositions, subordinators and a cluster containing only *than*. We also show results for another effective clustering, which again dealt with *IN* using only collocational features. The clusters here do not show such a discernible pattern. In both cases, but particularly the latter, we suspect overfitting due to the fact that the statistics for clustering were derived from the entire combined training/development set.

6.4 Final Testing

To evaluate the validity of our suspicions of overfitting by the clustering algorithm we also show in Table 4 a final round of testing using sections 0-21 as training data and sections 22-23 (which had been held out until this point and were not used to generate clusters) as a test set. This also facilated comparison between the linguistically motivated modifications and the clustering modifications. We would expect the linguistically motivated modifications, which were generated in a fairly data-independent manner (apart from the selection of different modifications on the basis of performance over the development set) to display more consistent improvments over held-out data than the data-driven clusters.

7 Discussion

It is clear from the results shown here[2] that to an extent the intuitions of Marcus et al. (1993) about data sparseness were justified. Table 3 shows that coming up with a modification which reduces performance is easy; we have demonstrated here that finding a set of non-detrimental modifications is difficult. There are probably several reasons for this. It is the most difficult 3% of tokens which we are attempting to tag correctly. Among these are words which probably cannot be tagged correctly with a small context window, words for which humans would have difficulty agreeing on a tag, and words which are tagged incorrectly in the gold standard (a fact which was explored in Ratnaparkhi (1996)).

However despite this, there are still reasons to believe that there is room for improvement. As noted in Brill and Wu (1998), there is high degree of complementarity in errors made by maximum entropy and TBL-based taggers (among others), suggesting that even though these taggers use similar contextual features, the differences in the way these features are combined result in errors over different words. This tends to imply that at least some of the time, there is sufficient information available, but that the different underlying algorithms fail to apply it correctly in all cases.

Given this, the lack of success so far in applying linguistic intuition was surprising. While the highest-performing modification was the linguistically-motivated reintroduction of subordinators, accuracy in this best case was not significantly different from using an unmodified tagset. However the worst of the linguistically motivated modifications resulted in markedly lower accuracy than the benchmark. Even modifications targeted at addressing a specific confusion (such as RB–DEG) actually reduced performance. Additionally, most of these linguistic modifications were outperformed by the best naive frequency-based approach.

The clustering was not designed on a particularly firm theoretical basis; rather, we attempted it as a comparison with the linguistically motivated methods. Despite this, it has produced some intra-POS clusters which (slightly) improve performance, however some of this may be due to overfitting. The performance over the test set, at least for SVMTool, could be seen to support the validity of the result. However examining the output from all three taggers together shows there is very little evidence of consistent improvement from any individual mapping. While they can produce slight improvements for certain taggers in certain cases, these improvements are not significant, and there is little firm evidence on the basis of this experiment for the significant utility of either the data-driven or linguistic approaches.

It is apparent from Table 1 that the best results from various methods seem to asymptote towards the benchmark using the unmodified tagset, which is indicative of the inherent difficulty of the task. Even when we make justifiable modifications, the increased data sparseness usually results in a net performance decrease. While we would not rule out improved results from this line of experimentation, it is likely at least that some variation on the strategy will be necessary for an appreciable increment in tagging accuracy.

Possible further strategies we plan to investigate include adding a two-tiered classification system, by systematically adding delimiters to newly created tags, and adding contextual features dependent on the portion of the POS tag preceding or following the delimiter. Multiple levels of classification of POS tags are used successfully in the JAWS tagging system (Garside et al., 1997) but do not appear to have been applied to the the Penn Treebank. This method would give the taggers access to the more densely populated coarse-tag features when necessary, but when the subtler distinc-

[2] For a more extensive set of results which support the same conclusion, as well as a more detailed discussion of methodology, see MacKinlay (2005)

tions we have added are useful the taggers can utilise them. This is of course a question requiring further experimentation.

References

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 1, pages 191–195, Montreal.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–65.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *National Conference on Artificial Intelligence*, pages 784–789.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of EACL'03: 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 00–00, Budapest, Hungary.

W. N. Francis and H. Kučera. 1979. Brown Corpus manual: Manual of information to accompany a standard corpus of present-day edited American English for use with digital computers. Brown University, Providence, Rhode Island, USA.

Roger Garside, Geoffrey Leech, and Geoffrey Sampson, editors. 1987. *A Computational Analysis of English*. Longman Group UK, Essex, England.

Roger Garside, Geoffrey Leech, and Anthony McEnery, editors. 1997. *Corpus Annotation: Linguistic Information from Computer Text Corpora*. Addison Wesley Longman Ltd, New York, USA.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal.

Rodney Huddleston and Geoffrey K. Pullum, editors. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge, UK.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Andrew MacKinlay. 2005. The effects of part-of-speech tagsets on tagger performance. Honours thesis, University of Melbourne.

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

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

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

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Somerset, New Jersey. Association for Computational Linguistics.

Kristina Toutanova and Christoper D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *2000 Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, Hong Kong, China.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA.

Augmenting Approximate Similarity Searching with Lexical Information

James Gorman and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{jgorman2,james}@it.usyd.edu.au

Abstract

Accurately representing synonymy using distributional similarity requires large volumes of data to reliably represent infrequent words. However, the naïve nearest-neighbour approach to compare context vectors extracted from large corpora scales poorly. The Spatial Approximation Sample Hierarchy (SASH) is a data-structure for performing approximate nearest-neighbour queries, and has been previously used to improve the scalability of distributional similarity searches. We add lexical semantic information from WordNet to the SASH in an attempt to improve the accuracy and efficiency of similarity searches.

1 Introduction

Lexical semantic resources and electronic thesauri are regularly used to solve NLP problems, including collocation discovery (Pearce, 2001), smoothing and estimation (Brown et al., 1992; Clark and Weir, 2001) and question answering (Pasca and Harabagiu, 2001). These use similarity relationships between words, as given in the resources, to enhance corpus-based statistics.

It is difficult to account for the needs of the many domains in which NLP techniques are now being applied and for rapid change in language use. Manual creation is expensive and time consuming, and open to the problems of bias, inconsistency and limited coverage. The assisted or automatic creation and maintenance of these resources would be of great advantage.

Much of the existing work on automatically extracting lexical semantic resources is based on the *distributional hypothesis* that *similar words appear in similar contexts*. Terms are described by collating information about their contexts in a corpus into a vector. These *context vectors* are then compared for similarity. Existing approaches differ primarily in their definition of "context", e.g. the surrounding words or the entire document, and their choice of distance metric for calculating similarity between the context vectors representing each term.

Finding synonyms using distributional similarity requires a nearest-neighbour search over the context vectors of each term. This is computationally intensive, scaling to the number of terms and the size of their context vectors. Curran and Moens (2002) have demonstrated that dramatically increasing the volume of raw input text used to extract context information significantly improves the quality of extracted synonyms. This will increase the size of the vocabulary, decreasing the efficiency of a naïve nearest-neighbour approach.

Using a data-structure such as the Spatial Approximation Sample Hierarchy (SASH; Houle and Sakuma, 2005) allows us to reduce the original $O(n)$ complexity (for an n term vocabulary) to $O(\log n)$ (Gorman and Curran, 2005).

The SASH represents the distributional space as a hierarchical directed graph in which each node is connected to several near-neighbour children, deriving its structure from the distribution of the space it represents. The SASH is searched by traversing these edges.

WordNet (Fellbaum, 1998) is an electronic lexical database. The main unit of organisation within WordNet is the synset, which is a collection of synonymous words. In the case of nouns, there is a secondary organisation based on hyponymy. The structure of WordNet was derived from a model of how humans understand language.

WordNet has been used successfully to solve NLP problems. Clark and Weir (2001) use the WordNet hierarchy to improve probability models of noun-predicate relationships. Pearce (2001) uses WordNet's synsets to improve collocation discovery. We investigate whether using WordNet can improve the accuracy or the efficiency of the SASH algorithm by informing the internal representation with gold-standard lexical semantic knowledge.

2 Measuring Distributional Similarity

We are measuring two classes of semantic relation using distributional similarity: synonymy and hy-

Proceedings of the Australasian Language Technology Workshop 2005, pages 49–56,
Sydney, Australia, December 2005.

ponymy/hypernymy (Curran, 2004). It is hard to distinguish between these two classes using distributional similarity.

Synonymy relates to the nearness of word meaning. Very few cases of true synonymy exist. Instead what exists is near-synonymy, where two words are not directly substitutable, but share some close common meaning. The distinction between loud and noisy is an example of this. They both represent the idea of high volume sound, but noisy also has a negative connotation not present in loud.

Measuring distributional similarity first requires the extraction of context information for each of the vocabulary terms from raw text. These terms are then compared for similarity using a nearest-neighbour search or clustering based on distance calculations between the statistical descriptions of their contexts.

2.1 Extraction Method

A *context relation* is defined as a tuple (w, r, w') where w is a term, which occurs in some grammatical relation r with another word w' in some sentence. We refer to the tuple (r, w') as an *attribute* of w. For example, (dog, direct-obj, walk) indicates that dog was the direct object of walk in a sentence.

Context extraction begins with a Maximum Entropy POS tagger and chunker (Ratnaparkhi, 1996). The SEXTANT relation extractor (Grefenstette, 1994) produces context relations that are then lemmatised using the Minnen et al. (2000) morphological analyser. The relations for each term are collected together and counted, producing a vector of attributes and their frequencies in the corpus.

The syntactic contexts that are extracted by SEXTANT are:

1. term is the subject of a verb
2. term is the (direct/indirect) object of a verb
3. term is modified by a noun or adjective
4. term is modified by a prepositional phrase

2.2 Measures and Weights

Both nearest-neighbour and cluster analysis methods require a distance measure to calculate the similarity between context vectors. Curran (2004) decomposes this into *measure* and *weight* functions. The *measure* function calculates the similarity between two weighted context vectors and the *weight* function calculates a weight from the raw frequency information for each context relation.

For these experiments we use the JACCARD (1) measure and the TTEST (2) weight functions, as Curran (2004) found them to have the best performance in his comparison of many distance measures.

$$\frac{\sum_{(r,w')} \min(\mathrm{wgt}(\mathrm{w}_m, r, w'), \mathrm{wgt}(\mathrm{w}_n, r, w'))}{\sum_{(r,w')} \max(\mathrm{wgt}(\mathrm{w}_m, r, w'), \mathrm{wgt}(\mathrm{w}_n, r, w'))} \quad (1)$$

$$\frac{p(w, r, w') - p(*, r, w')p(w, *, *)}{\sqrt{p(*, r, w')p(w, *, *)}} \quad (2)$$

2.3 Nearest-neighbour search

The simplest algorithm for finding synonyms is a k-nearest-neighbour (k-NN) search, which involves pair-wise vector comparison of the target term with every term in the vocabulary. Given an n term vocabulary and up to m attributes for each term, the asymptotic time complexity of nearest-neighbour search is $O(n^2m)$. This is very expensive, with even a moderate vocabulary making the use of huge datasets infeasible. It is for this reason that the SASH data-structure is used to reduce the time complexity.

3 The SASH

The SASH approximates a k-NN search by precomputing some near neighbours for each node (terms in our case). This produces multiple paths between terms, allowing the SASH to shape itself to the data set (Houle, 2003). The following description is adapted from Houle and Sakuma (2005).

The SASH is a directed, edge-weighted graph with the following properties (see Figure 1):

- Each term corresponds to a unique node.
- The nodes are arranged into a hierarchy of levels, with the bottom level containing $\frac{n}{2}$ nodes and the top containing a single root node. Each level, except the top, will contain half as many nodes as the level below. These are numbered from 1 (top) to h.
- Edges between nodes are linked from consecutive levels. Each node will have at most p *parent* nodes in the level above, and c *child* nodes in the level below.
- Every node must have at least one parent so that all nodes are reachable from the root.

Construction begins with the nodes being randomly distributed between the levels. The SASH is then constructed iteratively by each node finding its closest p parents in the level above. The parent will keep the closest c of these children, forming edges in the graph, and reject the rest. Any nodes without parents after being rejected are then assigned as children of the nearest node in the previous level with fewer than c children.

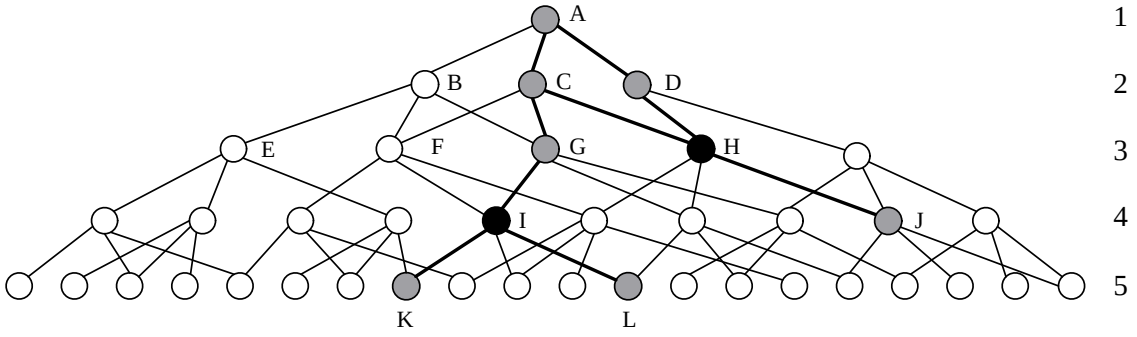


Figure 1: A SASH, where $p = 2$, $c = 3$ and $k = 2$

Searching is performed by finding the k nearest nodes at each level, which are added to a set of near nodes. To limit the search, only those nodes whose parents were found to be nearest at the previous level are searched. The k closest nodes from the set of near nodes are then returned. The search complexity is $ck \log_2 n$.

In Figure 1, the filled nodes demonstrate a search for the near-neighbours of some node q, using $k = 2$. Our search begins with the root node A. As we are using $k = 2$, we must find the two nearest children of A using our similarity measure. In this case, C and D are closer than B. We now find the closest two children of C and D. E is not checked as it is only a child of B. All other nodes are checked, including F and G, which are shared as children by B and C. From this level we chose G and H. We then consider the fourth and fifth levels similarly.

At this point we now have the list of near nodes A, C, D, G, H, I, J, K and L. From this we chose the two nodes nearest q: H and I marked in black. These are returned as the near-neighbours of q.

k can be varied at each level to force a larger number of elements to be tested at the base of the SASH using, for instance, the equation:

$$k_i = \max\{ k^{1-\frac{h-i}{\log_2 n}}, \frac{1}{2}pc \} \quad (3)$$

This changes our search complexity to:

$$\frac{k^{1+\frac{1}{\log_2 n}}}{k^{\frac{1}{\log_2 n}-1}} + \frac{pc^2}{2} \log_2 n \quad (4)$$

(Houle and Sakuma, 2005). We use this geometric function in our experiments.

4 WordNet

Within WordNet (Fellbaum, 1998), words are divided into four syntactic categories: noun, verb, adjective and adverb. Each of these categories has a different structure, representing their use. We are only concerned with nouns in these experiments and, when referring to WordNet, we only refer to this part of it.

The key building block of WordNet is the *synset*: a set of synonymous terms. Words in a synset may not be fully interchangeable, but are in at least some contexts. Because words are organised by concept, polysemous words will appear in several synsets.

Synsets are arranged in a hierarchy based on hyponymic relations. Those near the root are more general, and those near the leaves are more specific.

WordNet 2.1 consists of 117,097 unique terms in 81,426 synsets. Of these terms 15,776 are polysemous, yielding a total of 145,104 word-sense pairs. Our experimental corpus consists of 246,067 unique terms, of which 88,925 remain after a frequency cut-off of 5 is applied. 22,537 terms occur in both WordNet and our corpus, yielding 32,057 senses.

A coarse-grained sense distinction is made by 25 lexicographer files (see Table 1). Each of these represent distinct conceptual and lexical domains and were selected to cover all possible English nouns. These map to the top most synsets in the WordNet hierarchy, either uniquely or as hyponyms.

Table 1 also show the proportion of WordNet covered by each domain (by type), and the proportion of the terms in both the BNC and WordNet in each domain (by token from the BNC). We represent our corpus statistics by token as this is indicative of how reliable the context information is for each domain. Where a term appears in several domains, its count is divided by the number of domains and spread evenly between them, following the Resnik (1995) uniform mass splitting strategy.

WordNet itself can be used to measure semantic similarity. Budanitsky and Hirst (2001) found the method proposed by Jiang and Conrath (1997) to be the most successful in malapropism detection. They used information content to measure the conditional probability of finding a child synset given a parent synset.

Leacock and Chodorow (1998) measure the log

act, activity	7.0%	11.4%	natural object	1.8%	1.7%
animal, fauna	10.9%	4.5%	natural phenomenon	0.8%	0.8%
artifact	12.3%	16.1%	person, human being	13.8%	14.7%
attribute	3.4%	6.5%	plant, flora	12.6%	2.6%
body	2.8%	2.5%	possession	1.1%	1.0%
cognition, knowledge	3.3%	4.9%	process	0.9%	1.3%
communication	6.3%	7.8%	quantity, amount	1.4%	2.0%
event, happening	1.2%	2.2%	relation	0.5%	0.6%
feeling, emotion	0.6%	1.3%	shape	0.4%	0.6%
food	2.8%	2.7%	state	4.4%	5.2%
group, grouping	3.0%	2.2%	substance	3.7%	4.7%
location	3.8%	1.1%	time	1.3%	1.1%
motivation, motive	0.1%	0.1%			

Table 1: 25 lexicographer files (Fellbaum, 1998)

of the path distance between two synsets, scaled by the overall depth of the hierarchy. This performed nearly as well as Jiang and Conrath's method.

5 Evaluation

Our evaluation uses a combination of three electronic thesauri: the Macquarie (Bernard, 1990), Roget's (Roget, 1911) and Moby (Ward, 1996) thesauri. It is possible to use precision and recall measures to evaluate the quality of the extracted thesaurus. To help overcome the problems of direct comparisons we use several measures of system performance: direct matches (DIRECT), inverse rank (INVR), and precision of the top n synonyms (P(n)), for $n = 1$, 5 and 10.

INVR is the sum of the inverse rank of each matching synonym, e.g. matches at ranks 3, 5 and 28 give an inverse rank score of $\frac{1}{3} + \frac{1}{5} + \frac{1}{28}$. With at most 100 synonyms, the maximum INVR score is 5.187. P(n) is the percentage of matching synonyms in the top n extracted synonyms.

The same 300 single-word nouns were used for the evaluation as used by Curran (2004) for his large scale evaluation. These were chosen randomly from WordNet such that they covered a range over the following properties:

frequency Penn Treebank and BNC frequencies

number of senses WordNet and Macquarie senses

specificity depth in the WordNet hierarchy

concreteness distribution across WordNet subtrees

For each of these terms, the closest 100 terms and their similarity score were extracted.

6 Experiments

The contexts were extracted from the non-speech portion of the British National Corpus (Burnard, 1995). All experiments used the JACCARD measure function, the TTEST weight function and a cut-off frequency of 5. The SASH was constructed using the geometric equation for k_i described in Section 3.

The values 1–4, 2–8, 4–16, 8–32 and 16–64 were chosen for number of parents (p) and children (c) in the SASH, giving are range of branching factors to test the balance between *sparseness* and *bushiness*.

As in Gorman and Curran (2005), we use the brute force k-NN search (NAIVE) as our base-line for all our experiments. We also reproduce the results for the fully random distribution (RANDOM), when ordered by frequency (SORT) and when *folded* about some number of relations (FOLD$\mathcal{M}$).

RANDOM is consistent with the original design of the SASH. In accordance with Zipf's law (Zipf, 1949), the majority of the terms have low frequencies, and comparisons with these low frequency terms are unreliable (Curran and Moens, 2002), SORT forces high frequency terms towards the root, producing more accurate results by providing more reliable initial search paths.

Unfortunately, these more reliable search paths are also more expensive to calculate. To mitigate this, FOLD$\mathcal{M}$ chooses *more* accurate initial paths, rather than *most* accurate paths. For each term, if its number of relations m_i is greater than some chosen number of relations $\mathcal{M}$, it is given a new ranking based on the score $\frac{\mathcal{M}^2}{m_i}$. Otherwise its ranking based on its number of relations. This has the effect of pushing very high and very low frequency terms away form the root. The folding points this was tested for were 500, 1000 and 1500.

7 Integrating WordNet

Integrating information from WordNet produces much more complicated sorting schemes. The most direct method of using WordNet would be to *use* the WordNet hierarchy as the top levels of the SASH. Those terms present in our vocabulary and in WordNet would be inserted into the SASH in the same order and with the same linkages as given by Word-

DIST	c	DIRECT	P(1)	P(5)	P(10)	INVR	Time
NAIVE		5.29	60%	47%	39%	1.72	12217ms
RANDOM	8	4.93	61%	47%	39%	1.71	520ms
RANDOM	16	5.23	60%	48%	39%	1.73	872ms
RANDOM	32	5.30	60%	47%	39%	1.74	1899ms
SORT	8	4.89	62%	47%	39%	1.71	317ms
SORT	16	5.30	61%	48%	39%	1.75	677ms
SORT	32	5.32	60%	48%	39%	1.74	1709ms

Table 2: Evaluation of random and fully sorted distributions

Net. Those terms in our vocabulary and not in WordNet would then be inserted into levels below the already linked terms, and then normal SASH building process would link them.

This method is very different to the original design of the SASH. Even when we order by frequency or number of relations, the ordering of semantic relations is still random because synonymy is not a function of frequency. The SASH relies on this randomness to cluster the terms successfully.

Despite the paths in WordNet being between semantically similar terms, the success of this method is doubtful. Many terms at the top of the hierarchy, where searches begin, will not produce reliable measurements. Some, such as thing, are too general to narrow a search. Others, such as psychological feature, will occur with such a low frequency as to make measurement unreliable.

The most specific terms at the bottom of the WordNet hierarchy will have those terms not in WordNet as children. These specific terms are likely to have a lower frequency than terms in the middle of the hierarchy. The low frequency WordNet terms will produce less accurate paths when they find their children during construction, resulting in unreliable searches for terms not in WordNet. The fixed structure of the WordNet paths will also reduce the ability to find new similarities within WordNet as the paths to these will not exist.

Rather than using the knowledge provided by the synsets and hyponymy relations directly, we use the knowledge that both WordNet and the SASH arrange terms as a graph. From WordNet, use additional knowledge from the 25 lexicographer files covering distinct conceptual domains (Table 1).

An analysis of the terms occuring in both WordNet and our corpus shows an uneven distribution. The act, artifact and person domains each represent 10–15% of these terms, while the motivation, relation and shape domains represent less than 1%. When randomly distributed, there will be many more high frequency domains represented at the top of the SASH. The initial paths formed at the top of a SASH determine the accruacy of searches. If

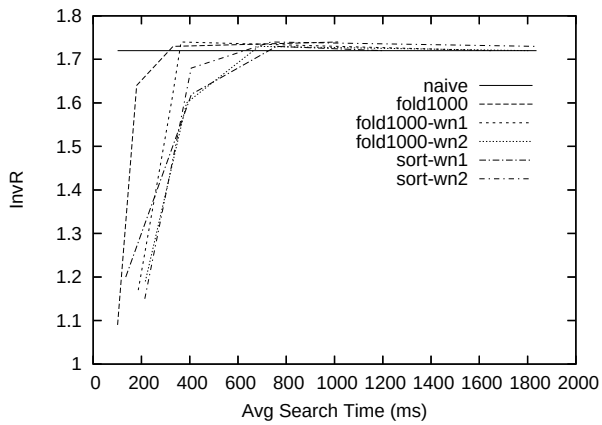


Figure 2: INVR against average search time

the initial path is innaccurate, then there is little chance of finding correct near-neighbours. If a domain is not represented at the top of the SASH, any searches for terms in that domain will first have to pass through domains with which they have little similarity. These paths are likely to be inaccurate, reducing the accuracy for the whole search.

We want all conceptual domains represented evenly at the top levels of the SASH, without overly affecting the distribution of the terms themselves. Both SORT and FOLD$\mathcal{M}$ improve the performance, preserving the distribution of terms, but do not guarantee the even distribution of the domains. We want to ensure this even distribution.

To combine information from WordNet and our existing sorting techniques, we split our vocabulary according to membership of domains. This provides us with 25 lists of terms that appear in WordNet, and single a list of those that do not.

Each of these lists are then sorted by one of the sorting schemes (RANDOM, SORT or FOLD$\mathcal{M}$). The lists are then merged by taking the current topmost term from each list and inserting it into a single list that will be used to create the SASH. For polysemous terms appearing in several lists, the list with the highest sorting is used.

Those terms not appearing in WordNet are treated in two ways. The first (WN1) is to treat them as

DIST	c	DIRECT	P(1)	P(5)	P(10)	INVR	Time
NAIVE		5.29	60%	47%	39%	1.72	12217ms
FOLD500	8	4.24	60%	45%	35%	1.60	185ms
FOLD500	16	5.15	62%	48%	39%	1.75	336ms
FOLD500	32	5.30	60%	48%	39%	1.74	961ms
FOLD1000	8	4.43	60%	46%	37%	1.64	180ms
FOLD1000	**16**	**5.21**	**61%**	**48%**	**39%**	**1.73**	**331ms**
FOLD1000	32	5.31	60%	48%	39%	1.74	1015ms
FOLD1500	8	4.43	59%	45%	37%	1.62	236ms
FOLD1500	16	5.21	61%	48%	39%	1.74	366ms
FOLD1500	32	5.31	60%	48%	39%	1.74	1157ms

Table 3: Evaluation of folded distributions

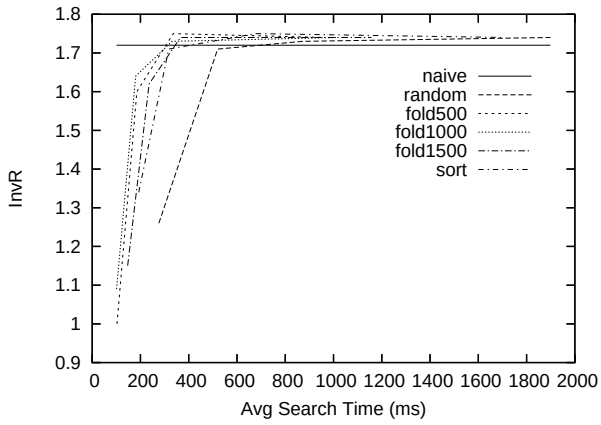


Figure 3: INVR against average search time

a twenty-sixth lexical category and merge them as such. The second (WN2) is to place these terms after those that appear in WordNet. This is more in the spirit of the method, as those terms not in WordNet are not all of a single domain.

Although it would seem that only having one quarter of the terms in the SASH arranged by domain would be too few to have an effect, this represents the top thirteen levels of our SASH, from at total of fifteen. As noise is more significant in initial path formation, the effect of changing the distribution at the top of the SASH has more affect than changing it at the bottom.

8 Results

Figure 2 plots the trade-off between accuracy and efficiency after we have introduced WordNet information into the SASH, using values of c between 4 and 64. The initial sharp increase in efficiency is for values of c from 4 to 8. We see knee points between 400 and 600ms for the WordNet distributions, and 300ms for FOLD1000, when c is between 8 and 16. After the INVR exceeds NAIVE, we have a long tail where INVR converges on NAIVE as the search time increases. What is most interesting in the *sharp* knee of FOLD1000-WN1. From performing worse than FOLD1000, it increases sharply to an equivalent performance, then converges to a an equivalent INVR to NAIVE.

Figure 3 plots the trade-off between accuracy and efficiency for RANDOM, SORT and FOLD1000 using INVR and search time, again using the values of c between 4 and 64. This can be contrasted with Figure 2. Again we have an initial sharp increase in efficiency, and a long tail converging to NAIVE. All the SASH distributions have their knee points at around 300-500ms, when c is between 8 and 16.

Table 2 presents the results for the original NAIVE, RANDOM and SORT experiments. These have been run using an improved implementation of the SASH from that used in Gorman and Curran (2005). Only the results for $c = 8$, 16 and 32 are shown, as these span the knee point. SORT consistently outperformed RANDOM in efficiency and outperformed RANDOM in accuracy for $c \geq 16$. Both SASH solutions outperformed NAIVE in efficiency by more than 14 times when $c = 16$. At $c = 16$, SORT produced similar results for DIRECT and outperformed in INVR by 1%. RANDOM produced a similar INVR and was outperformed in DIRECT by 1%.

Table 3 presents the results for the folded distributions. At $c = 16$, these produced accuracies equivalent to RANDOM, at twice the speed of SORT and 33 times the speed of NAIVE. FOLD1500 was the slowest, although only by 30ms, which cannot be considered significant. Its accuracy was 98% of DIRECT and equivalent INVR of NAIVE. FOLD500 has the highest INVR at 1.75, but the lowest DIRECT at 97% of NAIVE. FOLD1000 provided the best balance with the accuracy of FOLD1500 and the speed of FOLD500.

Table 4 presents the results when WordNet information is used. RANDOM-WN1 similar accuracy to, but is nearly time as fast as RANDOM. RANDOM-WN2 produces similar accuracy, but with only a minor increase in efficiency. SORT-WN1

DIST	DIRECT	P(1)	P(5)	P(10)	INVR	Time
NAIVE	5.29	60%	47%	39%	1.72	12217ms
FOLD1000	**5.21**	**61%**	**48%**	**39%**	**1.73**	**331ms**
RANDOM-WN1	5.24	59%	47%	39%	1.72	488ms
RANDOM-WN2	5.25	59%	48%	39%	1.73	773ms
SORT-WN1	5.26	59%	48%	39%	1.73	759ms
SORT-WN2	5.30	59%	48%	39%	1.74	737ms
FOLD1000-WN1	5.23	59%	48%	39%	1.73	686ms
FOLD1000-WN2	5.23	59%	48%	39%	1.73	686ms

Table 4: Evaluation of WordNet distributions

produces a similar accuracy SORT, but is slower. SORT-WN2 is also slower and suffer a minor accuracy penalty. FOLD1000-WN1 produces a similar accuracy to FOLD1000 and a similar search time. FOLD1000-WN2 produces a similar accuracy and is nearly twice as slow.

The consistent pattern in the results is that once we order by frequency or relations, any improvements in accuracy are not significant. In addition, any improvements from using WordNet information are inconsistent.

9 Analysis

The results for using the SASH without WordNet show that it provides a significant improvement over a naïve search. It is less clear whether adding the WordNet information brings further improvement.

FOLD1000-WN1 produces a result that is similar to the best results for FOLD1000. RANDOM-WN1 is much faster than RANDOM without a loss in accuracy All other results using WordNet are worse.

What we see most here is that there is no obvious pattern to the effects of adding WordNet information to the SASH. In most cases it simply degrades performance, but sometime it improves aspects of it. This occurs for both WN1 and WN2, using different base distributions. A deeper analysis is needed.

There was no general pattern where a distribution of the SASH was more accurate for some term than others except for approximately 25 terms which scored consistently lower or higher when WordNet information was used. These words were compared for polysemy, lexical file membership, depth in the hierarchy, distance, corpus frequency and number of relations. None of these provided any pattern as to identifying either high or low scoring terms.

The analysis of the SASH covered both the construction and the searches. The construction considered the distribution of terms and the number of children of each term. Term distribution was measured by calculating the proportion of terms shared between two distributions for a certain number of terms a the top the distribution. RANDOM distributions share an average of 1% of the top 1000 terms with any other distribution. Between 57% and 69% of terms were shared between distributions with and without WordNet information. Although there was a pattern following the distribution, there was none that indicated its success.

Children were counted to determine if there was a change in the bushiness of the SASH as the distribution changed. This was considered both globally and for each level of the SASH. Again trends were only indicative of the distribution. This was extended to consider the average distance to and the number of relations of each child without yielding further information.

Searches were analysed by measuring the proportion of searching done at each level. This considered the number of terms compared, the number number of relations compared and distance to the search term. This showed no trends.

Given that no trends were found indicating which broad structural and distributional changes had a positive influence, we are left to conclude that the problem lies in the way the SASH clusters particular distributions.

When used as designed the SASH is robust. Our initial distribution functions all produce stable results for various values of c and p. WordNetinformation can improve the performance of the RANDOM distribution, but our SORT and FOLD$\mathcal{M}$ ordering functions increase the stability of the data at the top of the SASH, improving results without needing additional lexical information.

10 Conclusion

We have used lexical semantic information from WordNet to inform the internal structure of the Spacial Approximation Sample Hierarchy (SASH). The SASH has shown the current methods of improving performance to be stable enough that adding this information does not provide any benefit.

That we had some positive results using WordNet, albeit inconsistently, indicates that using lexical information may still provide some improve-

ment in accuracy or efficiency. What this information is and how it should be combined are questions that are yet to be answered. Although dismissed in its simplest form, using WordNet more directly in the SASH presents one possible direction.

We intend to further investigate using lexical semantic information to improve performance, implement other term ordering strategies, as well as further investigating the canonical vector heuristic presented in Gorman and Curran (2005).

Having set out with the aim of applying lexical knowledge to approximate distributional similarity searches, we have found that the existing methods for improving the performance of the SASH are sufficiently robust that this is unnecessary.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback and corrections. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

John R. L. Bernard, editor. 1990. *The Macquarie Encyclopedic Thesaurus*. The Macquarie Library, Sydney, Australia.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Alexander Budanitsky and Graeme Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA, USA, 2–7 June.

Lou Burnard, editor. 1995. *Users Reference Guide British National Corpus Version 1.0*. Oxford University Computing Services, Oxford, UK.

Stephen Clark and David Weir. 2001. Class-based probability estimation using a semantic hierarchy. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 95–102, Pittsburgh, PA, USA, 2–7 June.

James Curran and Marc Moens. 2002. Scaling context space. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 231–238, Philadelphia, PA, USA, 7–12 July.

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

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

James Gorman and James Curran. 2005. Approximate searching for distributional similarity. In *ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, Ann Arbor, MI, USA, 30 June.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston, MA, USA.

Michael E. Houle and Jun Sakuma. 2005. Fast approximate similarity search in extremely high-dimensional data sets. In *Proceedings of the 21st International Conference on Data Engineering*, pages 619–630, Tokyo, Japan, 5–8 April.

Michael E. Houle. 2003. Navigating massive data sets via local clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–552, Washington, DC, USA, 24–27 August.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of 10th International Conference on Research in Computational Linguistics*, pages 19–33, Taipei, Taiwan, 22–24 August.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. In *Fellbaum, 1998*, pages 265–283.

Guido Minnen, John Carroll, and Darren Pearce. 2000. Robust applied morphological generation. In *Proceedings of the 1st International Natural Language Generation Conference*, pages 201–208, Mitzpe Ramon, Israel, 12–16 June.

Marius Pasca and Sanda Harabagiu. 2001. The informative role of WordNet in open-domain question answering. In *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 138–143, Pittsburgh, PA, USA, 2–7 June.

Darren Pearce. 2001. Synonymy in collocation extraction. In *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 41–46, Pittsburgh, PA, USA, 2–7 June.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, 17–18 May.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, Canada, 20–25 August.

Peter Roget. 1911. *Thesaurus of English words and phrases*. Longmans, Green and Co., London, UK.

Grady Ward. 1996. *Moby Thesaurus*. Moby Project.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, USA.

Word Prediction in a Running Text: A Statistical Language Modeling for the Persian Language

Masood Ghayoomi
Institute for Humanities and Cultural Studies, Tehran, 14374
masood29@yahoo.com

Seyyed Mostafa Assi
Institute for Humanities and Cultural Studies, Tehran, 14374
s_m_assi@ihcs.ac.ir

Abstract

Word prediction is the problem of guessing which words are likely to follow in a given segment of a text to help a user with disabilities. As the user enters each letters of the required word, the system displays a list of the most probable words that could appear in that position. In our research we designed and implemented a word predictor for the Persian language. Three standard performance metrics were used to evaluate the system including keystroke saving, the most important one. The system achieved 57.57% saving in keystrokes.

1. Introduction

A word prediction system facilitates typing of a text for a user with physical or cognitive disabilities. As the user enters each letter of the required word, the system displays a list of the most likely completions of the partially typed word. As the user continues typing more letters, the system updates the suggestion list accordingly based on the new context. If the required word is in the list, the user can select it with a single keystroke. Then, the system tries to predict the next word. It displays a list of suggestions to the user. If he finds the next intended word, he selects it; otherwise he enters the first letter of the next word to restrict the suggestions. The process continues to complete the text.

For someone with physical disabilities, each keystroke is an effort; as a result, the prediction system saves the user's energy by reducing his physical effort and also the system assists the user in the composition of the well-formed text qualitatively and quantitatively (Fazly, 2002). Moreover, the system increases user's concentration (Klund and Novak, 2001).

Traditionally, word predictors have been built based on statistical language modeling (SLM) (Gustavii and Pederssen, 2003). SLM is based on the probability of a sequence of *n* given words (*n*-gram). A number of word prediction systems are available today for English, Swedish, and other European languages. Most of these systems have used *n*-gram language modeling.

The current research deals with the design and implementation of a word prediction system based on SLM for the Persian language.

2. Related Works

By looking back, early prediction systems mostly were developed in the 1980s. They were used as a writing assistance system for the one with disabilities. In the early systems, they only suggested the high frequency words that matched the partially typed word and ignored the entire previous context (Swiffin et al, 1985). SoothSayer is such a system. To make suggestions more appropriate, some systems look at a larger context by exploiting word bigram language model beside the word unigram. WordQ (Nantais et al, 2001; Shein et al, 2001) is a system which is developed for English. Profet (Carlberger et al, 1997a; Carlberger et al, 1997b) is a system developed in four

Proceedings of the Australasian Language Technology Workshop 2005, pages 57–63,
Sydney, Australia, December 2005.

languages: English, Norwegian, Polish and French. PAL (Predicative Adaptive Lexicon) is one of the major projects at ACSD (Applied Computer Studies Division) at Dundee University, Scotland (Booth et al, 1990). These systems have used word unigrams and bigrams; also, the systems try being adapted to the user's typing behavior by employing information on the user's recency and frequency of use.
Since there are no previous works of any developed word prediction systems for Persian, what we have done is the first attempt to design and implement a word predictor for this language. We have used the experience of the developed systems for the English and Swedish languages in our research. Details are presented in Ghayoomi (2004).

3. Some Facts about the Persian Language

Persian is a member of the Indo-European languages and has many features in common with them in morphology, syntax, the sound system, and the lexicon. Arabic is from the Semitic family and differs from Persian in many respects.
The Persian alphabet is a modified version of the Arabic alphabet. Hence it is more appropriate to the Arabic sound system and less suitable for Persian. For instance 'ز', 'ذ', 'ض' and 'ظ' are four alphabets both in Persian and Arabic, but all pronounced the same /z/ in Persian and differently in Arabic. So there is a little correspondence between Persian letters and sounds. Although some alphabets are written differently and there is no difference in their pronunciations, they make differentiations in the meanings of words.
Letters have joined or disjoined forms; i.e. based on the position that the letters appear in a word, they have different forms.
Persian writing system is right to left, the same as Arabic; but quite contrary to the European languages that have left to right writing system.
The vocabularies have been greatly influenced by Arabic and to some extent by French, and a great amount of words are borrowed from these languages.
Talking about Persian syntax, only verbs are inflected in the language. The subjective mood is widely used in it. It is an SOV language, and also a free word order language. The language does not make use of gender; not even the third person of he or she distinctions that exists in English (Assi, 2004).

4. N-gram Word Model

The task of predicting the next word can be stated as attempting to estimate the probability function P:

$$P(W_n|W_1,\ldots, W_{n-1})$$

In such a stochastic problem, we use the previous word(s), the history, to predict the next word. To give reasonable prediction to the words which appear together, we try to use Markov assumption that only the last few words affect the next word. So if we construct a model where all histories restrict the word that would appear in the next position, we have then an $(n-1)^{th}$ order Markov model or an *n*-gram word model. (Manning and Schüdze, 1999; Jurafsky and Martin, 2000)
The aim of our study is to design a word predictor that uses a unigram (n=1), bigram (n=2), and trigram (n=3) word model for Persian.

4.1. Word Prediction Algorithm

Suppose the user is typing a sentence and the following sequence has been entered so far from right to left based on Persian writing system:

$$CW_i \quad W_{i-1} \quad W_{i-2} \quad \ldots$$

where W_{i-2} and W_{i-1} are the most recently completed words and CW_i is the current word that is going to be predicted or completed. Let W be the set of all words in the lexicon that likely would appear in that

position. A statistical word prediction algorithm attempts to select the N most appropriate words from W that are likely to be the user's intended words, where N is usually between 1, 5, 9 or 10 based on the experiment done by Soede and Foulds (1986). The general approach is to estimate the probability of each candidate word, $w_i \in W$, being the user's required word in that context.

5. Methodology

5.1.Corpus

To do our research, we made a balanced corpus in different genres from 8 months of the on-line Hamshahri newspaper archive on the web. Although the corpus was small, it was a good representative for the Persian language. The corpus contained approximately 8 million tokens. After downloading the web pages, HTML pages were converted to their plain text equivalents.

5.2. Annotation

The plain text corpus was annotated. One of the annotations was replacing various spellings of a word by a selected spelling. In Persian, some words have various spellings without any changes in the meaning. To choose one spelling among various ones, the highest frequency of use was used to consider the word as the default spelling, and the various spellings were replaced by the selected one. Replacing was done manually. By doing so, the distribution of frequencies of a word with different spellings would be gathered together to assign a single frequency to the selected spelling; because of the smallness of the corpus. For example, these four words were found in the corpus: "امریکایی /?emrikāyi/", "امریکائی /?emrikā?i/", "آمریکایی /?āmrikāyi/", "آمریکائی /?āmrikā?i/". All the words mean "American". Between them, only the spelling "آمریکایی" with the highest frequency of use was selected as default and the other spellings were replaced to that.

The other annotation was removing words or phrases in the corpus from other languages or other Persian dialects comparing to the standard language that do not belong to Persian at all and not be used by native speakers of the language. Email or internet addresses were removed from the corpus. Headlines, footnotes and references in the articles were also removed.

5.3.Tokenization

After annotation, the corpus was divided into three sections: one was the training corpus that contained 6258000 tokens, and 72494 types; the other section was used as the developing corpus which contained 872450 tokens, and the last section was used as the test corpus which contained 11960 tokens.
To do the tokenization process, the training corpus was ran on NSP (N-gram Statistic Package), a program which was written in Perl in Linux (Banerjee and Pedersen, 2003), and uni-, bi-, and trigram statistics were extracted. Words with frequency of one and two regarded as Out-Of-Vocabulary (OOV) and only the most common sequence of words with the frequency of three and more were taken into account and the statistics of word uni-, bi-, and trigrams were extracted. In NSP a token is defined as a continuance sequence of characters to be space delimited alphanumeric strings or individual characters.

5.4. Solving Sparseness

Since a big corpus includes only a fraction of n-grams, increasing n makes the distribution of the events rarer. We have used the Simple Linear Interpolation (SLI) method (Manning and Schüdze, 1999) to smooth the probability distribution.

6. Implementation

6.1. The Algorithm

The architecture of our algorithm is shown in figure 1. The system we developed has four

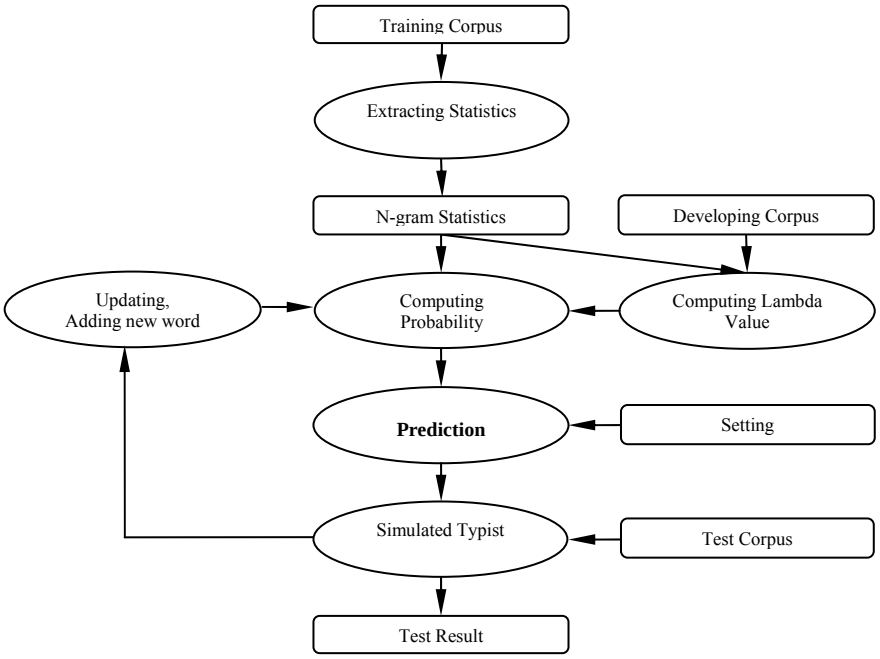


Figure 1: The architecture of our algorithm

major components: a) the statistical information extracted from the training corpus for the prediction algorithm; b) the predictive program that tries to suggest words to the simulated user. This component has two parts: one is word completion and the other one is word prediction. The prediction algorithm first completes the partially spelled word and then it predicts the probable words and present them in the suggestion list; c) a simulate user that types the test text. The simulated typist is a perfect user who always chooses the desired word when it is available in the prediction list and does not miss it; d) the component of updating the statistics of the words' recency of use and adding new words along with their frequency of use. To get the system adaptive to the user, two processes will be done. One is extracting word uni-, bi-, and trigrams from the current text that is being entered. The other process is saving and updating the recent extracted statistical information in a dynamic file. The recent information is related to the static file which keeps the statistical information resulted from the training corpus. When the predictor tries to predict words, first it searches the dynamic file and gives more weight to the words that are recently used; then, it uses the statistical information of the static file. Gradually as the user enters more texts, the system saves and updates the information and gets adapted to the user's style of writing and brings up more appropriate suggestions in the prediction list.

6.2. Conditions

In addition to the word prediction algorithm themselves, the parameter that varied in our experiments was the number of suggestions in the prediction list. It is assumed that the higher number of words in the suggestion list, the greater the chance of having the intended word among the suggestions; but it imposes a cognitive load on the user, because it takes the search time for the desired word longer and it is more likely that the user would miss the word they are looking for. Different users of word prediction systems may prefer different values for this parameter according to their type and level of disabilities. As it has been stated in section 4.1, Soede and Foulds (1986) experimentally identified the number of suggestions. In our work, we selected the values 1, 5 and 9 for the number of

suggestions.
In our system, the sorting order of words in the list is based on the frequency of use in which the most probable words would appear on the top of the list.
Also, in our research we designed a word processor to be compatible with the Persian specifications such as having a right to left writing system to have the cursor in its right direction.

6.3. Performance Measures

To evaluate our system, three standard performance metrics have been used in our research (Woods, 1996; Fazly, 2002):
Keystroke Saving (KSS): The percentage of keystrokes that the user saves by using the word prediction system. A higher value for keystroke saving implies a better performance.
Hit Rate (HR): The percentage of correct words that appear in the suggestion list without entering any letters of the next word. A higher hit rate implies a better performance.
Keystroke until Prediction (KuP): The average number of keystrokes that the user enters for each word before it appears in the prediction list. A lower value for this measure implies a better performance.

7. Results

To test our system, test corpus was given to the simulated typist. The length of the test corpus was 11960 words and contained 46637 characters without considering space as a character. The reason of not considering space is that after selecting any words a space will be entered automatically and the result is having a keystroke saving. On the other hand, to select a word from the list one of the Function Keys, F1 to F9, are required to be pressed to drag and drop the intended word to the text being typed. The result is that the keystroke which is saved by entering the automatic space would be lost.
The virtual typist is a Visual C^{++} program that reads in each text letter by letter. After reading each letters, it determines what the correct prediction for the current position is. The prediction program then is called and a list of suggestions is returned to the user. The user searches the prediction list for the correct prediction. If it is found in the list, the user increases the amount of correct predictions by the predictor. The correctly predicted word is then completed and the user continues to read the rest of the text.
The gained results are presented in table 1 for 1, 5 and 9 numbers of suggestions:

	KSS%	HR%	KuP
1 suggestion	31.67	5.56	2.66
5 suggestions	52.28	18.69	1.86
9 suggestions	57.57	24.42	1.65

Table 1: The summary of the gained results based on the test corpus

Based on table 1, clearly increasing the number of suggestions would increase the percentage of KSS, and HR; and decreasing KuP. The highest KSS is achieved when the numbers of suggestions are 9. The 57.57% KSS means for each 100 characters that the user is required to type to enter a text segment, more than half of the text is entered by the system, and the rest by the user. 24.42% of words appeared in the prediction list before entering any letters of the next word. On average 1.65 keystrokes were needed to be pressed by the user to type any words on the system. There is no valid average word length for Persian, but based on a sampling method from our Persian corpus, the average length is 3.91.

8. Discussion

We conducted another experiment by dividing the test corpus into 23 parts based on their subjects (genres) in the newspaper. Each text segment equally contained 1000 characters, without considering space. Then each text was given to the virtual typist one by one. The results are available in table 2. Using a development set, we found that by using 9 numbers of suggestions we gained the highest KSS. Therefore our final setup

uses the same 9 numbers of suggestions as its default.

Subjects of News	KSS %	HR (9) %	KuP
Arts	67.21	29.21	1.32
Arts and Literature	55.98	24.90	1.55
Cinema	62.50	29.57	1.42
City	62.48	32.22	1.36
Council	66.76	29.88	1.30
Disables	64.53	25.00	1.40
Economics	68.22	35.24	1.19
Education	73.57	40.84	1.03
Environment	61.83	29.85	1.39
Foreign News	69.83	34.76	1.16
Literature	51.54	29.59	1.46
Media	69.00	31.47	1.21
Music	51.53	22.67	1.76
Political News	72.22	38.33	1.07
Rights of Citizens	60.34	31.36	1.44
Science	65.21	30.48	1.27
Science and Culture	56.92	27.43	1.45
Social News	59.03	30.38	1.54
Society	64.08	30.51	1.29
Sports News	70.84	34.63	1.11
Tehran News	68.55	34.00	1.25
Thought	70.78	39.03	1.06
World Sports	63.41	29.14	1.45

Table 2: KSS, HR and KuP performance measures for different genres of test corpora with 1000 characterS

By comparing the results, we observed when KSS increases, HR increases, and KuP decreases; and vice versa. This observation shows that there is a one-to-one correspondence between KSS and HR but they are quite contrary to KuP. Some subjects (genres) such as *Education* achieved the highest KSS, the highest HR and the lowest KuP. But *Music* achieved the lowest KSS, the lowest HR, and the highest KuP.
In general, we saw based on the sequence of words in different genres, it has different effects on the gained results. It seems that the texts on the subjects of *Thought*, *Sports News*, *Political News*, and *Education* which gained the keystroke saving of more than 70%, have more words and sequences of words in common, and the words are more predictable as a result. It means the dependency of words with each other being collocated is high. But the texts on the genres of *Music*, *Literature*, *Arts and Literature*, *Science and Culture*, and *Social News* which gained the keystroke saving of less than 60% have some words that are not available in the lexicon of the program and/or the sequence of the words that come together on these genres are rare and less predictable consequently.
Of course by adapting the system for a special purpose, a better result would be gained as it was described in section 6.1.

9. Conclusion

We have designed, implemented and tested a word predictor for Persian. To the best of our knowledge this is the first attempt for the language. Using such a system saved a great number of keystrokes; and it led to reduction of user's effort.

10. Further Work

Our future work is adding a spell-checker to the system to replace various spellings of a word to the available word in the lexicon of the system, adding syntactic and later semantic information of the Persian language to the system to make predictions more appropriate syntactically and semantically.

Bibliography

Assi, S.M. (2004) "Persian language and IT" In *Proceedings of the 2nd Workshop on Information Technology and Its Disciplines (WITID), Kish Island, Iran*, Feb. 24-26, 2004, pp. 85-94.

Banerjee, S. and T. Pedersen. (2003) "The design, implementation and use of the Ngram Statistics Package (NSP)." In *Proceedings of the 4th International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City*, pp. 370-381.

Booth, L. and W. Beattie and A. Newell (1990) "I know what you mean". *Special Children*, pp. 26-27.

Carlbeger, A. and T. Magnuson and J. Carlberger and H. Wachtmeister and S. Hunnicutt. (1997a)

"Probability-based word prediction for writing support in dyslexia." In Barner, R., Heldner, M., Sullivan, K., and Wretling, P., editors, *Proceedings of Fonetik '97 Conference*, Volume 4, pp. 17-20.

Carlberger, A. and J. Carlberger and T. Magnuson and M.S. Hunnicutt and S.E. Palazuelos-Cagigas and S.A. Navarro. (1997b) "Profet, a new generation of word prediction: An evaluation study." Copestake, A., Langer, S. and Palazuelos-Cagigas S., editors, Natural Language Processing for Communication aids, In *Proceedings of a workshop sponsored by ACL, Madrid, Spain*, pp 23-28.

Fazly, A. (2002) *The Use of Syntax in Word Completion Utilities*. Master dissertation. Canada: University of Toronto.

Ghayoomi, M. (2004) *Word Prediction in Computational Processing of the Persian Language*. Master dissertation. Iran: Islamic Azad University, Tehran Central Branch.

Gustavii, E. and E Pettersson (2003) *A Swedish Grammar for Word Prediction*. Stockholm: Uppsala University

Jurafsky, D. and J.H. Martin. (2000) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey: Prentice-Hall.

Klund, J. and M. Novak (2001) "If word prediction can help, which program do you choose?"
http://trace.wisc.edu/docs/wordprediction2001/index.htm

Manning, C.D., and H. Schütze. (1999) *Foundations of Statistical Natural Language Processing*. The MIT Press.

Nantais, T. and F. Shein and M. Johansson. (2001) "Efficacy of the word prediction algorithm in WordQ™." In *Proceedings of the 24th Annual Conference on Technology and Disability, RESNA.*

Shein, F. and T. Nantais and R. Nishiyama and C. Tam and P. Marshall. (2001) "Word cueing for persons with writing difficulties: WordQ." *The16th Annual International Conference on Technology and Persons with Disabilities, California State University at Northridge, Los Angeles, CA*, March.

Soede, M. and R.A. Foulds (1986) "Dilemma of prediction in communication aids and mental load." *In Proceedings of the 9th Annual Conference on Rehabilitation Technology*, 357-359.

Swiffin, A.L. and J.A. Pickering and J. L. Arnott, and A. F. Newell (1985) "PAL: An effort efficient portable communication aid and keyboard emulator." In *Proceedings of the 8th Annual Conference on Rehabilitation Technology*, pp. 197-199.

Wood, M.E.J. (1996) *Syntactic Pre-Processing in Single-Word Prediction for Disabled People*. Ph.D. dissertation. University of Bristol, Bristol.

Using Diverse Information Sources to Retrieve Samples of Low-Density Languages

Andrew MacKinlay
Department of Computer Science and Software Engineering
University of Melbourne
Parkville VIC 3051 Australia

Abstract

Language samples are useful as an object of study for a diverse range of people. Samples of low-density languages in particular are often valuable in their own right, yet it is these samples which are most difficult to locate, especially in a vast repository of information such as the World Wide Web. We identify here some shortcomings to the more obvious approaches to locating such samples and present an alternative technique based on a search query using publicly available wordlists augmented with geospatial evidence, and show that the technique is successful for a number of languages.

1 Introduction

The utility of language samples to anyone with an interest in a given language is obvious: they can be valuable to linguists, language technologists and speakers of the language, among others. The World Wide Web (WWW) is vast repository of information with potentially large numbers of samples[1] of many languages, but locating these samples reliably is a non-trivial task. While language-specific search tools on search engines such as Google are useful for the languages they cover, for the vast majority of languages which have fewer speakers and a smaller online presence (low-density languages) they provide no information, and it is these lesser-known languages for which language resources are likely to be most useful to language researchers.

There are existing online resources which provide varying degrees of coverage for a large number of languages but the amount of data stored in these for even the best covered languages is generally quite limited. It is likely that there is a large number of documents on the WWW which for a variety of reasons have not made their way into these repositories.

When we expand our focus to include the entire Web, it essentially becomes a problem of language classification, but instead of looking at just a handful of documents as is often the norm in language classification tasks, the number of potential matches is all of the ∼8 billion documents on the WWW indexed by Google. This means the first stage of the classification will necessarily be a search to vastly cut down the number of potential matches.

There are of course other alternatives to finding web-based data for these low-density languages. The most obvious choice of a WWW query using the name of a language works in some cases, but there are of course complications. Languages often have more than one name, such as Adamawe Fulfulde (Niger-Congo, West Africa) which has roughly 33 distinct names including vastly different names such as Biira, Gapelta and Taareyo[2], and, language names can co-incide with words in high-density languages, such as Even (Altaic, Russia). Of course, webpages may not even mention the language in which they are written. The sparseness of these languages means that is crucial to have more than one approach to finding data in order to maximise recall.

The focus here is to investigate an alternative method for classifying and obtaining webpages with a substantial proportion of content written in a given low-density language based on a list of words known to be in the language, and show the additional classification steps required (specifically, analysis of locations mentioned in each document, and a word-unigram classifier) to produce reasonable precision in the documents returned. We have deliberately avoided alternative search methods such as the language-name based web searches mentioned

[1] In the context of this paper, we define a sample of a particular language as a webpage with a substantial proportion of content written in the language

[2] Source: *http://www.rosettaproject.org/*

Proceedings of the Australasian Language Technology Workshop 2005, pages 64–70,
Sydney, Australia, December 2005.

above to focus on the utility of a particular technique.

Section 2 describes the the components of the technique we use, section 3 shows a feasibility test and results for some low-density languages and in section 4 we discuss successes and shortcomings.

2 Method

2.1 Word Unigram-Based Querying and Classification

There are a number of features of a document which can be used to identify its language; our focus here on web-based data retrieval and the word-based nature of most search engines leads naturally to using the presence of particular word-unigrams as our first stage discriminator. For example, a document containing the word 'the', even with uniform prior probabilities for all languages, has a very strong chance of being at least partly in English. For many languages, we can pick a small set of words which will be entirely or partially included in any document in the target language.

Obviously a prerequisite for word-based language sample retrieval is a word list to act as a seed for the search process. For low-density languages, locating such a word-list can be non-trivial. Initially, when we are simply trying to identify candidate documents, the words need not be the most likely words of the language, although if such a list was available it would provide higher initial precision and recall. Assuming that we know very little about the language, a list of words known to be in the language will suffice. Such lists are already available online for some 1,400 languages at the Rosetta Project website. The website contains 'Swadesh lists' for many languages comprised of the translations of a fixed set of 100 or 200 English word-forms.

These Swadesh lists are the basis of the first step of the heuristic described here. To find potential language samples, we obtain a machine-readable version of the Swadesh list (manually converted from the online version due to the presence of idiosyncrasies in individual lists). All of the words are potential search queries; before performing the query, the first stage of filtering takes place, in which the words are compared against a list of stopwords from 17 high-density languages[3] and words that co-incide with these stopwords are removed to avoid too many erroneous matches. In stage 1 of the classification process, the remaining words are passed one-by-one as search queries using the Google Web API to give a list of possible language samples, and the ten top-ranked pages for that search term (this being the maximum allowed by the API) are retrieved and all non plain-text elements removed.

One problem with this search technique is caused by the the Google's page-ranking technique. Samples of low-density languages are unlikely to be among the ten highest ranked documents returned using the Web API, since these ranks are determined by a version of the PageRank algorithm (Brin and Page, 1998) which assigns a ranking based on the number of hypertext links to a webpage. This is probably the largest schortcoming of our technique. Ideally, it would be preferable to use a search engine which ranks pages based on the well-known TF-IDF metric used in information retrieval, however it is also important that the search engine has good coverage and an interface for automated querying. We were not able to locate a search engine which definitely fulfilled all of these criteria, not least because the details of the underlying ranking algorithms are generally trade secrets so determining the extent of TF-IDF usage is almost impossible. These considerations encouraged persisting with the default choice of Google.

The only way we found to alleviate the problem is based on the observation that if a query term produces just a few search results, there is a reasonable chance one of our target documents occurs in the top ten, but as the number of results becomes larger, the likelihood of this decreases. Therefore, if we group retrieved pages by the query term which produced them, and rank them in increasing order according to the number of hits produced by the query term, the most likely matches will be ordered first. This information can then be used as a heuristic to guide the manual refinement stage discussed below. Note that, while the motivation differs somewhat, in practice this is essentially the same as the inverse document frequency in the well-known TF-IDF metric used in information retrieval, since we are effectively weighting each term with a function which decreases monotonically as its document frequency increases.

[3]These are from some of the most widely spoken Roman-script languages including most crucially English, French, German and Spanish, the four most common languages for web content according to Google

2.2 Adding Geospatial Evidence

After fetching the query matches, the set of possible samples of the target language contains a high proportion of false positives, and in line with the assumption that we have very little information on the language, we assume we do not have a language classifier trained to distinguish between the documents. Clearly another layer of refinement is necessary, but it cannot depend on knowledge of the language.

For this, we use another piece of supporting evidence for the language of a document: the geospatial locations mentioned in it. Webpages frequently include local references and we would expect that, since lower-density languages tend to be confined to one geographic area, the locations mentioned in such documents will show similar geographic restrictions. Thus, the presence in a document of a reference to a location which we know to be in the area where the language in question is spoken provides reasonable evidence for the language of a document – not indisputable evidence, but certainly a sign that the document should be considered quite a likely candidate. This classification method will be denoted LLC for 'location lookup classifier'

To obviate the need for a location tagger trained specifically for the language, we take what amounts to a 'brute force' approach for stage 2 of the classification process. The target locations are taken to be any location in the countries where the language, according to the Rosetta Project site, is spoken. The list of cities and region in each country is easily obtainable from the UN LoCODES database.[4] The appropriate locations are extracted from the file, and we perform an uninformed linear search over each putative sample for any of the location names. Any documents which contain an appropriate location reference are tagged as such.

2.3 Further Refinement and Manual Analysis

One final layer of refinement is applied to the data to vastly cut down the manual analysis work. An existing language classifier trained on the high-density languages, such as van Noord's implementation TextCat[5] of the algorithm in Cavnar and Trenkle (1994), is used to test any documents which were candidate matches according to the previous two criteria. If the classifier can unambiguously determine the language of the document, it is almost certainly a sample of that language rather than of the target language. However, if there is any ambiguity, the document is deemed worthy of further examination by a human.

Applying the filters described so far cuts the number of potential samples from ∼4 billion to a much more manageable number. From the results obtained, the number of potential matches at this point is between zero and 600. At this point a manual analysis to find at least one matching document is quite feasible by looking at the set of search results ranked in order of the number of query-term matches.

There is an obvious difficulty of determining whether a document is in a language with which we are not familiar. In practice there are a large number of hints we can use: the URL or the presence of the name of the language, or the aforementioned lexical resources. At this point most of the potential matches are pathological webpages containing very few words, genuine incorrect languages that TextCat failed to recognise unambiguously, documents containing more than one language, samples of similar languages, and genuine samples of the target language. Of these only the samples of similar languages are any trouble to distinguish from genuine samples, and bilingual documents can be set aside for reference.

Once at least one potential sample is identified, we use it to train some sort of probabilistic classifier to identify other samples either in the remainder of the possible matches from stage 2 if the number was large, or in the residue of documents retrieved in stage 1 but discarded due to the absence of an appropriate location. We are simply trying to make the binary distinction between genuine and spurious samples, so any legitimate Bayesian classifier at this stage would suffice, as long as it could be trained on a single document and achieve reasonable performance. As noted by Dunning (1994) a word unigram-based classifier should produce acceptable performance in this case, given that we are only interested in samples of reasonable length, and these are the samples on which such a classifier can perform well. Additionally it has the advantage of being able to easily produce a list of the most likely words.

We simply obtain a word unigram frequency distribution from one document and produce a smoothed probability distribution from it using Witten-Bell discounting (Jurafsky and Martin,

[4] *http://www.unece.org/locode*

[5] *http://odur.let.rug.nl/~vannoord/TextCat/*

2000). Then, simplifying the method of Dunning (1994), we use the distibution to determine the logarithm of the probability of each test document according to the model as the sum of log probabilities of each word according to the model. This is then divided by the number of tokens in the document to give a normalised mean log probability per token for each document according to the trained model, which is essentially the cross entropy of each test document.

Since each document needs only to be classified as a positive or negative sample, they are classified as genuine or spurious on the basis of their cross entropy relative to a threshold. Any estimate of the appropriate cutoff value amounts to making arbitrary assumptions about the distribution. Instead we arbitrarily choose a cutoff point between the ranges of the minimum and maximum entropy values, which is empirically determined. In practice the value is used to rank the documents and it is a simple task for a human annotator to determine a reasonable cutoff in each case.

At this point we hopefully have a reasonably sized list of language samples. If this is the case, the same classifier is then trained on all of the documents and the most probable words are output. The number here can vary – ideally we would like 5-7 common words (probably stopwords of that language) after the stop-words from the high-density languages have been removed. This is a manageable number that can them form the basis of a new round of Google queries, but since we have the most common words of the language a query on the words in combination can be almost guaranteed to include any valid samples of the target language (albeit possibly not in the top 10), but is a more restrictive query, thus avoiding large numbers of spurious matches. So at this stage we perform a Google query on every possible combination of one or more of the common words. Since we have restricted the number of words, this is quite manageable: $2^n - 1$ for n words, giving 127 queries for 7 words.

The result of this query is new set of pages which should be even more likely to be language samples than in the previous iteration of the process. The previous refinement stages can then be run again in a similar fashion, and at this stage assuming no bottlenecks in the process, we have a set of sample documents as well a set of stopwords which can be used for further queries.

3 Results

3.1 Feasibility testing

The initial phase of testing was determining the feasibility of the method. To achieve this, it was necessary to choose languages for which there was a clear easily accessible gold-standard. The obvious choice was to use languages identifiable by TextCat. We chose four arbitrary medium density languages: Finnish, Polish, Rumanian and Icelandic. The method obviously differs slightly from that outlined above – we check to see whether TextCat exactly guesses the target language, rather than whether it is unsure of the language.

We evaluated the stages of the technique in several ways. In table 1 we evaluate the precision at each stage of the refinement process. It is clear that the technique of a Google query on the terms in a language works reasonably well in producing a reasonable proportion of genuine samples, and also that precisions ranging from acceptable to very impressive can be obtained through the combination of a web-query and geospatial location lookup. In Table 2 we show the precision and recall for the location lookup method relative to the refined set of language samples retrieved from the web-query. We included this to show that, working from the sample space of the URLs fetched in stage 1, the location-lookup method retrieves a reasonable proportion of the positive samples from all of the possible samples it receives. Note that the figures for precision are identical in tables 1 and 2 since we are looking at the same set of classifications each time.

It is also necessary to evaluate the validity of the technique of using word unigram cross entropy estimation and picking an arbitrary cutoff point. We selected a random correctly classified sample as training data, and for each of the genuine and spurious matches from the documents classified positively after a location lookup and web-query we calculated the cross entropy relative to this training data. This enabled evaluation of the precision and recall we would have achieved by selecting a particular cutoff value for the cross entropy. If we have minimum and maximum cross entropy values H_{min} and H_{max} respectively, and an absolute cross-entropy cutoff H_{thresh}, the factor k in ta-

Language	Icelandic	Finnish	Rumanian	Polish
Documents Retrieved in web query	1314	2090	1765	1921
True Samples Retrieved	438	912	385	673
Precision of Rosetta-based query	33.3%	43.6%	21.8%	35.0%
Positive Classifications by Location Lookup	153	261	340	481
Genuine Samples in Positives	146	176	267	389
Precision of Location Lookup/web query	95.4%	78.5%	67.4%	80.9%

Table 1: Precision of Rosetta-seeded web query and location lookup/web query combined

Language	Icelandic	Finnish	Rumanian	Polish
Genuine Samples Previously Retrieved	438	912	385	673
Positive Classifications	153	261	340	481
Genuine Samples in Positives	146	176	267	389
Recall	33.3%	19.3%	69.4%	57.8%
Precision	95.4%	78.5%	67.4%	80.9%
F-Score ($\alpha = 0.5$)	49.4%	30.0%	73.7%	67.4%

Table 2: Precision, Recall and F-Score of location lookup relative to retrieved query matches

ble 3 is calculated as:

$$k = \frac{H_{thresh} - H_{min}}{H_{max} - H_{min}}$$

Here the exemplifying language was Romanian; other languages give similar results. Clearly a high F-score can be obtained by careful selection of the cutoff point.

3.2 Genuine Low Density Languages

These results make it clear that the technique is at least worth pursuing. While we would only expect the results to get worse when we move into lower density languages, there is certainly enough accuracy that we can reasonably expect the technique to work in some cases.

In Table 4 the number of documents in each classification stage are shown with 'LLC +' denoting the number of documents positively classified by LLC and 'TC ?' denoting the number of documents ambiguous or unknown according to TextCat (which, as we have explained above, are the documents which are possible samples of the language). The results were obtained by checking whether there was at least one potential training document, and selecting an appropriate document from this set as training data for classifying genuine samples, then using all sufficiently uniform genuine samples as training for the purpose of extracting the most frequent words. Five or six infrequent words which did not coincide with stopwords for high-density languages were then selected for the second iteration.

As mentioned above, the number of genuine samples here was determined by manually examining documents in order of cross-entropy relative to a suitable training document and determining a cutoff point. Note that uncertainty figures are due to the presence of non-canonical documents in the word-list such as bilingual documents often containing little observable structure (such as many retrieved from a Pampangan web forum) and very short documents for which the language was unclear. In the second iteration, some of the documents retrieved were repetitions of pages already retrieved; the column labelled "New" only counts those that were unique to the second iteration.

While from these figures it may appear that the location lookup in the second iteration achieved very little, some additional experiments tended to indicate that no extra samples wore omitted by this technique.

4 Discussion and Further Work

While there are some encouraging results, certain aspects of the technique were not as useful as we had predicted. One aspect worth commenting on is that the crude location lookup was often less useful than we expected from our experiments with medium density languages. In many cases (notably the two where the most samples were retrieved for the genuine low-density languages), this stage of refinement re-

Cutoff k	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Samples Included	44	82	128	150	173	195	216	230	251	259
Precision	100.0	100.0	100.0	100.0	96.0	88.3	80.2	75.8	69.4	67.6
Recall	25.7	47.4	73.7	86.3	95.4	98.9	99.4	100.0	100.0	100.0
F-Score ($\alpha = 0.5$)	40.9	64.3	84.9	92.6	95.7	93.3	88.8	86.2	82.0	80.6

Table 3: Precision, Recall and F-Score (%) over Romanian documents for different cross entropy cutoffs

Language	1st Iteration				2nd Iteration				
	URLs	LLC +	TC ?	Genuine	URLs	LLC +	TC ?	Genuine	New
Mikir	1557	104	31	1	64	11	5	2	0
Fasu	1494	106	32	0	–	–	–	–	–
Hixkaryana	620	580	181	0	–	–	–	–	–
Hmong Daw	841	830	299	120±10	117	116	60	30±5	30±5
Mopon Maya	769	26	4	0	–	–	–	–	–
Pampangan	1839	1797	621	22 ±8	110	109	45	8	5
Warao	998	27	6	0	–	–	–	–	–
Fasu	1046	84	25	0	–	–	–	–	–
Tulu	852	80	18	0	–	–	–	–	–
Quiché	625	29	7	3	97	5	5	5	3

Table 4: Number of Documents at Each Stage of Classification

moved less than 10% of the possible samples, even though in subsequent stages many were positively identified as samples of other languages by TextCat and consequently removed. It would be possible to refine this aspect by applying the TF-IDF metric to the LLC method; this is an area for furher work.

The technique could also be augmented with other search techniques such as as the obvious web query for the name(s) of the language. Alternatively, one or more stages could be used in a more comprehensive language retrieval system.

4.1 Shortcomings and Solutions

While there is some promise to the technique, here we note a number of areas of weakness of this work-in-progress, with potential solutions where they can be identified:

1. There is no obvious starting point when there is no Rosetta wordlist or when the wordlist exists but the orthography is non-standard, either because no standard orthography exists or because the contributor failed to use it. In cases like these obviously another wordlist could be substituted, or if a training document is available we could render the first stage of the iteration unnecessary. While documents obtained in an *ad hoc* fashion could be used for this purpose, there are freely accessible repositories of possible training documents. The UN Declaration of Human Rights has been translated in over 300 languages [6], and while this does not approach the coverage of the Rosetta lists, the UN documents avoid the aforementioned problems with orthography in these lists.

2. The Google page-rank problem already outlined is an obvious issue. In the first iteration many possible matches are missed. We might expect that the second iteration would obtain some of these missed documents with its more precise queries, but the top-ranked among these tend to be the same documents previously retrieved, meaning that some proportion of the top-10 list will be wasted, and the number of douments retrieved is still limited. We have already discussed in Section 2.1 the desirability of a search engines with different underlying algorithms, and the difficulty of locating such a search engine. It is difficult to tell whether the tests in table 4 which produced no results did so because of the page-rank problem or because such a small amount of data is available.

3. There are of course other possible charac-

[6] Source: *http://www.unhchr.ch/udhr/navigate/alpha.htm*

teristics of the language which could render them unsuitable for this method. It will work best on isolating languages, as will probably be the case for any word-based search. For languages with large amounts of morphology, such as polysynthetic languages in the extreme case, the existence of large numbers of wordforms for any lemma means that this technique is less likely to work. Nonetheless, it is clear that technique works effectively on at least one agglutinative language (Finnish) and one inflectional one (Icelandic). Indeed, this possibly explains some of the variation between results. One way to alleviate the problem might be to replace word unigrams in the classifier with character n-grams, which are a well-studied method for language identification described by Dunning (1994), or even to use the documents as training data for a TextCat-like classifier (which is also n-gram based albeit with no reference to cross-entropy). The relative performance of such a technique would need to be evalutated empirically.

5 Conclusion

We have presented an alternative technique for locating samples of low-density languages based on web queries using publicly available wordlists, and refining the results using geospatial evidence and existing language identification algorithms. Despite the novelty of the technique presented and the crude underlying methods in certain parts, we have shown it can in certain cases be an effective means of retrieving samples of low-density languages, free of some of the shortcomings of more obvious techniques.

6 Acknowledgements

We thank Steven Bird and Baden Hughes for guidance and formulating the original idea, and the anonymous reviewers for many helpful comments which have contributed to this final paper.

References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Ted Dunning. 1994. Statistical identification of language. Technical Report MCCS 940-273, Computing Research Laboratory, New Mexico State University.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall Series in Artificial Intelligence. Prentice-Hall.

Faking it: Synthetic text-to-speech synthesis for under-resourced languages – Experimental design

Harold Somers
School of Informatics
University of Manchester
Manchester, UK
Harold.Somers@manchester.ac.uk

Abstract

Speech synthesis or text-to-speech (TTS) systems are currently available for a number of the world's major languages, but for thousands of the world's 'minor' languages no such technology is available. While awaiting the development of such technology, we would like to try the stop-gap solution of using an existing TTS system for a major language (the base language) to 'fake' TTS for a minor language (the target language). This paper describes the design for an experiment which involves finding a suitable base language for the Australian Aboriginal language Pitjantjajara as a target language, and evaluating its usability in the real-life situation of providing language technology support for speakers of the target language whose understanding of the local majority language is limited, for example in the scenario of going to the doctor.

1. Introduction

Speech synthesis systems, in particular text-to-speech (TTS) systems which 'read out' ordinary text on the computer, are now fairly widespread and are sufficiently reliable and of a suitable quality for wide acceptance and use. However, this is only true for the 'major' languages. For example, Microsoft's *Agent* includes American and British English, Dutch, French, German, Italian, Japanese, Korean, Portuguese, Russian and Spanish. Scansoft's *RealSpeak* provides for all of the above, plus Basque, Cantonese, Mandarin, Danish, two varieties of Dutch, Australian and Indian English, Canadian French, Norwegian, Polish, and two varieties of Portuguese. The list is impressive, but there are still thousands of languages not covered.

Our interest is in providing language technology-based support for speakers of 'minor' languages[1] when they find themselves in situations where their lack of ability in another language is a significant disadvantage: we have been focusing on the case of newly arrived immigrants seeking healthcare (a visit to the doctor), but the possibilities are almost endless. This is the scenario envisaged by the CANES framework,[2] as described by Somers and Lovel (2003) and Somers et al. (2004). This envisages software support for getting general information about healthcare problems, arranging an appointment at the clinic or hospital, understanding information leaflets and instructions regarding treatment and drugs, and of course face-to-face meetings with healthcare providers, notably GPs and nurses. Other projects in this field have focused on spoken language translation (SLT) of the doctor–patient interview (Rayner et al., 2003; Narayanan et al., 2004). We have recognized that, while face-to-face dialogues have an important role in the pathway to healthcare, other means of communication play an equally important role, some of them text-based. In all these cases, we see TTS as an essential technology, particularly for users who not only may have limited or no English, but also whose reading ability in their own language may be poor, whether due to low literacy, dyslexia or visual impairment.

A long-term solution is of course to develop TTS tools for more languages, but this is by no means trivial. Currently, development of a TTS system depends on an extensive phonological analysis of the language to identify the individual speech sounds (phonemes) and their variants (allophones); development of text-to-phoneme rules to identify how the orthography of the language relates to the phonology, and rules to determine the pitch and duration features (prosody); and, depending on the approach taken, recording of human speech and extracting

[1] A 'minor' language is any language which is not a 'major' language, as (extensionally) defined above. This is not to be confused with a 'minority' language, since the list of minor languages includes some of the most widely spoken languages in the world (e.g. lists of languages with the most speakers include Bengali, Hindi/Urdu and Arabic in the top 5 or 6).

[2] Computer Assistance for Non-English Speakers.

Proceedings of the Australasian Language Technology Workshop 2005, pages 71–77,
Sydney, Australia, December 2005.

hundreds of individual speech elements (diphones, triphones or demisyllables) or modelling a similar number of elements using a formant synthesizer.

While waiting for this work to be done, in the meantime we want to try using an existing major-language TTS system, as is, to fake TTS for a minor language, in this case, the Australian Aboriginal language Pitjantjatjara.[3] We are undertaking a similar experiment with Somali (Somers et al., 2006)

The idea has been briefly explored by Evans et al. (2002), who have dubbed the process 'gibbering',[4] whereby speech synthesizers for new languages that are suitable for use with a screen reader are produced with a minimum amount of development time, and can be made available at no cost to the user. They suggest that

> … the minimum requirement is that the speech has to be consistent and understandable, but does not necessarily have to be the especially natural sounding or indeed linguistically accurate. The key requirement is that the speech synthesiser speaks the language consistently and can be fully understood by a speaker of the language. (p. 576)

2. Text-to-speech synthesis

Most TTS systems consist of two elements: a text-to-phoneme stage, where the basic pronunciation of the text is determined, and a phoneme-to-speech stage, where the actual speech sounds are generated. It is beyond the scope of this paper to describe detail how TTS works, but we do need to explain how the basic design of TTS systems relates to faking it.

2.1. Text to phonemes

The first stage involves identifying the phonemes to be uttered, but also the pitch and duration, in order to produce appropriate prosody (intonation and stress). This is generally done on the basis of letter-to-sound mapping rules, together with a dictionary where any irregular cases are made explicit. As well as cases of anomalous spellings,[5] the dictionary must 'spell out' abbreviations, numbers, symbols (e.g. &)[6] and so on. The text-to-phoneme module must also contain rules that indicate unusual readings for sequences of symbols, e.g. *$5* is pronounced <five dollars>, not <dollar five>. In addition, most languages have homographs, the pronunciation of which may need more or less sophisticated syntactic analysis to determine. This analysis may also contribute information about prosody.

The problems with this module for faking it arise first from differences in the letter-to-sound mapping rules between the language for which the TTS system was designed (henceforth, following Evans et al. (2002), the 'base language', BL) and the language we are trying to produce (the 'target language', TL). For example, while a *j* is pronounced [dʒ] in English, in Spanish it is [x]. Further, some words in the TL may be written the same way as words in the BL, but pronounced differently, e.g. *train* in English [tɹein] and French [tʁɛ̃]. The rules for reading symbols are generally language-specific, as are any rules relating to prosody.

It may be of course that the TL does not use the same writing system as the BL, or does not have a writing system at all.

In all the above cases, one thing we can do is to try to rewrite the TL word so that it follows the rules of the BL, for example rewriting French *train* as <tran> (though see next section on phoneme-to-speech).

Some TTS synthesizers accept as input streams of phonemes instead of plain text. Depending on the software, these may be in a kind of transcription (e.g. <dh-ah-s> for *thus*), or IPA symbols may be used. In their approach, Evans et al. (2002) go somewhat further:

> The rules [for text-to-phoneme translation] are contained in a text-based table and applied by a generic piece of software that is capable of applying any appropriately specified rule set. Thus, to construct a new language the text to phoneme rules for the target language need to be developed and encoded in a table. (p. 578)

In fact, this approach is a significant alternative to our method, with some advantages and disadvantages. They have to define the mapping for all the phonemes, whereas we only define the ones that are different. But they

---

[3] This work was initiated while the author was on study leave at the Centre for Language Technology, Macquarie University , Sydney.

[4] To 'gibber' is to talk foolishly, or in a way that cannot be understood, hence 'gibberish'. In our case, the input to the TTS system is gibberish from the system's point of view, but, we hope, meaningful to the listener.

[5] It is interesting that most laypersons usually assume that if a word is not pronounced as it is spelt it is because the pronunciation is anomalous; however, since writing is a representation of speech and not vice versa, it is of course the written form that is 'wrong'. Whichever view is taken, it is clear that we must have a list of words for which the 'rules' do not apply.

[6] In this paper the following conventions are used: text strings and graphemes are shown *in italics*; strings to be read out by the TTS system are shown <in angle brackets>; as is customary, IPA phonetic symbols are shown in square brackets [ð□s], while slant brackets / / are used to indicate phonemes.

potentially have more control over intonation, and can define a dictionary of special pronunciations, e.g. for the numbers and symbols.

2.2. Phonemes to speech

In the second stage, the actual speech sounds are generated, whether by concatenating prerecorded human speech or by formant synthesis. The main problem for faking it is that the set of speech sounds for the TL will almost certainly not be the same as for the BL, and even if they are similar, it is likely that the rules for allophonic variation will be different. Some phonemes from the TL will simply be missing; others may be similar to phonemes in the BL, but may differ in the realization of different allophones.

The trick is obviously to choose a BL where this problem is minimized, though there may be a considerable trade-off between finding a BL with a good overlap in the letter-to-sound mapping rules mentioned in the previous section, versus good coverage of the target phonemes and allophones.

One of the goals in the research described here is to try to evaluate which of a number of BLs is most suitable for a given TL, and to identify which factors make identifying the best BL easier.

3. Faking it for Pitjantjatjara

Pitjantjatjara is an Australian Aboriginal language, spoken by the Anangu people, best known to tourists as the traditional owners of the land which includes Uluru, formerly known as Ayers Rock, in Northern Territory. Pitjantjatjara has about 1,300 speakers, but is one of a group of mutually intelligible dialects which form the Western Desert language group which, with around 4,000 speakers is one of the three or four 'largest' Aboriginal languages. According to Eckert and Hudson (1991), Pitjantjatjara has been written in the Roman alphabet since the 1940s, and the orthography has been more or less standardized since the 1979 meeting of Pitjantjatjara literates, and confirmed by the publication in 1987 of a Pitjantjatjara–English dictionary (Institute for Aboriginal Development, Alice Springs).

3.1. Pitjantjatjara phonology

Like many Aboriginal languages, Pitjantjatjara has a relatively simple set of consonant phonemes (Table 1). Five places of articulation are used: bilabial, alveolar, retroflex, palatal and velar, with a single plosive and nasal phoneme at each place. There are three lateral phonemes, and three approximants, plus a trill or flap alveolar /r/ sound. There are no fricatives. As in other Aboriginal languages, there is no phonemic distinction between voiced and voiceless consonants.

	bilabial	alveolar	retroflex	palatal	velar
plosive	[p] *p*	[t] *t*	[ʈ] *ṯ*	[c] *tj*	[k] *k*
nasal	[m] *m*	[n] *n*	[ɳ] *ṉ*	[ɲ] *ny*	[ŋ] *ng*
trill/flap		[r] *r*			
lateral		[l] *l*	[ɭ] *ḷ*	[ʎ] *ly*	
approximant	[w] *w*		[ɻ] *ṛ*	[j] *y*	

Table 1: Pitjantjatjara consonants: phonetic symbol in [ ] and standard orthography in *italics*.

Pitjantjatjara has six vowel phonemes, long and short [i], [u] and [a]. In the orthography, long vowels are doubled, *ii*, *uu*, *aa*. Syllables are highly constrained: all syllables are (C)V(C): there are no consonant clusters except across syllable boundaries. Most word-final syllables are CV. Word stress in Pitjantjatjara is quite regular, always on the first syllable, with subsequent syllables receiving equal prominence.

3.2. Choosing a BL for faking Pitjantjatjara

Perhaps the most difficult part of the experiment is to choose the best BL for our fake Pitjantjatjara TTS system. Nowadays, as mentioned in the first paragraph, developers of TTS systems are marketing more and more versions, often of surprisingly good quality. Practically speaking, we cannot test all of them on native speakers, so we need to have some criteria to enable us to narrow down the field. Three factors, possibly conflicting, will guide our decision: phoneme sets, comparative prosody, and orthographic mapping.

3.2.1. Phoneme sets

Perhaps the most obvious requirement is to find a BL which as nearly as possible covers the same set of phonemes as the TL. There are two sides to this problem: (a) when the TL phoneme has no equivalent in the BL, or (b) when the 'equivalent' phoneme is significantly different.

The first case may not be as important as it may seem: Pitjantjatjara for example has retroflex and palatal consonants which we may not find in any of the candidate BLs. But we can find a way around this: *sequences* of BL phonemes may sound sufficiently like the target phonemes, e.g. <rd> for [ɻ]. Alternatively, if the BL is forced to conflate a phonemic distinction, this may result only in the synthetic speech 'having an accent', rather than rendering it incomprehensible.

Possibly more damaging will be the second scenario. Thinking again of the 'r'-like sounds in Pitjantjatjara [r ɻ] plus the retroflex sounds [ʈ ɳ ɭ], BLs such as French and Portuguese which have an /r/ phoneme realised as a uvular fricative are probably not going to be suitable. Vowel

phonemes could be a significant problem for other applications, though fortunately for us the Pitjantjatjara vowel system is about as simple as any found in all the languages of the world.

3.2.2. Prosody

Prosody plays a big part in how realistic a TTS system is judged to be, and developers of TTS systems work very hard to get this right. Even when given 'gibberish' to read, good TTS systems will do so with the appropriate intonation and accent, which can be very distinctive. The typical prosody for Pitjantjatjara is rather flat, so any BL with a wide-ranging prosody, such as Swedish and Danish, may well be ruled out. The stress patterns of Pitjantjatjara are also very simple, with word-initial stress, and evenly stressed syllables. This makes languages like English, which almost never has word-initial stress on long words, quite unsuitable. Intuitively, of the BLs available, Basque turns out to be a better bet, with its flat intonation and syllable-timed rhythm, despite the fact that Basque words are said to be stressed word-finally.

3.2.3. Text-to-phoneme mapping

Against the phoneme and prosody match is the question of the built-in text-to-phoneme mapping rules. Again there are two sides to this: How are the individual letters in the BL pronounced? And what is the likelihood of TL *words* being the same as BL words, but with a different pronunciation?

Since Pitjantjatjara orthography is based on English, there is in general a good text-to-phoneme mapping between English and Pitjantjatjara. The writing system is also largely 'phonetic', by which we mean that there is a simple mapping between letters and phonemes. Thinking of other BL–TL pairings, typical 'problem letters' are *c*, *g*, *j*, *v*, *w*, *x*, *y* and *z*, and all vowels, plus digraphs. This problem is minimised for Pitjantjatjara, since only 12 consonant letters and 3 vowel letters are used. The writing system has 4 digraphs (*tj*, *ny*, *ly*, *ng*), the 4 underlined letters representing retroflex sounds (*t*, *n*, *l*, *r*), plus the three long vowel digraphs. None of the digraphs found in English (such as *ch*, *sh*, *th*) appear in (native) Pitjantjatjara words.

Bearing in mind the comments relating to prosody, if we choose Basque as the BL, we have to consider *its* orthography. Fortunately, Basque too has only recently had its orthography standardized (in 1964), and so we find that the letter-to-phoneme mappings are quite straightforward. Table 2 shows the mapping between Pitjantjatjara phonemes and Basque spelling. Note that the letter *y* has to be avoided (it only occurs in borrowings in Basque, when it has its Spanish value [i]). The palatal sounds are well catered for in Basque: the palatal stop, written <tt>, is used in diminutives and childish forms, while Spanish <ñ> and <ll> are needed for Spanish borrowings, and so give us palatal *ny* and *ly*.

p <p>	*t* <t>	*t̲* <rt>	*tj* <tt>	*k* <k>
m <m>	*n* <n>	*n̲* <rn>	*ny* <ñ>	*ng* <ng>
	r <rr>			
	l	*l̲* <rl>	*ly* <ll>	
w <w>		*r̲* <r>	*y* <i>	

Table 2: Mapping from Pitjantjatjara orthography in *italics* onto Basque 'letters' in<angle brackets>.

3.3. Implementation

All the transliterations identified in Table 2 can be applied to Pitjantjatjara text by a simple string-substitution program. Examples (1) and (2) show Pitjantjatjara texts in standard orthography (a) and as they are input to the Basque TTS (b).

(1) a. *Tjitji nyanga kati atjupitilakutu.*
b. ttitti ñanga kati attupitilakutu.
'Take this child to the hospital.'

(2) a. *Yaaltjitu aranа tjikinma?*
b. iaalttitu ararna ttikinma?
'How many times should I drink it?

4. Evaluation

Evans et al. (2002) describe a modest evaluation of a prototype Greek synthesizer using both English and Spanish as the BL with just three native Greek speakers who were also fluent English speakers. A first evaluation involved a variation of the Modified Rhyme Test (House et al. 1965, Miner and Danhauer, 1976, Logan et al. 1989, Goldstein 1995) in which subjects must match from a list of five options the word which they think they have heard. The subjects underwent 15 tests, with 5 words in each test. Evans et al. report that "with the Spanish synthesiser … there were a total of 7 errors in 45 trials", a 97% success rate.[7] With the English synthesizer there were 10 errors (96%). A second evaluation involved the use of nonsense words, with much lower recognition rates (50% for both systems). In a third evaluation, the subjects were tested with a number of complete sentences, including "common simple sentences and a small number of 'tongue-twisters'". Correct identification in this case was 100%, though it is unclear whether subjects had to identify what they heard,

[7] It is not explicitly stated, but Evans (personal communication) confirms that the 45 trials were 15 tests × 3 subjects. Since each test had 5 items, the total number of items is 225.

or choose from a number of alternatives.

Evans et al. readily admit the shortcomings in this evaluation, mentioning the small sample size, and the fact that all the subjects were fluent English speakers, long-term UK residents, therefore familiar with English phonemes and prosody. In addition, they note that the tests did not reflect the actual intended use of the software, in their case as a screen reader.

Our evaluation attempts to bypass these shortcomings: we aim to recruit sufficient subjects to make the results statistically significant; English (or Basque) language ability will not play a role in the experiment; and experience/exposure to English is more or less constant for all Pitjantjatjara speakers; and, most important of all, we attempt to simulate better the situation in which the software might be used.

4.1. A realistic scenario

Our methodology is based on the method used by Somers and Sugita (2003) in their evaluation of SLT software. SLT research, almost without exception, focuses on one particular type of application, namely task-oriented co-operative dialogues, for example, scheduling a meeting, or arranging travel. Accordingly, Somers and Sugita chose to evaluate the SLT software by translating phrases taken from a tourist's phrasebook. Importantly, they were interested in "the subject's ability to infer correctly the *intended* meaning of the utterance" (emphasis added) rather than the grammar or style of the translation. In a similar manner, we are interested above all in whether the faked output is intelligible, with little interest in naturalness and phonetic accuracy, unless it impinges on intelligibility, in our healthcare scenario.

In our experiment, participants will be told (in their own language by a native speaker experimenter) to imagine that they have gone to the doctor's office with some specific medical problem, say, respiratory difficulties, and that whatever the doctor says is going to be translated and 'spoken' by the computer. They will then be asked to listen to the synthesised speech, and to tell the experimenter what they understood. Because the syntax etc. of the 'translation' is not an issue, it will be acceptable if they simply repeat verbatim what they have heard. The experimenter will make a judgment about whether they have understood, and will ask clarificational questions if necessary. Sessions will be recorded to enable judgments to be corroborated.

As in the Somers and Sugita (2003) experiment, five different sub-scenarios will be presented, each with a contextualisation (e.g. the doctor asks you about your symptoms, the nurse will explain the treatment, the receptionist wants to make a follow-up appointment), with five phrases in each, giving a total of 25 phrases (see Appendix). Sessions should last about 15 minutes. Phrases have been specifically tailored so as to contain essential information which must be repeated by the participant if the experimenter is to judge the phrase to have been understood, e.g. *Take two puffs every four hours*. At the end of each session, the experimenter will elicit any general reactions and opinion from the participants in an informal interview.

If enough subjects can be found, a variant of the experiment will use human recordings (rather than synthetic speech) for a selection of the phrases, to provide a base-line control. Alternatively, it would be interesting to contrast the use of Basque as the BL, with preprocessing to adapt the orthography, against the cheaper but 'dumber' implementation using the free English-based TTS software typically available on most computers (e.g. using the voices of Microsoft Sam and Microsoft Mary).

In our experiments with Somali (Somers et al., 2006), we have tested several different implementations, combining the approach described here with the 'gibbering' approach of Evans et al. (2002), as mentioned above. We have told the subjects that the computer would have several different 'voices', and fitted the scenario around this, with each section associated with a different speaker: a receptionist, the doctor, a pharmacist, and so on. Obviously we need to sample each 'voice' with each section, so the experimental set-up is a Latin square of voices × sections, which determines the minimum number of participants needed. As each different voice is introduced, a sample is introduced and played: for example, the experimenter will say "The doctor says: *I want to get some background details*", and then plays the sample.

5. Pilot studies

At the time of writing we are still hoping to set up an experiment with Pitjantjatjara speakers via a number of agencies. During pilot experiments with Somali speakers (Somers et al., 2006), a number of interesting elements have led us to adjust the experimental design.

The first of these concerns the translations. It will not come as a surprise to find that Pitjantjatjara does not have its own words for some vocabulary items such as *pollution, medical history*. There are also some constructions which are hard to render in Pitjantjatjara, for example comparatives (e.g. *Is it worse at night?*). It might

even turn out that asthma is not an appropriate testbed for that particular locale. The significance of these is that, as we quickly discovered with the Somalis, loanwords were difficult to understand: subjects who otherwise were obtaining good scores for comprehension were baffled by Somali words such as *dhamb* 'damp', *boolushan* 'pollution' and *hayla* 'inhaler'.

A second idea to emerge from the pilot experiments was to adjust our scoring system. We have found that subjects would like to hear each sample two or three times, and they typically 'get' more of the message on each hearing. So we are devising a scoring method which takes this into account, and quantifies not just whether the subjects could understand the synthetic speech, but how many repetitions they need (up to, say, three). In a practical situation it is not unreasonable to ask to hear something again.

Finally, we have noticed that subjects quickly get used to the voice, and seem to have a better level of recognition towards the end of the test. This is something we should try to measure, although the experimental design with multiple 'voices' may mask this effect considerably.

We look forward to reporting results of the experiment in due course.

Acknowledgements

Our thanks go to Andrew Longmire at the Department of Environment and Heritage's Cultural Centre, Uluru–Kata Tjuta National Park, Yulara NT, and to Bill Edwards, of the Unaipon School, University of South Australia, Adelaide, for their interest in the experiment, and, we hope eventually, for their assistance in conducting it.

Appendix – Rubric and test items to be translated into Pitjantjatjara

[The text in italic is spoken by a human – the experimenter – in the participant's own language – text in bold represents the sample in each case]

You are going to hear some sentences spoken by the computer. I want you to tell me in your own words what you understand the computer to be saying. You don't have to answer the questions. I can play them up to three times each. Don't worry if you don't really understand it – it's not you who is being tested!

The computer has different voices. Each time I will play you an example and tell you what it is saying.

Please imagine that you want to see the doctor because you have difficulty with breathing.

1. The receptionist says: ***First we have to make an appointment.***

a. What is your name?
b. Have you been to this clinic before?
c. Can you come next Thursday?
d. How about ten o'clock?
e. Do you want an appointment card?

2. The doctor's assistant says: ***I am going to ask you some questions about your circumstances.***

a. Do you smoke, or have you ever smoked?
b. Do you have any pets?
c. Do you take regular exercise?
d. Does anyone in your family have asthma?
e. Is your home dry?

3. *The doctor says*: ***I want to get some background details***.

a. When did you first have difficulty with breathing?
b. Is your breathing worse at any particular time of day?
c. Does it prevent you from sleeping?
d. Is it worse after strenuous activity?
e. Does it depend on the weather?

4. *The nurse says:* ***I will make some suggestions***.

a. Your asthma may be triggered by the weather.
b. Check your bathroom for mold.
c. Use the brown inhaler if you have a mild attack.
d. Use it once a day for a week after an attack.
e. Take two puffs every four hours.

5. The pharmacist says: ***Your prescription is ready***.

a. Are you taking any other medication?
b. What is your name and address?
c. Take this three times a day.
d. Are you entitled to a free prescription?
e. That will be six dollars fifty please.

References

Eckert, P. and J. Hudson (1991) *Wangka Wiru: A Handbook for the Pitjantjatjara Language Learner*. Underdale, SA: University of South Australia.

Evans, G., K. Polyzoaki and P. Blenkhorn (2002) An approach to producing new languages for talking applications for use by blind people. In K. Miesenberger, J. Klaus and W. Zagler (eds) *8th ICCHP, Computers Helping People with Special Needs*, (LNCS 2398), Berlin: Springer Verlag, pp. 575–582.

Goldstein, M. (1995) Classification of methods used for assessment of text-to-speech systems

according to the demands placed on the listener. *Speech Communication* **16**:225–244.

House, A.S., C.E. Williams, M.H.L. Hecker and K.D. Kryter (1965) Articulation-testing methods: Consonant differentiation with a closed-response set. *Journal of the Acoustic Society of America* **37**:158–166.

Logan, J., B. Greene and D. Pisoni (1989) Segmental intelligibility of synthetic speech produced by rule. *Journal of the Acoustical Society of America* **86**:566–581.

Miner, R. and J.L. Danhauer (1976) Modified rhyme test and synthetic sentence identification test scores of normal and hearing-impaired subjects listening in multitalker noise. *Journal of the American Audiological Society* **2**:61–67.

Narayanan, S., S. Ananthakrishnan, R. Belvin, E. Ettelaie, S. Gandhe, S. Ganjavi, P.G. Georgiou, C.M. Hein, S. Kadambe, K. Knight, D. Marcu, H.E. Neely, N. Srinivasamurthy, D. Traum, and D. Wang (2004) The Transonics spoken dialogue translator: An aid for English-Persian doctor-patient interviews. In: T. Bickmore (ed.), *Dialogue Systems for Health Communication: Papers from the AAAI Fall Symposium*, Menlo Park, CA: The AAAI Press, pp. 97–103.

Rayner, M., P. Bouillon, V. Van Dalsem, H. Isahara, K. Kanzaki and B.A. Hockey (2003) A limited-domain English to Japanese medical speech translator built using REGULUS 2. *Companion Volume, 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 137–140.

Somers, H., G. Evans and Z. Mohamed (2006) Developing speech synthesis for under-resourced languages by 'faking it': An experiment with Somali. Paper submitted to *LREC 2006* (Genoa).

Somers, H. and H. Lovel (2003) Computer-based support for patients with limited English. *Association for Computational Linguistics EACL 2003, 10th Conference of The European Chapter, Proceedings of the 7th International EAMT Workshop on MT and other language technology* tools, Budapest, pp. 41–49.

Somers, H., H. Lovel, M. Johnson and Z. Mohamed (2004) Language technology for patients with limited English. *EACH International Conference on Communication in Healthcare*, Bruges, P04.09.

Somers, H. and Y. Sugita (2003) Evaluating commercial spoken language translation software. *MT Summit IX: Proceedings of the Ninth Machine Translation Summit*, New Orleans, pp. 370–377.

Dual-Type Automatic Speech Recogniser Designs for Spoken Dialogue Systems

Jason Littlefield and Michael Broughton
Command and Control Division,
Defence Science and Technology Organisation
PO Box 1500
Edinburgh, Australia, 5111
Jason.Littlefield@dsto.defence.gov.au
Michael.Broughton@dsto.defence.gov.au

Abstract

A Dual-Type automatic speech recogniser (ASR) is a multi-pass ASR system that incorporates both a speaker-independent (SI) and a speaker-dependent (SD) ASR. The purpose of this approach is to improve the robustness of spoken dialogue systems for a broader range of applications. This paper identifies feasible Dual-Type multi-pass ASR system designs that are intended to overcome limitations arising from the use of a single type of ASR. Implementation issues are also discussed.

1.Introduction

Current implementations of Spoken Dialogue Systems (SDS) are developed around a single ASR. This design limits the overall system's recognition accuracy to the performance of the installed ASR, while the useability is determined by the individual ASR type. ASRs can be categorised into two types, either SD or SI, with each having their own strengths and weaknesses.

SI ASRs have the advantage of not requiring a prior enrolment or customised training session for their end users, thereby allowing any user of a given regional dialect to effectively use the system. These systems rely on an underlying grammar that typically needs to be relatively small, or at least, only have a small portion of the grammar active at any point in time. Due to requiring limited grammar size for optimum recognition accuracy, these systems are often used in a system led interaction, whereby the machine asks questions of the user that elicit simple responses. These responses may be single words, or small continuous strings.

SD ASRs, on the other hand, require training for each individual user. This training is relatively short, generally less than ten minutes, and involves the speaker reading aloud a prepared text, which is analysed by the ASR for generation of the speaker's acoustic model. A speaker profile is created by combining this acoustic model with a vocabulary and a regional language model. SD systems are adaptive and have the advantage of being able to recognise free speech or constrained grammar speech, although extracting semantic content from the free speech is more difficult than with a formalised grammar.

In addition to the fundamental differences between the two ASR types, there are also individual differences within each ASR type. The various commercial and research implementations are developed from different algorithms and techniques, typically providing varying output for the same paragraph of spoken text.

ASRs are also further classified by their speech continuity as well as grammar and vocabulary size. Speech continuity describes whether words are spoken in isolation, as connected speech or as continuous speech (Zue et al., 1997). Connected speech ASRs require pauses between multiple word phrases, whereas continuous speech ASRs do not. The grammar and vocabulary size refers to the number of phrases and words that can be spoken and recognized. A grammar can be characterised by the number of plausible alternatives (perplexity), the number of rules and the number of words (Gibbon et al., 1997).

A prototype Multimodal Dialogue System, incorporating an SDS, has been developed for the Future Operations Centre Analysis Laboratory (FOCAL) at Australia's Defence Science and Technology Organisation (DSTO). FOCAL is a collaborative environment that is exploring new paradigms for situation awareness and command and control in military command centres (Wark et al. 2004). An SDS was initially implemented for FOCAL to enable natural dialogue with its Virtual Advisers (Broughton et al. 2002) using an SD ASR and later using an SI ASR (refer section 3). These Virtual Advisers are real-time animated talking heads that can deliver briefs or be queried for additional information. Figure 1 shows some of FOCAL's Virtual Advisers on the main display during an interactive briefing session.

Proceedings of the Australasian Language Technology Workshop 2005, pages 78–86,
Sydney, Australia, December 2005.

SDSs rely on accurate speech-to-text transcription (spoken utterance decoding) from their ASR to perform well. State-of-the-art ASRs perform optimally in quiet environments but are sensitive to interference from ambient noise, overlapping speech and reverberation (Littlefield et al., 2002). Due to the 3.6 metre radius 150° spherical screen, the reverberation characteristics of FOCAL are less than ideal, particularly near the focal point of the screen. This causes degradation in performance of ASRs used in this environment.

Figure 1: Photograph of FOCAL with screen.

To overcome these limitations, we are interested in the development of multi-pass systems, those requiring two or more ASR engines to improve robustness and overcome deficiencies in single ASR based SDSs. The ASR engines can either be of the same or different type, with the overall aim of improving recognition accuracy in a broader range of applications, by utilising the best features of each ASR in the SDS system. An example of an existing multi-pass system is SpeechMAX™ (Custom Speech USA, 2005), a dual-engine system that utilises two SD ASRs, in this case Scansoft's Dragon NaturallySpeaking and IBM's ViaVoice (ScanSoft, 2005). Pellom and Hacioglu (2003) incorporated two passes in the University of Colorado's SONIC ASR to improve robustness in noisy environments. Furthermore, Pérez-Piñar López and García Mateo (2005) use a multiple-pass ASR system where the ASRs have language models adapted from distinct topics.

We are interested in a new area of research that incorporates a Dual-Type ASR to improve SDS robustness. A *Dual-Type ASR* is a multi-pass ASR that incorporates both a SI and a SD ASR. As discussed, SD and SI ASRs have differing advantages and disadvantages to each other, and the aim of this proposed research is to exploit the benefits of these systems to improve recognition accuracy in situations that would normally be detrimental to these systems if used in a traditional single-pass design. Hockey et al. (2003) have developed an SDS that uses two ASRs, a grammar-based SI primary ASR a Statistical Language Model ASR

Section 2 introduces components of an SDS while section 3 describes the past and present SDS in FOCAL. Components of a Dual-Type ASR are identified and explained in section 4. Section 5 describes the alternative designs for a Dual-Type ASR and issues common to all the designs are examined in section 6. Section 7 describes future implementation and experimentation. Finally, the conclusion is provided in section 8.

2.Components of an SDS

The components of an SDS include a microphone, an ASR, a grammar and a Dialogue Manager.

The *microphone* physical design, directionality, frequency response and electrical output are characteristics that help describe different microphone types and aid in the correct microphone selection for specific applications. The microphones that are used in the FOCAL environment include analogue and digital super-cardioid headset microphones, and analogue super-cardioid gooseneck microphones.

An *ASR* decodes audio from spoken utterances into one or more recognition results in the form of text. By default ASRs often display only one speech-to-text interpretation, the most probable interpretation. However, ASRs can produce a list of alternative interpretations, each with a confidence score expressed as a probability or percentage. It is useful to use more than one interpretation in an SDS when another component, such as the Dialogue Manager, has more contextual information than the ASR to select the most likely recognition result.

A speech recognition *grammar* is a list of rules and symbols that can be spoken and recognised by an ASR, often represented as a context free grammar (CFG). The format of the CFG used by an ASR is usually a standard format, such as Nuance Grammar Specification Language (GSL) (Nuance, 2001) or Java Speech Grammar Format (JSGF) (Sun Microsystems, 1998).

The *Dialogue Manager* controls the flow of dialogue with the user and coordinates system responses. The Dialogue Manager, as implemented within FOCAL, can receive one or more recognition results from the ASR system. The additional recognition results are compared within the current dialogue context to improve likelihood of correct recognition.

3.FOCAL's Current SDS

FOCAL's initial SDS (Broughton et al. 2002) was developed around the SD ASR Dragon NaturallySpeaking™, chosen because of its high recognition accuracy, availability and developer support. Additional software for natural language understanding and dialogue management was developed using Natlink (Gould, 2001). Natlink enabled the development of macros and grammars for Dragon NaturallySpeaking. This initial concept system demonstrated the ability to interact with FOCAL's Virtual Advisers. However, the major limiting factor of SD ASRs meant that only those trained with the ASR could use the system. More sophisticated grammars also needed to be implemented to enable scalability of the system.

To address these issues, a second SDS was developed using a SI ASR and a more sophisticated Dialogue Manager based on an agent-based architecture (Estival et al., 2003). In this system, Nuance 8.0 (Nuance, 2001) was chosen as the SI ASR as it provided high reliability, a developer's toolkit, and an Australian-New Zealand acoustic language model. Regulus (Rayner et al., 2001, Regulus, 2005) was incorporated for language processing, enabling the development of typed unification grammars and their compiling into Nuance compatible context-free grammar language models. The agent-based dialogue management system was incorporated into the larger FOCAL agent architecture (Wark et al., 2004) to enable broader application within FOCAL. Currently this system has been implemented to enable users to dialogue with the Virtual Advisers during their presentation of a brief. It enables any one of four Virtual Advisers to be asked questions relevant to their presented information.

The SI ASR in FOCAL's current SDS has two functions. Firstly, it detects, records and saves the audio from spoken utterances as wavefiles. Secondly, it decodes the audio input from the spoken utterance into recognition results. The recognition results are a set of text strings that most closely match rules in the ASR's small CFG.

The Queensland University of Technology Universal Background Model (QUT-UBM) Speaker Identification System (SID) (Pelecanos and Sridharan, 2001) which recognises a person from the sound of their voice, has also been integrated into FOCAL. The SID performs acoustic analysis of the audio from a spoken utterance and tries to match the pattern with that of a trained target user model. The system's response is either the name of the matched target user model or "unknown".

In addition to our current SDS with the Virtual Advisers, we are also exploring multimodal input with an immersive geospatial application (Wark et al., 2005). This system builds on our current SDS, to enable deictic referencing from pointing devices.

4.Components of a Dual-Type ASR

The components of a Dual-Type ASR include a microphone, spoken Utterance Recorder, speech recognition grammar, SI ASR, SD ASR coupled with an SID, and recognition result Error Detector and Reconciler. Configurations of these components are described in section 5.

The *Utterance Recorder* is used to detect, record and save the audio from spoken utterances as wavefiles. Although ASRs are capable of recording spoken utterances, we propose that the use of an independent spoken utterance recorder will lead to a more scalable and flexible system. This is important because more than one of the components requires the audio from spoken utterances at the same time. However, this incurs a delay, since the ASRs cannot begin to decode a spoken utterance until that utterance has finished and has been saved as a wavefile. It takes roughly as long as the duration of an utterance to decode an utterance from a wavefile.

Because there is an independent Utterance Recorder, the *SI ASR* is only required to decode the audio input from spoken utterances into recognition results.

The *SD ASR* also decodes the audio input from the spoken utterance into a set of recognition results. However, because this ASR has a more accurate model of a speaker's voice pattern than the SI ASR, it can use larger grammars. Hence, SD ASRs can operate in at least two different modes: large vocabulary continuous speech (dictation mode) or small vocabulary connected speech (command mode). The dictation mode uses a large vocabulary of 20000 words or more (Zue et al., 1997). The command mode employs a user-defined CFG in a standard format.

Since the SD ASR needs to know the speaker's identity, we couple a *SID* system with the SD ASR in an attempt to automate this process.

The *Error Detector* will select the best recognition result interpretations, measure agreement between the best interpretations, and identify erroneous segments of interpretations.

The recognition results from each ASR include an ordered list of possible interpretations within the grammar, with each interpretation having a confidence score associated with it. The best interpretations will be selected by examining the confidence scores and choosing those above a

predefined threshold.

The interpretation with the highest confidence score from each ASR will be compared for agreement. The assumption here is that if the ASRs produce the same recognition result and this recognition result receives a high confidence score, then it is likely to be correct. In this case a second ASR reinforces the best result of the first ASR. This comparison will be accomplished using Sclite (NIST, 2001), a software tool from the US National Institute of Standards and Technology, that provides word error rate between the two strings, a reference and a hypothesis. If the word error rate is zero, the Error Detector will flag agreement. Since there is only one recognition result in this case, the Reconciler is not required, and is bypassed. If the word error rate is greater than zero, the recognition results are aligned and compared again using Sclite. Sclite aligns the strings and identifies substitution, insertion and deletion misalignments. Part of an example report from Sclite for insertion, substitution and deletion misalignments follows.

```
REF: the brown ** fox JUMPED over THE lazy dog
HYP: the brown IN fox LUMPED over *** lazy dog
Eval:          I      S           D
```

Note that the reference (REF) is only the best recognition result based in confidence scores, not necessarily a correct recognition result. Hence, the substitution, deletion and insertion misalignments are only possible sources of errors.

The degree of agreement (word error rate produced by Sclite) and the location and type of misalignments will be passed on to the Dialogue Manager which will lead to a response to query the user about the error. The best recognition results will be passed on to the Reconciler to process.

It is expected that the Error Detector will require minimal processing for smaller grammars due to the high recognition accuracy achievable with them. The high recognition accuracy will provide identical outputs from both the SD and SI systems and therefore minimal work for the error detection system. As the grammars become more complex however, variation between the two ASRs is expected and providing the correct output in this situation is one of the aims of this research.

The *Reconciler* will receive a set of recognition results from more than one ASR and produce the most probable interpretation. The Reconciler will use an existing system in the speech and language technology domain that makes a selection from multiple output strings. Multi-engine machine translation systems require a component similar to the recognition result Reconciler presented here. DEMOCRAT is an example of such a component for deciding between multiple outputs created by automatic translation (van Zaanen and Somers, 2005).

The best two or three interpretations from each of the ASRs will be sorted in order of confidence and passed on to DEMOCRAT. However, the relationship between the confidence scores from one ASR to another is unknown. The Reconciler will need to take this into account when selecting the best candidate interpretations. DEMOCRAT will produce a consensus interpretation by taking the best segments of each interpretation (van Zaanen and Somers, 2005).

5.Proposed Dual-Type ASR Designs

The following Dual-Type ASR designs we have proposed incorporate one or more ASR to decode the audio input from spoken utterances into recognition results. Each iteration through an ASR is a recognition pass, and therefore, a design using one ASR is a single-pass system, and a design using two ASRs is a two-pass system and so on.

The four proposed Dual-Type ASR designs are:

1. Single-pass ASR
2. Two-pass ASR in parallel
3. Two-pass ASR in parallel with error detection
4. Three-pass ASR in parallel with error detection.

The first Dual-Type ASR system design being proposed is a single-pass ASR which includes a Utterance Recorder followed by a SID system where the speaker's identity is decoded. This design is illustrated in figure 2.

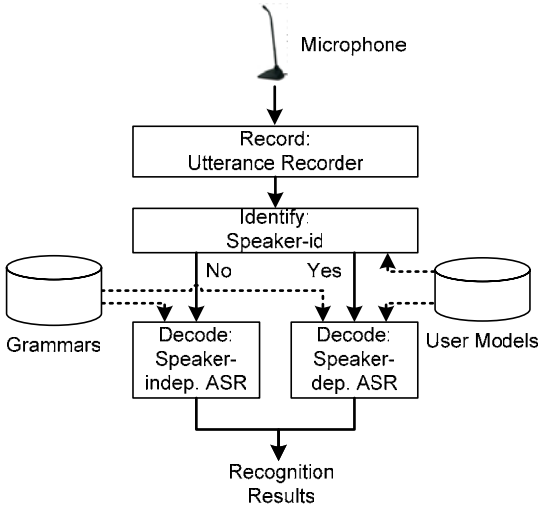


Figure 2: System design of the Dual-Type single-pass ASR.

If the speaker is identified, then the wavefile for the utterance is passed to the SD ASR for decoding into a recognition result. If the speaker is not identified then the wavefile for the utterance is

passed to the SI ASR for decoding. This assumes that the SD ASR is at least as accurate as the SI ASR for large vocabularies as referred to by Merino (2001). This system does not require a Reconciler or Error Detector component.

The second system design, shown in figure 3, proposes a two-pass ASR in parallel where two ASRs decode all spoken utterances concurrently. A spoken utterance recorder detects and records utterances as wavefiles, which are then decoded simultaneously using a SI ASR and a SD ASR incorporating a SID system. The Reconciler compares the recognition results and provides a reconciled result for the SDS Dialogue Manager.

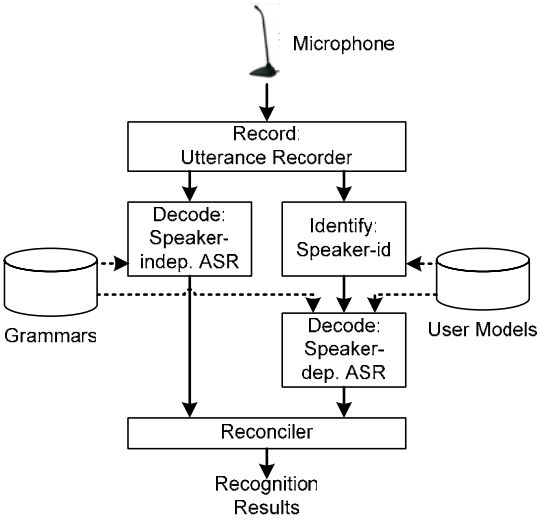


Figure 3: System design for the Dual-Type two-pass ASR in parallel.

The third Dual-Type ASR system design being proposed is an extension of the second system. It is a two-pass system, where two ASRs decode all spoken utterances in parallel, with the addition of an Error Detector. In an effort to be more efficient, a first-pass using a SI ASR will be used every time, whereas the second-pass using a SD ASR will be used only if the Error Detector decides it is required.

As before, a spoken utterance recorder detects and records utterances as wavefiles and the spoken utterances are decoded by the SI ASR and the SD ASR incorporating a SID system. However, the SI ASR recognition result is assessed for errors. If an error is detected, then the result is passed to the Reconciler and compared to the SD ASR recognition result. The Reconciler then passes a reconciled result to the SDS Dialogue Manager. If no errors are found, then the Reconciler is bypassed, and the result from the SI ASR is passed on directly to the SDS Dialogue Manager. Figure 4 illustrates the Dual-Type two-pass ASR in Parallel with Error Detection system design.

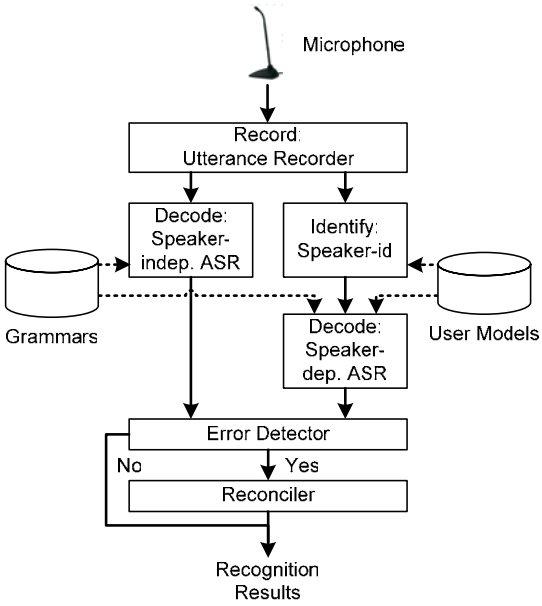


Figure 4: System design of the Dual-Type two-pass ASR in Parallel with Error Detection.

The last proposed design shown in figure 5 incorporates three recognition passes. As in proposal 3 (figure 4), the SI and SD ASR will be used in parallel with error detection. The third pass in this proposal is another SD ASR in dictation mode without a constrained grammar. This would be a useful approach in situations where there are out of vocabulary errors using constrained grammars. The SD ASR in dictation mode has a much larger vocabulary, which could help overcome out of vocabulary errors with constrained grammars and potentially provide a more accurate recognition result.

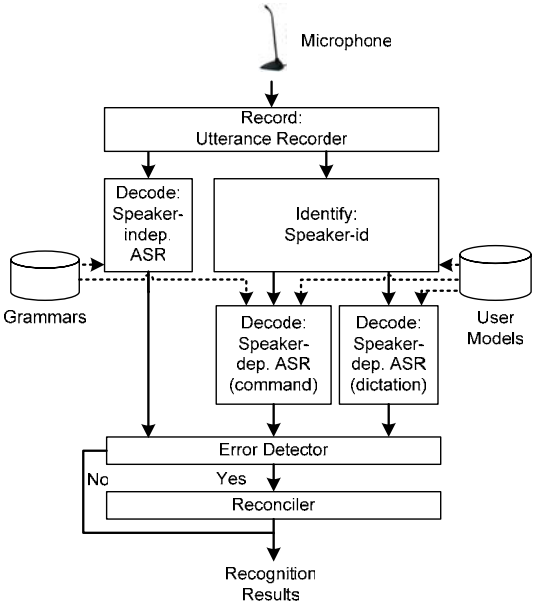


Figure 5: System design of the Dual-Type three-pass ASR in Parallel with Error Detection.

6.Pro and Cons of Each Proposed Design

The speed, accuracy and complexity of each proposed design and its effect on SDS robustness will be compared to determine the most promising approach. A breakdown of the time delay overall is discussed in section 7.1. In the single-speaker situation, the speed of each proposed Dual-Type ASR is estimated to be $2t + 2$ seconds, where t is the length of the spoken utterance in seconds. This assumes the Error Detector and Reconciler incur a negligible time delay.

The accuracy and robustness of each proposed design will be determined through experimentation. Proposed designs 3 and 4 are expected to perform better due to the use of the Error Detector component. This is due to the agreement of recognition results with high confidence scores between ASRs. Also, the additional alignment data enables the Dialogue Manager to query the user when conflicting recognition results occur. That is the ability to query the user for clarification of an utterance segment when required.

The complexity of each of the proposed designs can be described in terms of the number of ASR passes and the number and type of required components. Table 1 shows these terms for each of the proposed designs, 1 though 4. The more complex the design, the more effort required to build and maintain.

Design	No. of Passes	SI ASR	SID	SD ASR	Rec.	Err. Det.
1	1	✓	✓	✓	×	×
2	2	✓	✓	✓	✓	×
3	2	✓	✓	✓	✓	✓
4	3	✓	✓	✓	✓	✓

Table 1: Required components for each proposed Dual-Type design.

7.Foreseeable Design Issues

Before implementing these proposed designs for experimentation, there are some design issues that need to be considered.

7.1.Time Delay

The time delay between the end of spoken utterance and the SDS executing an action is crucial to user satisfaction. A brief investigation was conducted into the duration of spoken utterances and SDS response times in FOCAL. Short sentences, such as 'Yes' or 'No', were about 0.5 second long, while the longest sentence was about 4 seconds long. The corresponding SDS response times were estimated to be between 5 and 10 seconds depending on the complexity of the sentence and resulting action.

The proposed Dual-Type ASR designs incur further time delay. The Utterance Recorder takes the duration of the spoken utterance, which is between 0.5 and 4 seconds, to detect, record and save an utterance. The SID system takes about 2 seconds to identify speakers, while both types of ASRs take about the length of the utterance to produce recognition results. However, if the SD ASR needs to load a different speaker profile, there is an additional delay. Dragon NaturallySpeaking 8 takes about 6 seconds to do this. The recognition result Reconciler and Error Detector have not been implemented yet, however for small grammars their duration is expected to be negligible.

Hence, for spoken utterances of between 0.5 and 4 seconds the estimated overall delay for an SDS incorporating a proposed Dual-Type ASR design will be between 8 and 26 seconds. Note that the Dual-Type ASR is responsible for between 3 and 16s of this estimate. In a multi-speaker environment, the SD ASR will only need to load a speaker profile if the speaker changes. This is not the case all the time. Table 2 shows the breakdown of estimated time delay for 0.5 and 4 second long utterances with and without a change in speaker. The time delay will be measured in future experimentation.

Utt & UR	***SID***	***SD. ASR***		***DM***	***Total***
		Loading	***Transcribing***		
0.5s	2s	-	0.5s	5s	**8s**
4s	2s	-	4s	10s	**20s**
0.5s	2s	6s	0.5s	5s	**14s**
4s	2s	6s	4s	10s	**26s**

Table 2: Estimated time delay for responses with an SDS incorporating a proposed Dual-Type ASR.

The breakdown includes the utterance duration (Utt.), the utterance recorder (UR), the SID system, the SD ASR (loading and transcribing) the Dialogue Manager (DM) and the total.

7.2.Speaker Identification Accuracy

A preliminary trial was conducted by Zschorn (2005) testing a SID system across different spoken utterance lengths. The results demonstrated that for utterance lengths of 0.5, 2.0, 4.0 and 8.0 seconds, the error rates were 57%, 18%, 10% and 6% respectively. Initial investigations into the length of typical spoken utterances using a question-answer (QA) SDS were between 0.5 and 4 seconds. Hence, the accuracy of the SID for very short utterances is expected to be

poor. Both the length of spoken utterances and SID error rates will be measured in future experimentation.

7.3.Grammar Compatibility

In a system where SD and SI ASRs are used in parallel, a single grammar format would be ideal. However there are many different grammar formats. SI ASRs such as Nuance uses GSL and Sphinx 4 uses JSGF. SD ASRs such as Dragon NaturallySpeaking use the Microsoft Speech API 4 (SAPI 4) BNF grammar format and Microsoft's Speech Recognition Engine (MSRE) 5.1 uses GRXML.

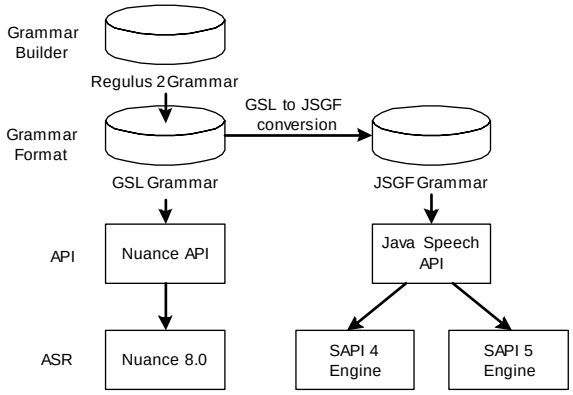


Figure 6: Grammar format and APIs for leading commercial ASRs.

The grammars for the SDS in FOCAL are generated using a grammar building tool called Regulus (Rayner et al., 2001). Regulus can build grammars in GSL format for Nuance. For SD ASRs, any SAPI 4 or SAPI 5 compliant ASR can use JSGF via the Java Speech API (JSAPI), including Dragon NaturallySpeaking, IBM ViaVoice and Microsoft's SAPI 5.1 engine. GSL and JSGF are commonly used standard grammar formats endorsed by World Wide Web Consortium (W3C) in the VoiceXML 2.0 specification. Figure 6 illustrates the relationship between grammar format, API and ASR engines. A GSL to JSGF grammar conversion tool would simplify integration of a Dual-Type ASR. This will be the topic of a Summer Vacation student project at DSTO during the summer 05-06.

8.Implementation

Development of a Dual-Type ASR and integrating it with the SDS in the FOCAL agent-based architecture has already begun. The components of the Dual-Type ASR will be integrated in the current agent-based framework (Estival et al., 2003) so that each of the proposed designs can be tested for experimentation. An agent will be created for the Utterance Recorder, SID system, SD ASR and SI ASR. These agents will interact via a Dual-Type ASR Speech Input agent that will direct data as required. The Dual-Type ASR Speech Input agent will also handle the functions of the recognition results Error Detector and Reconciler as required and interact with the existing Multimodal Input Processor (MIP) in the SDS. The MIP fuses input from multiple modalities and forwards this to the Dialogue Manager (DM). Figure 7 shows the overall design of the Dual-Type ASR and the SDS.

The Utterance Recorder agent will detect spoken utterances, record begin and end timestamps and saves the audio as a wavefile independently.

The SID agent uses the QUT-UBM SID system to decode the identity of the speaker.

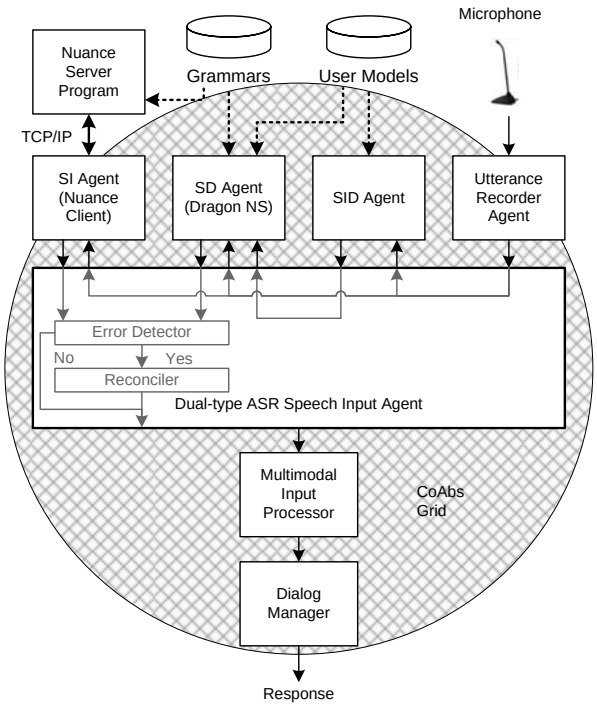


Figure 7: Proposed Dual-Type ASR Speech Input agent with independent Utterance Recorder.

The SD agent will return recognition results. The SD agent consists of the JSGF Grammar and a SD ASR such as Dragon NaturallySpeaking, IBM ViaVoice or MSRE 5.1 integrated using CloudGarden's TalkingJava JSAPI (CloudGarden, 2005). In a multi-speaker environment, the time delay incurred by the SD ASR switching user profiles can be eliminated by using one computer per participant, each with a SD agent. The SI agent consists of the GSL Grammar and Nuance Client. The Nuance Client communicates with the Nuance Server via TCP/IP. This implementation will allow each of the proposed designs will be tested by modifying the routing procedure within the Dual-Type ASR Speech Input agent.

9.Conclusion

In this paper we have presented four alternate designs for a Dual-Type ASR, a system that combines both SI and SD ASRs. The motivation is to provide a more robust SDS system than is currently achievable with a single ASR of either type. We aim to achieve improved robustness through the provision of alternate recognition results from different types of ASR. The designs enable improved user flexibility over a SD system by also providing a SI alternative.

The final design of the Dual-Type ASR system may incorporate several of the proposed designs to maximise the available advantages. These will be reported after the planned development and evaluation of these initial designs.

Acknowledgements

We wish to thank Dr Dominique Estival for her continued guidance and Andrew Zschorn for his consistent contribution to spoken dialogue systems. We would also like to thank Nuance Communications for providing Research membership support for their products.

References

Broughton, M., Carr, O., Taplin, P., Estival, D., Wark, S. and Lambert, D. 2002. Conversing with Franco, FOCAL's Virtual Adviser. *Proc. Virtual Conversational Characters (VCC) Workshop, Human Factors Conference (HF2002)*, Melbourne, Australia.

CloudGarden. 2005. TalkingJava SDK with Java Speech API implementation, http://www.cloudgarden.com/JSAPI/, last accessed 2 November 2005.

Custom Speech USA.. 2005. SpeechMax, http://www.customspeechusa.com/, last accessed 20 August 2005.

Estival, D., Broughton, M., Zschorn, A. and Pronger, E. 2003. *Spoken Dialogue for Virtual Advisers in a semi-immersive Command and Control environment*, 4th SIGdial Workshop on Discourse and Dialogue, Sapporo, Japan.

Gibbon, D., Moore, R. and Winski, R. 1997. *Handbook of standards and resources for spoken language systems*, Walter de Gruyter & Co., pp. 839-852 (Glossary).

Gould, J. 2001. Implementation and Acceptance of NatLink, a Python-Based Macro System for Dragon NaturallySpeaking. The *Ninth International Python Conference*, March 5-8, California.

Hockey, B. A., Lemon, O., Campana, E., Hiatt, L., Aist, G., Hieronymus, J., Gruenstein, A., and Dowding, J. 2003. Targeted help for spoken dialogue systems: intelligent feedback improves naive users' performance. In *Proceedings of the Tenth Conference on European Chapter of the Association For Computational Linguistics - Volume 1*, Budapest, Hungary, April 12 - 17, 2003, European Chapter Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, pp. 147-154.

Littlefield, J. and Hashemi-Sakhtsari, A. 2002. *The Effects of Background Noise on the Performance of an Automatic Speech Recogniser*, Research Report DSTO-RR-0248, Defence Science & Technology Organisation.

Merino, D. 2001. Speaker Compensation in Automatic Speech Recognition. In J.-C. Junqua and G. van Noord (Eds.), *Robustness in Languages and Speech Technology*, pp. 47-100. Telefónica. Netherlands.

NIST. 2001. NIST *sclite* version 2.2, part of Speech Recognition Scoring Toolkit (SCTK) version 1.2, http://www.nist.gov/speech/tools/, last accessed 11 August 2005.

Nuance. 2001. Nuance Speech Recognition System, Version 8.0: Introduction to the Nuance System, Nuance Communications, Inc.

Pelecanos, J. and Sridharan, S. 2001. Feature Warping for Robust Speaker Verification, in *Proc. ISCA Workshop on Speaker Recognition – 2001: A Speaker Odyssey*, June 2001.

Pellom, B. and Hacioglu, K. 2003. Recent improvements in the CU SONIC ASR system for noisy speech: The SPINE task, *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003, pp.14-17.

Pérez-Piñar López, D. and García Mateo, C. 2005. Application of confidence measures for dialogue systems through the use of parallel speech recognizers, In *Interspeech 2005*, pp.2785-2788.

Rayner, M., Dowding, J., Hockey, B.A. 2001. A Baseline Method for Compiling Typed Unification Grammars into Context Free Language Models, *in Proc. of Eurospeech 2001*, Aalborg, Denmark.

Regulus. 2005. Sourceforge Project: Regulus, https://sourceforge.net/projects /regulus/, last accessed 11August 2005.

ScanSoft. 2005. Dragon NaturallySpeaking and IBM ViaVoice, http://www.scansoft.com/, last accessed 5 September 2005.

Sun Microsystems. 1998. Grammar Format Specification, http://java.sun.com/, last accessed 10 September 2005.

van Zaanen, M. and Somers, H. 2005. DEMOCRAT: Deciding between Multiple Outputs Created by Automatic Translation, To appear *in Proc. of 10th Machine Translation Summit*, Phuket, Thailand.

Wark, S., Broughton, M., Nowina-Krowicki, M., Zschorn, A., Coleman, M., Taplin, P. and Estival, D. 2005. The FOCAL Point - Multimodal Dialogue with Virtual Geospatial Displays, *in Proc. SimTecT 2005*, Sydney, Australia.

Wark, S., Zschorn, A., Broughton, M., and Lambert, D. (2004) *FOCAL: A Collaborative Multimodal Multimedia Display Environment. Proc. SimTecT 2004.*

Zschorn, A. 2005. Speaker Identification Test Results, *DSTO informal report.*

Zue, V., Cole, R.A. and Ward, W. 1997. Spoken Language Input: 1.2 Speech Recognition, In Cole, R.A., Mariani, J., Uszkoreit, H., Zaenen, A. and Zue, V. editors. *Survey of the State of the Art in Human Language Technology*, Cambridge University Press, Cambridge, UK, 1997.

Phillips, S. and Rogers, A. 1999. Parallel Speech Recognition, *International Journal of Parallel Programming*, Volume 27, Issue 4, Aug 1999, pp. 257 – 288.

Efficient Knowledge Acquisition for Extracting Temporal Relations

Son Bao Pham and Achim Hoffmann
School of Computer Science and Engineering
University of New South Wales, Australia
{sonp,achim}@cse.unsw.edu.au

Abstract

We present KAFTIE – an incremental knowledge acquisition framework which utilizes expert knowledge to build high quality knowledge base annotators. Using KAFTIE, a knowledge base was built based on a small data set that outperforms machine learning algorithms trained on a much bigger data set for the task of recognizing temporal relations. In particular, this can be incorporated to bootstrap the process of labeling data for domains where annotated data is not available. Furthermore, we demonstrate how machine learning can be utilized to reduce the knowledge acquisition effort.

1 Introduction

Recent years have seen growing interests in temporal processing for many practical NLP applications. For example, question answering tasks try to find when and how long an event occurs or what events occur after a particular event.

Several works have addressed temporal processing: identification and normalization of time expressions (Mani and Wilson, 2000), time stamping of event clauses (Filatova and Hovy, 2001), temporally ordering of events (Mani et al., 2003), recognizing time-event relations in TimeML (Boguraev and Ando, 2005). At a higher level, these temporal expressions and their relations are essential for the task of reasoning about time, for example, to find contradictory information (Fikes et al., 2003).

In this emerging domain, there is a clear lack of a large annotated corpus to build machine learning classifiers for detecting temporal relations. We pursued the idea that an incremental knowledge acquisition approach could be used to develop a knowledge base of rules that utilizes experts' knowledge to overcome the paucity of annotated data. In fact, this approach could be combined nicely with the process of annotating data. When a new piece of data is annotated differently to what an existing KB proposes, the annotator specifies a justification for the decision in the form of a rule which is added to the knowledge base. Our experience shows that the time it takes to formulate a rule explaining why the data is annotated in a certain form is not much if the users have already spent time on deciding on the annotation. This is particularly true for complex tasks e.g. annotating relations between temporal expressions where it is not obvious whether there is any relation between temporal expressions. Importantly, rule formulation time does not depend on the size of the knowledge base.

Our incremental knowledge acquisition framework is inspired by Ripple Down Rules (Compton and Jansen, 1990) which allows users to add rules to the knowledge base (KB) incrementally while automatically ensuring that the knowledge base is always consistent. A new rule added to the KB is only applicable to those cases where the current knowledge base did not perform satisfactorily according to the users. This effectively avoids the adverse interactions of multiple rules in the KB.

We show that with our framework KAFTIE (Knowledge Acquisition Framework for Text classification and Information Extraction), we can quickly develop a large KB based on a small data set that performs better than machine learning approaches trained on a much bigger data set on the task of recognizing temporal relations.

2 TimeML

TimeML is intended as a Metadata Standard for Markup of events, their temporal anchoring and how they relate to each other (Pustejovsky et al., 2003). It aims at capturing the richness of time information by formally distinguishing events and temporal expressions in text. TimeML defines four temporal elements as tags with attributes: TIMEX3, SIGNAL, EVENT and LINK. TIMEX3 is modelled on TIMEX (Setzer and Gaizauskas, 2001) and TIMEX2 (Ferro et al., 2001). It marks up explicit temporal expressions such as times, dates, durations etc. SIGNAL is used to annotate function words that indicate how temporal objects are to be related to each other e.g. temporal connectives (*when*) or

Proceedings of the Australasian Language Technology Workshop 2005, pages 87–95,
Sydney, Australia, December 2005.

temporal prepositions (*on, during*). The EVENT tag covers elements in a text describing situations that occur or happen. Syntactically, EVENTs are tensed verbs, event nominals, stative adjectives and modifiers. The LINK tag encodes various relations that exist between temporal elements of a document which is divided into three subtypes namely: TLINK, SLINK and ALINK. TLINK is a temporal link representing a relation between an event and a time or between two events. SLINK represents a subordination relation between two events or an event and a signal. ALINK represents an aspectual relationship between two events.

In this paper, we report results on recognizing TLINK between an event and a time expression. The main reason for focusing on this subtask is to enable comparison with existing works. In fact, our approach is not geared towards this task and is general enough to be applicable to recognize other link types as well.

3 Knowledge Acquisition Methodology

In this section we present the basic idea of Ripple-Down Rules (Compton and Jansen, 1990) which inspired our approach. RDR was first used to build the expert system PEIRS for interpreting chemical pathology results (Edwards et al., 1993). PEIRS appears to have been the most comprehensive medical expert system yet in routine use, but all the rules were added by pathology experts without programming skill or knowledge engineering support whilst the system was in routine use. Ripple-Down Rules and some further developments are now successfully exploited commercially by a number of companies.

Knowledge Acquisition with Ripple Down Rules: Ripple Down Rules (RDR) is an unorthodox approach to knowledge acquisition. RDR does not follow the traditional approach to knowledge based systems (KBS) where a knowledge engineer together with a domain expert perform a thorough domain analysis in order to come up with a knowledge base. Instead a KBS is built with RDR incrementally, while the system is already in use. No knowledge engineer is required as it is the domain expert who repairs the KBS as soon as an unsatisfactory system response is encountered. The expert is merely required to provide an explanation for why in the given case, the classification should be different from the system's classification.

Suppose the system's classification was produced by some rule R_A. The explanation would refer to attributes of the case, such as patient data in the medical domain or a linguistic pattern matching the case in the natural language domain. The new rule R_B will only be applied to cases for which the provided conditions in R_B are true and for which rule R_A would produce the classification, if rule R_B had not been entered. I.e. in order for R_B to be applied to a case as an exception rule to R_A, rule R_A has to be satisfied as well. A sequence of nested exception rules of any depth may occur. Whenever a new exception rule is added, a difference to the previous rule has to be identified by the expert. This is a natural activity for the expert when justifying his/her decision to colleagues or apprentices. A number of RDR-based systems store the case which triggered the addition of an exception rule along with the new rule. This case, being called the *cornerstone case* of the new rule R, is retrieved when an exception to R needs to be entered. The cornerstone case is intended to assist the expert in coming up with a justification, since a valid justification must point at differences between the cornerstone case and the case at hand for which R does not perform satisfactorily.

Single Classification Ripple Down Rules: A single classification ripple down rule (SCRDR) tree is a finite binary tree with two distinct types of edges. These edges are typically called *except* and *if not* edges. See Figure 1. Associated with each node in a tree is a *rule*. A rule has the form: *if* α *then* β where α is called the *condition* and β the *conclusion*.

Cases in SCRDR are evaluated by passing a case to the root of the tree. At any node in the tree, if the condition of a node N's rule is satisfied by the case, the case is passed on to the exception child of N. Otherwise, the case is passed on the N's if-not child. The conclusion given by this process is the conclusion from the last node in the RDR tree which *fired*. To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default* node.

A new node is added to an SCRDR tree when the evaluation process returns the wrong conclusion. The new node is attached to the last node evaluated in the tree provided it is consistent with the existing rules. If the node has no exception link, the new node is attached using an exception link, otherwise an *if not* link is used. To determine the rule for the new node, the expert formulates a rule which is satisfied by the case at hand.

4 Our KAFTIE framework

We use SCRDR for building knowledge bases in the KAFTIE framework. While the process of incrementally developing knowledge bases will eventu-

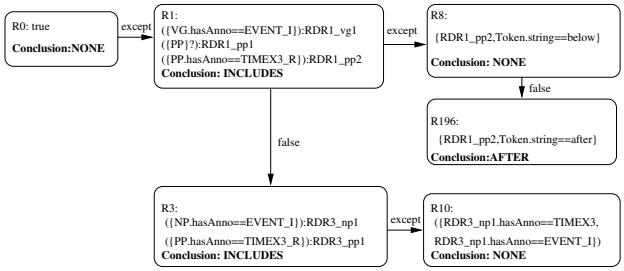


Figure 1: An extract of a KB for recognizing TLINK between an EVENT and a TIMEX3. Note that SCRDR is different from a decision tree: rules in internal nodes can be used to give conclusions to input cases.

ally lead to a reasonably accurate knowledge base, provided the domain does not drift and the experts are making the correct judgements, the time it takes to develop a good knowledge base depends heavily on the appropriateness of the used language in which conditions can be expressed by the expert.

Some levels of abstraction in the rule's condition is desirable to make the rule expressive enough in generalizing to unseen cases. To realize this, we use the idea of annotations where phrases that have similar roles (belong to the same concept) are deemed to belong to the same annotation type. Annotations contain the annotation type, the character locations of the beginning and ending position of the annotated text in the document, and a list of feature value pairs.

4.1 Rule description

A rule is composed of a condition part and a conclusion part. A condition is a regular expression pattern over annotations. It can also post new annotations over matched phrases of the pattern's subcomponents. The following is an example of a pattern which posts an annotation over the matched phrase:

({Noun}{VG.type==FVG}{Noun}):MATCH

This pattern would match phrases starting with a Noun annotation followed by a VG, which must have feature *type* equal to FVG, followed by another Noun annotation. When applying this pattern on a piece of text, MATCH annotations would be posted over phrases that match this pattern. As annotations have feature value pairs, we can impose constraints on annotations in the pattern by requiring that a feature of an annotation must have a particular value.

A piece of text is said to satisfy the rule condition if it has a substring that satisfies the condition pattern. The rule's conclusion contains the classification of the input text. In the task of recognizing temporal relations between a pair of temporal expressions (event or time), the conclusion is either the relation type or NONE.

Besides classification, our framework also offers an easy way to do information extraction. Since a rule's pattern can post annotations over components of the matched phrases, extracting those components is just a matter of selecting appropriate annotations. In this paper, the extraction feature is not used, though.

4.2 Annotations and Features

Built-in annotations: As our rules use patterns over annotations, the decision on what annotations and their corresponding features should be are important for the expressiveness of rules. Following annotations and features make patterns expressive enough to capture all rules we want to specify for various tasks.

We have Token annotations that cover every token with *string* feature holding the actual string, *category* feature holding the POS and *lemma* feature holding the token's lemma form.

As a result of the Shallow Parser module, which will be described in the next section, we have several forms of noun phrase annotations ranging from simple to complex noun phrases, e.g. NP (simple noun phrase), NPList (list of NPs) etc. All forms of noun phrase annotations are covered by a general Noun annotation.

There are also VG (verb groups) annotations with *type, voice, headVerbPos, headVerbString* etc. features and other annotations e.g. PP (prepositional phrase), SUB (subject), OBJ (object).

An important annotation that makes rules more general is Pair which annotates phrases that are bounded by commas or brackets. With this annotation, the following sentences:

> [PP On [TIMEX3 Monday TIMEX3] PP] [NP the company NP] [VG bought VG] ....
> [PP In [TIMEX3 recent months TIMEX3] PP] [NP a group of lenders NP] [Pair , led by Bank of America , Pair] [VG has extended VG] .....

could be covered by the following pattern:

{PP.hasAnno == TIMEX3}{NP}
({Pair})?{VG.type == FVG}

Every rule that has a non-empty pattern would post at least one annotation covering the entire matched phrase. Because rules in our knowledge base are stored in an exception structure, we want to be able to identify which annotations are posted by which rule. To facilitate that, we number every rule and enforce that all annotations posted by rule number x have the prefix RDRx_. Therefore, if a rule is an

exception of rule number x, it could use all annotations with the prefix RDRx_ in its condition pattern.

Apart from the requirement that an annotation's feature must have a particular value, we define extra constraints on annotations namely *hasAnno*, *hasString*, *underAnno*, *endsWithAnno*. For example, *hasAnno* requires that the text covered by the annotation must contain another specified annotation:

NP.hasAnno == TIMEX3

only matches NP annotations that has a TIMEX3 annotation covering its substring. This is used, for example, to differentiate a TIMEX3 in a noun group from a TIMEX3 in a verb group.

Custom annotations: Users could form new named lexicons during the knowledge acquisition process. The system would then post a corresponding annotation over every word in those lexicons. Doing this makes the effort of generalizing the rule quite easy and keeps the knowledge base compact.

4.3 The Knowledge Acquisition Process in KAFTIE

The knowledge acquisition process goes through a number of iterations. The user gives a text segment (e.g. a sentence) as input to the KB. The conclusion (e.g. classification) is suggested by the KB together with the *fired* rule R that gives the conclusion. If it is not the default rule, annotations posted by the rule R are also shown (see section 6.3) to help the user decide whether the conclusion is satisfactory.

If the user does not agree with the KB's performance, there are two options of addressing it: adding an exception rule to rule R or modifying rule R if possible. In either case, user's decision will be checked for consistency with the current KB before it gets committed to the KB.

To create a new exception rule, the user only has to consider why the current case should be given a different conclusion from rule R. This effort does not depend on the knowledge base size.

Modification of existing rules in the KB is not normally done with RDR as it is viewed that every rule entered to the KB has its reason for being there. However, we find that in many natural language applications it is desirable to modify previously entered rules to cover new cases.

To inspect results of a new exception rule or a modified rule R, the user can inspect the annotations posted by the rule of interest through a user interface shown in figure 2.[1] The user can quickly check the impact of the rule on a document or even on the whole training corpus. If the corpus is annotated, statistical performance of the rule will also be collected. It is found that a good GUI is necessary to productively create and test rules.

[1]The interface design is inspired by Boguraev's work.

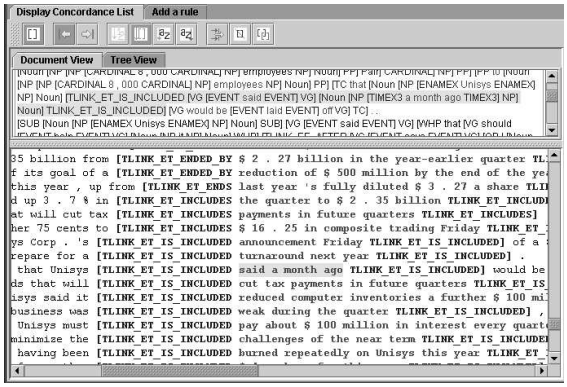


Figure 2: Display of Concordance list to quickly inspect results of rules.

5 Implementation

We built our framework KAFTIE using GATE (Cunningham et al., 2002). A set of reusable modules known as ANNIE is provided with GATE. These are able to perform basic language processing tasks such as POS tagging and semantic tagging. We use *Tokenizer, Sentence Splitter, Part-of-Speech Tagger and Semantic Tagger* processing resources from ANNIE. *Semantic Tagger* is a JAPE finite state transducer that annotate text based on JAPE grammars. Our rule's annotation pattern is implemented as a JAPE grammar with extensions to enable extra annotation feature constraints. We also developed additional processing resources for our tasks:

Shallow Parser: a processing resource using JAPE finite state transducer. The shallow parser module consists of cascaded JAPE grammars recognizing noun groups, verb groups, propositional phrases, different types of clauses, subjects and objects. These constituents are displayed hierarchically in a tree structure to help experts formulate patterns, see e.g. Figure 3. The Shallow Parser module could be refined as needed by modifying its grammars.

All these processing resources are run on the input text in a pipeline fashion. This is a preprocessing step which produces all necessary annotations before the knowledge base is applied on the text.

6 Experiments

We build a knowledge base using KAFTIE to recognize TLINK relations between an EVENT and a TIMEX3 using the TimeBank corpus.

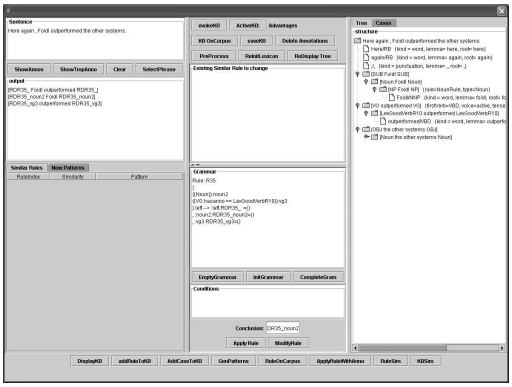


Figure 3: The interface to enter a new rule where the rule is automatically checked for consistency with the existing KB before it gets committed. Annotations including those created by the shallow parser module are shown in the tree in the *structure* box.

6.1 The TimeBank corpus

The TimeBank corpus is marked up for temporal expressions, events and basic temporal relations based on the specification of TimeML. Currently, the TimeBank corpus has 186 documents.

Excluding TIMEX3 in document's meta-data (doc creation time), the majority of TLINKs is between EVENTs and TIMEX3s within the same sentence. Hence, in all of our experiments, we focus on recognizing intra-sentential temporal relations.

The TimeBank annotation guidelines suggest distinctions among TLINK types between two EVENTs but do not explicitly specify how those types are different when it comes to relations between an EVENT and a TIMEX3. In fact, some of the TLINKs types between an EVENT and a TIMEX3 are hard to distinguish and a number of cases inconsistency is observed in the corpus. In this paper, we group similar types together: BEFORE and IBEFORE are merged, AFTER and IAFTER are merged and the rest is grouped into INCLUDES.

6.2 KAFTIE for extracting Temporal Relations

For the task of extracting EVENT-TIMEX3 temporal relations, we consider all pairs between an EVENT and a TIMEX3 in the same sentence and build a knowledge base to recognize their relations. The sentence containing the pair is used as the input to the knowledge base. As there could be more than one EVENT or TIMEX3 in the same sentence, we change the EVENT and the TIMEX3 annotations in focus to EVENT_I (instance) and TIMEX3_R (related_to) annotations respectively. This enables our rule's pattern to uniquely refer to the pair's arguments. For each pair, the KB's conclusion is the type of its temporal relation if there exists a relation between the pair's arguments or NONE otherwise.

6.3 How to build a Knowledge Base

The following examples are taken from the actual knowledge base (KB) discussed in section 6.4 and shown in figure 1. Suppose we start with an empty KB for recognizing temporal relations between an EVENT and a TIMEX3 within the same sentence. I.e. we would start with only a default rule that always produces a *NONE* conclusion. When the following sentence is encountered:

> Imports of the types of watches
> [VG [EVENT_I totaled EVENT_I] VG]
> [PP about $37.3 million PP]
> [PP in [NP [TIMEX3_R 1988 TIMEX3_R] NP]
> PP], ...

our empty KB would use the default rule to suggest the relation between EVENT_I and TIMEX3_R is NONE i.e. no relation exists. This can be corrected by adding the following rule to the KB:

> **Rule 1:**
> (({VG.hasanno==EVENT_I}):RDR1_vg1
> ({PP }?):RDR1_pp1
> ({PP.hasAnno == TIMEX3_R}):RDR1_pp2
>):RDR1_
> **Conclusion:** INCLUDES

Each of the component in the rule's pattern is automatically assigned a tag which will effectively post a new annotation over the matched token strings of the component if the pattern matches a text. New tags of rule i always start with $RDRi\_$. This rule would match phrases starting with a VG annotation which covers the EVENT_I annotation, followed by an optional PP annotation followed by a PP annotation covering the TIMEX3_R annotation. When the sentence containing the pair EVENT_I-TIMEX3_R is matched by this rule, the pair is deemed to be of INCLUDES relation type. Once matched, new annotations RDR1_vg1, RDR1_pp1 and RDR1_pp2 will be posted over the first VG, the first PP and the last PP in the pattern respectively. This is to enable exception rules to refer to the results of previously matched rules. Notice here that the first PP component is specified optional. It could be that the expert already *anticipates* future cases and make the current rule more general. Alternatively, experts always have a choice of modifying existing rule to cover new cases.[2] When we encounter this pair:

> The company's shares
> [**RDR1_vg1** [VG are [EVENT_I wallowing

[2] Automatic recommendation on which existing rules and how to modify to cover new cases is reported in (Pham and Hoffmann, 2005).

EVENT_I] far VG] **RDR1_vg1]**
[**RDR1_pp2** [PP below [NP their [TIMEX3_R 52-week TIMEX3_R] NP] PP] **RDR1_pp2]** high....

Rule **R1** *fires* and suggests that the pair has INCLUDES relation with the newly posted annotation highlighted in bold face. This is incorrect as there is no relation between the pair. The following exception rule is added to fix the misclassification:

Rule 8:[3]
({RDR1_pp2,Token.string==below}):RDR8_
Conclusion: NONE

This rule says that if the second PP matched by Rule 1 (RDR1_pp2) starts with a token string *below* then there is no relation between the pair. Notice that the sentence could have different PP annotations. As each rule posts unique annotations over the matched phrases, we can unambiguously refer to relevant annotations.

However, when we encounter the following case

It [**RDR1_vg1** [VG is deeply [EVENT_I discouraging EVENT_I] VG] **RDR1_vg1]**
[**RDR1_pp1** [PP for [NP the family NP] PP] **RDR1_pp1]**
[**RDR1_pp2** [PP after [NP [TIMEX3_R 22 months TIMEX3_R] NP] PP] **RDR1_pp2]** but ....

This case will be classified as INCLUDES type by Rule 1 while it should belong to AFTER type. We can add an exception to Rule1 catering for this case:

Rule 196:
({RDR1_pp2,Token.string==after}):RDR196_
Conclusion: AFTER

6.4 Experimental results

Out of 186 documents in the TimeBank corpus, we randomly took half of that as training data and keep the remaining half for testing. Using our KAFTIE framework, we built a knowledge base of 229 rules to recognize relations between an EVENT and a TIMEX3 in the same sentence using the training data.[4] The knowledge base uses NONE as its default conclusion. In other words, by default and initially when the KB is empty, all EVENT-TIMEX3 pairs are deemed not to have any TLINK relations. The overall results are shown in table 1 for two settings: *3 types* (BEFORE, INCLUDES and AFTER see section 6.1) and *without typing* - collapsing all TLINK types into one type, effectively detecting if the pair has a TLINK relation regardless of the type. While the accuracy reflects performance on the test data across all types including NONE, the F-measure is based on only TLINKs types, i.e. excluding NONE.[5] On the *without typing* setting, the built knowledge base achieved an F-measure of more than 75% and an accuracy of 86.7%.

Methods	3 types		w/o typing	
	F-m	Acc.	F-m	Acc.
KAFTIE	**71.3%**	**86.1%**	**75.4%**	**86.7%**
J48 (5-folds)	62.3%	78.7%	66.4%	81.2%
SMO (5-folds)	61.4%	77.4%	63.1%	78.7%

Table 1: Results on recognizing TLINKs for w/o typing and 3 types settings.

Comparison with machine learning: For comparison with standard machine learning approaches, we use Weka's J48 and SMO (Witten and Frank, 2000) as implementations for C4.5 and support vector machine algorithms respectively. To adapt machine learning algorithms to the task of extracting relations, we define the following feature representation which is capable of capturing the relation arguments (EVENTs and TIMEX3s) and the surrounding context. We break up the sentence containing the EVENT-TIMEX3 pair into five segments namely: spans of the two arguments (EVENT and TIMEX3), span between the two arguments and spans to the left/right of the left/right arguments. From each segment, we use token strings, token lemmas, parts-of-speech, bigrams of parts-of-speeches and all annotations from the Shallow Parser as features.

J48 and SMO are run using 5-fold cross validation. As the result could vary depending on the seed used for the cross validation, we report results averaged over 100 runs with different random seeds.As can be seen in table 1, the knowledge base built using our framework significantly outperforms standard J48 and SMO. In fact, our knowledge base with the initial 60 rules (as the result of seeing roughly 60 TLINK pairs) already outperforms J48 and SMO (see figure 4).

7 Reducing Knowledge Acquisition

In this section, we investigate how machine learning could be used to reduce the knowledge acquisition

[3]We only select some rules to show as examples, hence indices of rules are not consecutive

[4]To evaluate the TLINK recognition task alone, we use the EVENT and TIMEX3 annotations in the TimeBank corpus. That would also enable us to make a fair comparison with (Boguraev and Ando, 2005) as they also used *perfect* EVENT and TIMEX3 annotations from TimeBank.

[5]The NONE type occurs approximately 2.5 times more often than the TLINK types.

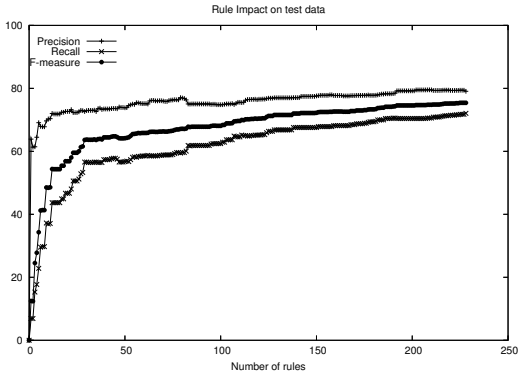


Figure 4: Impact of incrementally adding rules to the KB.

effort. A knowledge acquisition (KA) session corresponds to the creation of a new rule as a result of the KB performing incorrectly on a particular case. Notice that the KB's default rule is quite simple. It always returns the default conclusion which is *NONE* for the task of recognizing temporal relations. We conjecture that if we start with a better default rule then the number of knowledge acquisition sessions can be reduced. One way of doing it is to take some training data to train a classifier as the default rule. We carry out the following experiment:

We focus on the task of recognizing temporal relation in the *without typing* setting and use the exact training and test data from the previous section to build and test a KB respectively. The difference is that we now split the training data into two parts i.e. *ML data* and *KB data*. The *ML data* is used to train a classifier as a default rule in a KB while the *KB data* is used to add exception rules to the KB.

Instead of having real experts involved in the process of creating exception rules, we simulate it by consulting the KB built from the previous section, called *oracleKB*. For each example in the *KB data* that is misclassified by the current KB, we use the *fired* rule for the example from the *oracleKB* i.e. the rule that the *oracleKB* would use on the example.

Table 2 shows the results of using J48 and SMO which are trained on varying portions of the training data. Specifically, it shows the f-measure of the classifier alone, the KB with the classifier as the default rule on the test data as well as the number of rules in the KB. The number of rules in the KB reflects the number of KA sessions required for building the KB. All figures are averaged over 20 different runs using different seeds on splitting the training data into *ML data* and *KB data*.

As we increase the percentage of *ML data*, the number of KA sessions required gets smaller. Ideally, we want to minimize the number of KA sessions while maximize the f-measure. At one extreme end when the *ML data* is empty (0% of the training data), we effectively rebuild a KB in the same fashion as in the previous section with different orders of examples. The f-measure of 75.1% suggests that the performance of our approach are independent of the order of examples presented to the experts.

As it can be seen from table 2, the f-measures of the KBs with a classifier as the default rule follow the same trend. It first degrades as we start giving some data to the classifier and improves as more data is used to train the classifier. After certain thresholds, the f-measures start degrading again as we get less data for experts to add exception rules while the classifiers do not improve their performance.

Depending on the task and the classifier used, we can choose an appropriate amount of data to train a classifier as the default rule in order to substantially reduce the number of KA sessions involved while still achieve a reasonable performance. For example with J48 the best performance is at 72.5% when we use 60% of the training data for training J48 and the rest to add exceptions rules. Compared to a fully manual approach (using 0% data for the classifier), we achieve a 60% reduction in the number KA sessions.

As the *oracleKB* is built with the assumption that the default rule always returns *NONE*, all of the exception rules at level 1 (exception rules of the default rule) are rules covering *INCLUDES* instances. Even though rules at level 2 cover *NONE* instances, they have limited scopes because they were created as exceptions to level 1 rules. When the default rule gives an incorrect *INCLUDES* conclusion, it is likely that we would not be able to consult the *oracleKB* for an exception rule to fix this error.[6] It therefore suggests that if we use real experts, we could achieve a better result.

8 Discussions and Conclusions

Among the pioneering works on linking time and event expressions (Mani et al., 2003; Boguraev and Ando, 2005), only (Boguraev and Ando, 2005) reported results on publicly available data (TimeBank) which allows us to carry out performance comparison.[7] They use a robust risk minimization classifier utilizing a complex set of features including syntactic constructions derived from finite state

[6] A better simulation is to build another *oracleKB* with the default rule always returning *INCLUDES* conclusion.

[7] To the best of our knowledge, this is the only work done on TimeML compliant analyser using TimeBank corpus to date.

%	j48	j48 +kb	#KA	smo	smo +kb	#KA
0%	0	**75.1**	173	0	**75.1**	173
0.5%	28.4	**66.4**	146	27	**68.6**	151
1%	33.5	**65.2**	143	27.3	**71.6**	158
5%	45.3	**67.2**	138	54.3	**71.9**	142
10%	53.3	**69**	131	61.7	**72.4**	133
20%	62.9	**72.3**	117	64.3	**72.7**	118
30%	64.2	**71.9**	103	65.6	**72.7**	108
40%	66.6	**71.9**	91	66.1	**72**	93
50%	67.6	**72.2**	78	66.6	**71.8**	84
60%	68.5	**72.5**	65	67	**71.7**	70
70%	68.9	**71.7**	51	66.7	**71**	59
80%	69	**71.3**	38	66.3	**70.2**	44
90%	68	**70.3**	22	66	**68.5**	25

Table 2: Results of using j48/smo as the default rule for KBs averaged over 20 different random seeds. The first column is the percentage of the training data used to train a classifier (j48/smo). *j48* column contains the f-measures of the classifier alone on the test data. *j48+kb* column contains the f-measures of the KB with j48 as the default rule on the test data with the number of rules in the KB shown in column *#KA*. The last three columns are similar for the smo classifier.

analysis. We would assume that their features are geared towards the task, and presumably took substantial time to develop. Our rules created by the experts use annotations generated by a shallow parser. Importantly, our shallow parser is of a general purpose nature and does not generate extensive clause structures like in (Boguraev and Ando, 2005). In fact, we reuse the shallow parser developed for a different task in the technical papers domain (Pham and Hoffmann, 2004) with minor modifications. It indicates that our approach is not domain and task dependent as rules are crafted based on annotations generated by a general purpose shallow parser. It can be seen from table 3 that the KB built using our framework results in a better F-measure on all 3 settings of limiting the token distance between the EVENT and TIMEX3.

It should be noted that we used only half of the data for building the KB, while (Boguraev and Ando, 2005) used 80% of the data for training. Figure 4 shows the performance of our knowledge base on the test data as rules are incrementally added. Given the upwards trend of the graph as more rules are added, it is plausible that our KB would get to even higher performance had we used 80% of the available data for building the knowledge base.

We have shown that with our unconventional KA approach, namely RDR, we could quickly build a knowledge base that performs better than existing machine learning approaches while requiring much less data. As demonstrated in section 7, the process of building a KB can be boostraped by using machine learning algorithms. Looking at this from a different view, machine learning algorithms' performance can be improved by augmenting the KB built in KAFTIE.

Distance	Method	w/o typing
any	KAFTIE	**75.4**%
	Boguraev&Ando	74.8%
distance $\leq$ 16 tokens	KAFTIE	**77.2**%
	Boguraev&Ando	76.5%
distance $\leq$ 4 tokens	KAFTIE	**82.8**%
	Boguraev&Ando	81.8%

Table 3: Comparison with (Boguraev and Ando, 2005), who used 5-fold cross validation, i.e. 80% of the data for training while we only used 50% of the data to build the KB.

Independent of the knowledge base size, it took 7 minutes on average to create one rule. This includes the time needed to read the sentence to understand why there is a certain relation between the pair of an EVENT and a TIMEX3 as well as the time required to test the new rule before committed to the KB. If the users have to classify the pair's relation from scratch, when we do not have annotated data, then the actual time spent on creating a rule would be much less, as understanding the sentence takes most of the time. Importantly, we do not need to spend time engineering the features representation/selection for the task at hand which is usually done in machine learning approaches.

Thus our approach is particularly suitable for new tasks, when annotated data is not available or limited. Annotators could use KAFTIE to build an annotated corpus as well as a classifier at the same time. By requiring users to justify, in the form of rules, their decisions every time they annotate a case, it helps to annotate the corpus consistently. Furthermore, it also bootstraps the whole process as shown in section 6.4: after looking at half of the data to build a KB, the KB's accuracy on the other unseen half of the data is 86.7% in the '*without typing*' setting.

References

Branimir Boguraev and Rie Kubota Ando. 2005. TimeML-compliant text analysis for temporal reasoning. In *Proceedings of IJCAI*, UK.

Paul Compton and R. Jansen. 1990. A philosoph-

ical basis for knowledge acquisition. *Knowledge Acquisition*, 2:241–257.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. Gate: An architecture for development of robust hlt applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics(ACL)*, Philadelphia, PA.

G. Edwards, P. Compton, R. Malor, A. Srinivasan, and L. Lazarus. 1993. PEIRS: a pathologist maintained expert system for the interpretation of chemical pathology reports. *Pathology*, 25:27–34.

Lisa Ferro, Inderjeet Mani, Beth Sundheim, and George Wilson. 2001. TIDES temporal annotation guidelines, MTR 01W0000041 MITRE technical report.

Richard Fikes, Jessica Jenkins, and Gleb Frank. 2003. JTP: A system architecture and component library for hydbrid reasoning. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando Florida, USA.

Elena Filatova and Eduard Hovy. 2001. Assigning time-stamps to event-clauses. In *Proceedings of the 10th Conference of the EACL*, Toulouse, France.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th annual meetings of the ACL*, Hong Kong.

Inderjeet Mani, Barry Schiffmann, and Jianping Zhang. 2003. Inferring temporal ordering of events in news. In *Proceedings of the NAACL*, Edmonton, Canada.

Son Bao Pham and Achim Hoffmann. 2004. Extracting positive attributions from scientific papers. In *7th International Conference on Discovery Science*, Italy.

Son Bao Pham and Achim Hoffmann. 2005. Intelligent support for building knowledge bases for natural language processing. In *Workshop on Perspective of Intelligent System Assistance*, Palmerston North, New Zealand.

J. Pustejovsky, J. Castano, R. Ingria, R. Sauri, R. Gaizauskas, A. Setzer, G. Katz, and D. Radev. 2003. TimeML: Robust specification of event and temporal expressions in text. In *AAAI Spring Symposium on New Directions in Question Answering.*, Standford, CA.

A. Setzer and R. Gaizauskas. 2001. A pilot study on annotating temporal relations in text. In *Workshop on temporal and spatial information processing, ACL*, Toulouse.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann.

Formal Grammars for Linguistic Treebank Queries

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

Steve Cassidy
Centre for Language Technology
Macquarie University
Steve.Cassidy@mq.edu.au

Abstract

There has been recent interest in looking at what is required for a tree query language for linguistic corpora. One approach is to start from existing formal machinery, such as tree grammars and automata, to see what kind of machine is an appropriate underlying one for the query language. The goal of the paper is then to examine what is an appropriate machine for a linguistic tree query language, with a view to future work defining a query language based on it. In this paper we review work relating XPath to regular tree grammars, and as the paper's first contribution show how regular tree grammars can also be a basis for extensions proposed for XPath for common linguistic corpus querying. As the paper's second contribution we demonstrate that, on the other hand, regular tree grammars cannot describe a number of structures of interest; we then show that, instead, a slightly more powerful machine is appropriate, and indicate how linguistic tree query languages might be augmented to include this extra power.

1 Introduction

There has been recent interest in looking at what is required for a query language for annotated linguistic corpora (Lai and Bird, 2004). These corpora are used in a range of areas of natural language processing (NLP)—parsing, machine translation, and so on—where they form the basis of training data for statistical methods; and also in linguistics, where they are used to extract examples of particular phenomena for analysis and testing of hypotheses. As noted by Lai and Bird, the prototypical hierarchical linguistic annotation is the syntax tree, and consequently the type of query language that is of interest is a tree query language.

One approach to deciding what is required in a tree query language, taken by Lai and Bird (2004), is to examine and compare a range of existing ones: for example, TGrep2 (Rohde, 2001), TIGERSearch (König and Lezius, 2001), or Emu (Cassidy and Harrington, 2001). One of their goals is to understand better the formal properties required of query languages.

It has been noted in a number of places that treebank querying is in essence a specification of a tree pattern, which matches against the desired trees in the treebank corpus (Abiteboul, 1997). These tree patterns can be described by existing formal machinery such as tree grammars and automata. An approach complementary to the one mentioned above is thus to examine the extent to which this formal machinery is adequate for describing tree patterns relevant for the sorts of queries of interest in NLP or linguistics, and then to link this to a query language. A reason for being interested in this link between tree query languages and formal machinery is that standard results are available for these latter. For example, an algorithm for recognition exists that is linear in the number of nodes in the tree and the size of the automaton; it is decidable whether the set of matches will be empty; and so on (Comon et al., 1997). Further, there is the promise of the availability of standard tools and efficient techniques that could be used by a tree query language. This is the case for formal machines over strings (for example, the library of finite-state string transducers

Proceedings of the Australasian Language Technology Workshop 2005, pages 96–104,
Sydney, Australia, December 2005.

of Mohri (1997)), although not yet for trees.

A number of researchers have looked at this complementary approach. One alternative is to design from scratch a tree query language derived from a tree grammar or automaton; this is taken by, for example, Chidlovskii (2000). Another is to relate an existing query language to a formal machine: Murata et al. (2000) present a taxonomy of XML schema languages using formal language theory.

In this paper, we follow the second of these alternatives. Existing work, mostly focussed on XML, has looked only at regular tree grammars and automata for modelling query languages; we examine the extent to which this is the case for linguistic treebanks, and what other machinery might be appropriate for a linguistic tree query language. We argue that while regular tree grammars might be satisfactory for a querying a broad range of phenomena, not all queries over trees representing natural language can be based on a regular tree grammar. This is a structural analogue of the work of Shieber (1985), which showed that natural language as a string language cannot be generated by a context-free grammar, which corresponds at the tree level to a regular tree grammar.

In Section 2 we give the definition of regular tree grammars, along with an approach used to relate them to XPath. In Section 3 we look at extensions to XPath that Cassidy (2003) argues are necessary for linguistic corpus querying, and show that these can be captured by regular tree grammars. In Section 4, however, we present some examples from Dutch from the work of Bresnan et al. (1982) to show that not all desired queries can be represented by regular tree grammars, and examine the question of what strong generative capacity is necessary in a tree grammar for representing natural language. Section 5 gives the definition of a more powerful grammar, the context-free tree grammar, and shows how this can describe queries related to the Dutch instance, along with what properties a query language based on these might have. Finally, Section 6 concludes.

2 Regular Tree Grammars

Regular tree grammars (RTGs) are in essence those trees whose paths are defined by regular grammars. Comon et al. (1997) provide an introduction to necessary concepts in tree grammars, along with well known results such as that the string languages yielded by RTG trees are the context-free languages. In their treatment they divide tree representations into two types: those for ranked trees (that is, where each symbol has a fixed number of children, with this number constituting the rank of the symbol), and those for unranked trees. XML schema languages typically use unranked trees, so we adopt, slightly modified, the definitions of these from Brüggemann-Klein et al. (2001) and Murata et al. (2000).

A regular tree grammar is a 4-tuple $G = (\Sigma, N, P, S)$ such that

- Σ is a finite set of terminal symbols;
- N is a finite set of nonterminal symbols;
- P is a finite set of productions of the form $X \rightarrow a(R)$, where $X \in N$, $a \in \Sigma$, and R is a regular expression over $N \cup \Sigma$; and
- S is the start symbol, $S \in N$.

Derivation $\Rightarrow$ (with transitive closure $\stackrel{*}{\Rightarrow}$) is defined in the usual way, with nonterminals symbols rewritten by means of production rules, starting from the start symbol S. ϵ is the conventional null terminal symbol. The set of trees generated by G is $L(G)$.

Example 2.1 Let $G_1 = (\Sigma, N, P, S)$ such that $\Sigma = \{a, b\}$, $N = \{S, X\}$, and $P = \{S \rightarrow a(aSa), S \rightarrow a(bXb), X \rightarrow b(bXb), X \rightarrow b\}$. A sample derivation is given in Figure 1. $L(G_1)$ is thus the set of ternary-branching trees over the symbols a and b, where all nodes to a certain depth D are a nodes, and all below are b nodes.

To relate this to a query language, we now review the approach presented in Wood (2003) to relate these regular tree grammars to

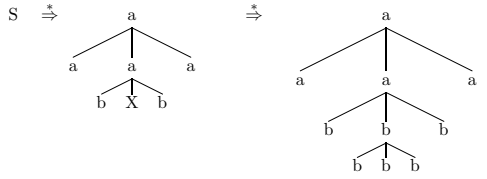


Figure 1: Derivation for RTG G_1

XPath (XPath, 1999). XPath is a language for selecting nodes from XML document trees, and is thus an important part of XSLT and XQuery. Expressions in XPath in themselves can be seen as simple queries over trees.

An XPath expression is a mapping from a node (the *context node*) to the set of all nodes reachable by the specified path. A path expression is written as a series of steps where each step defines the *axis* used to reach new nodes and a *node test* used to restrict the set of nodes reached along the axis. Axes include *child*, *descendent*, *following*, *attribute* and *self*. Node tests consist of two parts: a restriction on the element name and an optional predicate expression. Other features, such as built-in functions, are also allowed. Notationally, a null axis stands for the child axis, // the descendent axis, the wild card * any node label, and [] a predicate expression. The full XPath expression definition is fairly complex, and can be found at XPath (1999); here we give an example.

Example 2.2 [From Wood (2003)] The XPath query $a//b[*/i]/g$ selects nodes labelled with g (called g-nodes for short) that are children of b-nodes, such that the b-nodes are both descendants of the root a node and have an i-node as a grandchild. In the left-hand tree in Figure 2, these would be the nodes in bold font. (The return value of the XPath expression would be the node g, but here we are only concerned with the trees that would be matched.)
The possibility of arbitary functions for node tests means that XPath can be augmented to be arbitrarily powerful. To investigate questions of formal power, then, only subsets of XPath are examined. Wood's paper notes that, for this reason, a number of other researchers have been interested in the properties of different fragments of XPath, denoted $\text{XP}^{\{[]\}}$, $\text{XP}^{\{[],*\}}$ and $\text{XP}^{\{[],*,//\}}$, depending on which XPath constructs are included in the fragment. The work of Wood himself is on the fragment $\text{XP}^{\{[],*,//\}}$ and the question of whether the containment problem—whether one query is subsumed by another—under a Document Type Definition can be decided in the complexity class PTIME. To demonstrate that this is the case he conceives of XPath queries as *tree patterns* (Abiteboul, 1997; Deutsch et al., 1999), which can be described by regular tree grammars. For example, the XPath query of Example 2.2 could be pictured as the tree pattern on the right in Figure 2, where the dotted line indicates non-immediate dominance.

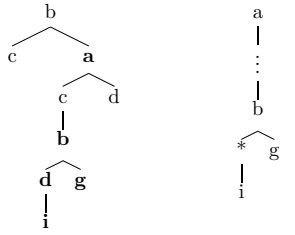


Figure 2: Tree matching $a//b[*/i]/g$, and corresponding tree pattern

An RTG then describes all trees matching this tree pattern. In order to define this RTG, Wood defines some shorthand notation, which we will also adopt. For an alphabet $\Sigma = \{a_1, \ldots, a_k\}$, we write $n \to \Sigma(r)$ for the set of productions $\{n \to a_1(r), \ldots, n \to a_k(r)\}$.

In order to generate an arbitrary tree over Σ, we define a nonterminal n_Σ by the shorthand production $n_\Sigma \to \Sigma((n_\Sigma)^*)$.

Example 2.3 For the query $Q = a/b$ over alphabet Σ, the productions for the corresponding RTG are

$$n_a \to a((n_\Sigma)^*\, n_b\, (n_\Sigma)^*)$$
$$n_b \to b((n_\Sigma)^*)$$

An additional shorthand is to compress the ordering of siblings implicit in RTGs. Each permutation of children at a node would require a separate production, so as shorthand

the & symbol is used: $a\&b$ represents ab and ba ($a, b \in \Sigma$).

Example 2.4 The productions from the query $a[b][c]$ are

$$n_a \rightarrow a((n_\Sigma)^* \,\&\, n_b \,\&\, (n_\Sigma)^* \,\&\, n_c \,\&\, (n_\Sigma)^*)$$
$$n_b \rightarrow b((n_\Sigma)^*)$$
$$n_c \rightarrow c((n_\Sigma)^*)$$

Example 2.5 The productions for the query $a//b$ are

$$n_a \rightarrow a((n_\Sigma)^* \, n_b \, (n_\Sigma)^*)$$
$$n_b \rightarrow b((n_\Sigma)^*)$$
$$n_b \rightarrow \Sigma((n_\Sigma)^* \, n_b \, (n_\Sigma)^*)$$

Wood then defines the following procedure for constructing an RTG G from a query Q, given an alphabet $\Sigma = \{a_1, \ldots, a_k, *\}$ and Q in $\mathrm{XP}^{\{[],*,//\}}$. First, each node in Q is numbered uniquely, with the root node numbered 1. Then G is given by (Σ, N, P, n_1), where $N = \{n_1, \ldots, n_m, n_\Sigma\}$, and P is constructed inductively as follows.

1. If node i in Q is a leaf, then P includes $n_i \rightarrow a_j((n_\Sigma)^*)$ if i has label $a_j \in \Sigma$, or $n_i \rightarrow \Sigma((n_\Sigma)^*)$ if i has label *.
2. If node i in Q has child nodes $j_1, \ldots, j_m$, then P includes $n_i \rightarrow a_l((n_\Sigma)^* \,\&\, n_{j_1} \,\&\, (n_\Sigma)^* \,\&\, \ldots \,\&\, (n_\Sigma)^* \,\&\, n_{j_m} \,\&\, (n_\Sigma)^*)$ if i has label $a_l \in \Sigma$, or $n_i \rightarrow \Sigma((n_\Sigma)^* \,\&\, n_{j_1} \,\&\, (n_\Sigma)^* \,\&\, \ldots \,\&\, (n_\Sigma)^* \,\&\, n_{j_m} \,\&\, (n_\Sigma)^*)$ if i has label *.
3. If node i in Q is connected to its parent by a descendent edge, then P includes $n_i \rightarrow \Sigma((n_\Sigma)^* \, n_i \, (n_\Sigma)^*)$

3 XPath Extensions and RTGs

Cassidy (2003) presented some extensions to XPath, based on the requirements of typical linguistic queries. An example is that of finding matches for a given syllable structure: "Find sequences of any number of consonant phonemes followed by a single vowel following by any number of consonants". This would require a regular expression (C+VC+) over the `/following` axis. Under the current definition of XPath, it is not possible to specify regular expressions over axes; nor is it possible to specify more complex definitions of where conditions on nodes should be applied. Cassidy consequently specifies an extension that allows this, which has the following components:

axis This is as for the axes in the standard XPath definition.

step This determines how many steps should be taken along the axis, and takes the form of a list of positive integers or a special value `inf`. A path length is allowed if it matches one of the integers in the list; any path length is allowed if only the value `inf` is given; path lengths greater than the highest integer are allowed if the list contains both integers and the value `inf`. For example, $[2, 3, 4]$ allows paths only of lengths 2, 3 or 4; $[1, 10, \texttt{inf}]$ allows paths of length 1 or of length greater than 10.

condition This is a general boolean condition on nodes in the same sense as XPath.

where This specifies where on the path the condition must hold, via a list of positive integers or the special values `inf` or `end`. An integer in the list means that the condition is applied to that node; `inf` means that the condition must hold for all nodes in the path; `end` means that the condition applies to the final node on the path. For example, the list $[1, 2, \texttt{end}]$ would find paths where the condition was satisfied by the first, second and last nodes on the path.

A proposed syntax for this, in keeping with the XPath syntax, would be

`/axis{step}::condition::{where}`

The **step** and **where** components would default to their XPath values (1 and `end` respectively).

Example 3.1 Cassidy gives an example for the C+VC+ query which uses the `/following` axis. An analogous example using the axes already presented in Section 2 would be one to match trees with a chain of VP nodes followed immediately below by a chain of NP nodes, such as the one in Figure 3. Such a query could be expressed as

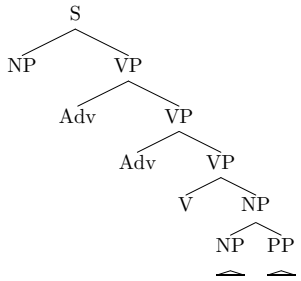


Figure 3: Tree with sequence of VPs and NPs

```
/child{0,inf}::VP::{0,all}
/child{0,inf}::NP::{0,all}
```

The question is then, Can this extension be represented by RTGs? The intuition is that it can—the essence of this aspect of the extension is to allow regular expressions over paths, which is also the essence of RTGs. We demonstrate this below by giving a construction of an RTG for each extended XPath query. As with the construction in Section 2, we will look at just a restricted subset of this XPath extension: **axis** `/child` and `/descendent` (which is in fact just an infinite `/child`); **step** as defined above; **condition** only labels on nodes, or *; and **where** as above. We call this XP-ext.

Example 3.2 The query in Example 3.1 would be represented by the RTG $(\{VP, NP, *\}, \{n_S, n_V, n_N\}, P, n_S\})$, where P is the set of productions

$$n_S \rightarrow \Sigma((n_\Sigma)^* \, n_V \, (n_\Sigma)^*)$$
$$n_V \rightarrow VP((n_\Sigma)^* \, n_V \, (n_\Sigma)^*)$$
$$n_V \rightarrow VP((n_\Sigma)^* \, n_N \, (n_\Sigma)^*)$$
$$n_N \rightarrow NP((n_\Sigma)^*)$$
$$n_N \rightarrow NP((n_\Sigma)^* \, n_N \, (n_\Sigma)^*)$$

To derive an RTG corresponding to a query in XP-ext, we identify the corresponding tree pattern(s). First, to allow regular expressions over paths in tree patterns, we make a further notational extension, either adding the symbol c next to the non-immediate dominance link in the tree pattern, to indicate that the condition holds over the non-immediate dominance link, or the symbol $*$ if it does not.

Now, we need to look at two separate cases. The first case is the general one: in aiming only to match sets of trees by tree patterns, only one tree pattern is generally necessary, regardless of the number of elements specified in the `step` and `where` components. This is because although one pattern tree might be expected for each `step` value, in fact one describing the smallest step suffices, as it subsumes any other. For example, in the query `/child{2,4}::S::{2,4}`, any tree which matches the appropriate pattern tree of height 4 will match the pattern tree of height 2. Note that these different **step**s and **where**s are differentiated with respect to the XPath return values; however, that is beyond the scope of this paper, which concerns itself only with trees matched.

The second case deals with the specific case of `where` value `end`; the subsumption relationship does not hold here. For example, consider the query `/child{2,4}::S::{end}` over trees consisting of a single path. There will be trees where the end of the path (`step`) of length 2 (matching the label S) do not have an S node at the end of the path of length 4, and vice versa; that is, there is no subset relation between the sets of trees specified for `step`s of lengths 2 and 4.

We define these two cases below. In these definitions, we take **axis** `/child` and **condition** (i.e. node label) `c`.

Case 1 For query Q with **step** $[i_1, \ldots, i_n, \texttt{inf}]$ $(i_1 < \ldots < i_n)$, and **where** $[j_1, \ldots, j_m, \texttt{inf}]$ $(j_1 < \ldots < j_m < \texttt{inf})$, we construct a tree pattern of height i_p, where i_p is the smallest element of $[i_1, \ldots, i_n, \texttt{inf}]$ with label `c` for each node j_p $(p < m, j_p < i_p)$. If i_p is `inf`, we label the non-immediate dominance edge `c` if **where** contains `inf`, * otherwise.

Case 2 For query Q with **step** $[i_1, \ldots, i_n, \texttt{inf}]$ $(i_1 < \ldots < i_n)$, and **where** `[end]`, we construct one tree pattern for each element of **step**, with the last node in each tree pattern labelled `c`, and any non-immediate dominance edge labelled $*$.

Example 3.3 The query in XP-ext

Figure 4: Pattern tree for Example 3.3

`/child{inf}::S::{1,3,inf}` would give the pattern tree in Figure 4.

Our construction for an RTG is then as for the construction in Section 2, but with part 3 replaced by with 3':

3' If node i in Q is connected to its parent by a non-immediate dominance edge labelled with $a \in \Sigma$, then P includes $n_i \rightarrow a((n_\Sigma)^* n_i (n_\Sigma)^*)$ or if unlabelled, then P includes $n_i \rightarrow \Sigma((n_\Sigma)^* n_i (n_\Sigma)^*)$

4 Non-Regular Queries

However, some queries that are of interest to (computational) linguists are not able to be expressed by RTGs. There was much discussion in the mid- to late-twentieth century regarding whether natural languages could be described by context-free (string) grammars (CFGs). A fairly common belief was that they could not be, accompanied by attempts to prove this; Pullum and Gazdar (1982) refuted these earlier arguments, and it was not definitively shown that natural language as a string language was not context-free until the work of Shieber on Swiss German (Shieber, 1985).

Bresnan et al. (1982), in addition, made an argument based on syntactic structure, using the example of Dutch. We represent their argument, where they show that an RTG cannot describe this kind of structure—which obviously has some linguistic interest—and hence neither can any tree query language based on it. We then consider what formal tree machine is minimally needed for describing natural language.

Cross-serial dependencies occur when dependencies in a sentence are interleaved with each other, such that in a string $a_1 a_2 \ldots a_n b_1 b_2 \ldots b_n$ there are dependencies between elements a_i and b_i ($i \in \{1 \ldots n\}$). A well known example from Dutch, given in Bresnan et al. (1982), is in Figure 5.

Pullum and Gazdar (1982) showed that it was possible to describe the string language using a CFG, but noted that the associated structure would not necessarily be useful. Bresnan et al. (1982) then comprehensively investigated what structures would be appropriate on linguistic grounds. One proposal they considered was for a flat structure of NPs, with right-branching VPs, as in the leftmost tree of Figure 5, with evidence for the right-branching VPs coming from possibilities of conjunctions in Dutch. However, they noted that the sequence of NPs has more constituent structure than indicated in the leftmost tree of Figure 5, and conclude that the structure in the centre of Figure 5 is the one that is consistent with the data. They note that this proposed structure should be uncontroversial, as it embodies only predicate-argument relations, rather than any aspects of syntax whose representations may be more open to question.

They use a pumping lemma for regular tree languages to demonstrate that sets of these sorts of trees from Figures 5 cannot be generated by RTGs. Comon et al. (1997) give a more precise definition, but broadly, for any tree T of height greater than some constant k in a tree language L, there is a non-trivial segment of T that can be 'pumped' in a manner analogous to in the pumping lemma for regular string grammars; and any tree formed from T with an arbitrary number of these segments inserted appropriately will also be in L. In the case of the centre tree of Figure 5, there is no segment of the tree that can be chosen to be pumped that will maintain equal numbers of NPs and Vs (that is, specifying the same tree language). Roughly speaking, there cannot be counting or matching of numbers of internal symbols in tree patterns.

We note here that there is an additional possibility that they do not discuss that is also

consistent with the data, differing only in the placement of the subtree headed with the V'; this is the rightmost tree in Figure 5. The same pumping lemma shows that the set of these trees also cannot be generated by an RTG.

5 Context-Free Tree Grammar

What, then, can generate these sets of trees? One possibility is a context-free tree grammar (CFTG). The basic idea of these is that, whereas trees generated by RTGs have regular paths, CTFGs have context-free paths.

CFTGs were introduced in Rounds (1970). We do not have space for a full formal presentation of them, and we would also note that unlike the RTGs defined above they are defined over ranked trees. However, here is a brief definition, along with an example. A *context-free tree grammar* G is a 4-tuple $(F, \Phi, P, \mathcal{K}_0)$ where F is a finite ranked alphabet; $\Phi = \{\mathcal{K}_0, \mathcal{K}_1, \ldots, \mathcal{K}_n\}$ is a finite ranked alphabet of nonterminals; P is the set of production rules, a finite set of pairs $(\mathcal{K}_i(x_1, \ldots, x_m), t_x)$, where $i = 0, \ldots, n$, $K_i \in \mathcal{F}$, x_i are variables, and t_x is a tree over F, Φ and variables x_i; and $\mathcal{K}_0$ is the initial nonterminal.

An application of a production rule to a tree T involves choosing a nonterminal $\mathcal{K}_i$ with m children, taking a rule with lefthand side $\mathcal{K}_i(x_1, \ldots, x_m)$, identifying the m subtrees of $\mathcal{K}_i$ in T with the variables $x_1, \ldots, x_m$, and replacing the chosen subtree of T rooted in $\mathcal{K}_i$ with the righthand side of the rule along with appropriately substituted variables $x_1, \ldots, x_m$.

No CFTG is possible for the leftmost tree of Figure 5, as symbols are ranked, and that tree would require an infinite number of ranked symbols with label S to describe the unbounded number of children *NP*.

A CFTG to describe the centre tree of Figure 5 would be $G = (F, \Phi, P, \mathcal{K}_0)$, where $F = \{S, NP, VP, V'\}$, $\Phi = \{\mathcal{K}_0, \mathcal{K}_1\}$, and P is the set of productions given in Figure 6.

A CFTG to describe the rightmost tree of Figure 5 would be $G' = (F, \Phi, P', \mathcal{K}_0)$, where

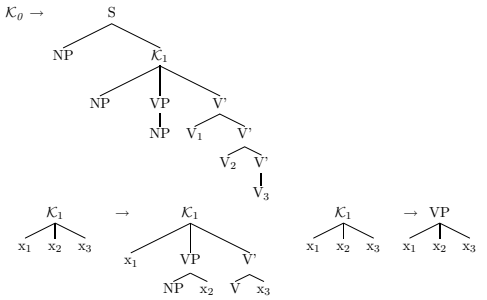


Figure 6: CFTG for cross-serial option 2

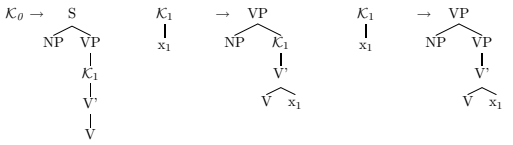


Figure 7: CFTG for cross-serial option 3

F and Φ are as for G above, and P' is the set of productions in Figure 7.

In essence, CFTGs allow 'counting' throughout a tree—in the examples above, the counts of NPs and Vs match—in the same way as CFGs allow counting in a string. Thus CFTGs might be a suitable backbone for a tree query language. However, they are not computationally very attractive. Just as the string languages yielded by RTGs are the context-free languages, the string languages yielded by CFTGs are the indexed languages, which include exponentially increasing languages such as $\{a^{2^n} \mid n \geq 1\}$ that are not a feature of any human language.

However, the sets of productions P and P' are actually quite different. It is straightforward to demonstrate that those in P' are in fact *spinal-formed* under the definition of Fujiyoshi and Kasai (2000). In essence, a spinal-formed CFTG disallows duplication of counts along different paths (as in the duplicate counts of NPs and Vs in separate subtrees of the centre tree of Figure 5). Spinal-formed CFTGs form a proper subset of CFTGs with much restricted power; interestingly, their string languages have been proved by Fujiyoshi and Kasai (2000) to be the class of mildly context-sensitive languages, and recent work (Fujiyoshi and Kawaharada, 2005) has included promising

(1) ...dat Jan Piet Marie de kinderen zag helpen laten zwemmen
...that Jan Piet Marie the children see-PAST help-INF make-INF swim-INF
...that Jan saw Piet help Marie make the children swim

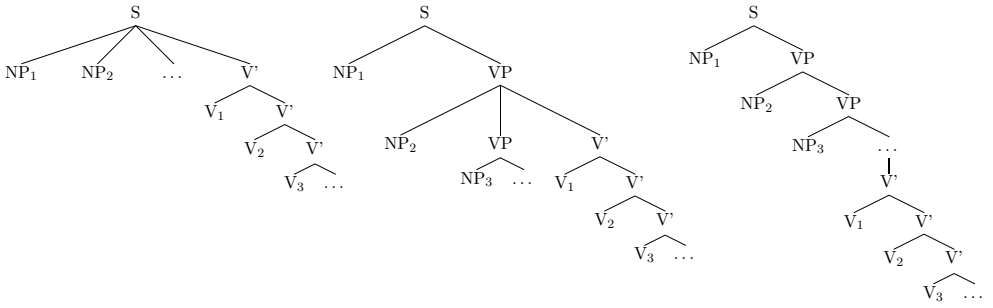


Figure 5: Proposals for cross-serial dependencies

results on recognition complexity.

This rightmost tree of Figure 5, then, establishes that a language to query trees representing the syntax of natural language requires an underlying tree machine beyond RTGs, but not necessarily beyond the power of spinal-formed CFTGs.

An additional result of Fujiyoshi and Kasai (2000) is the definition of a Linear Pushdown Tree Automaton (L-PDTA) that recognises exactly the class of trees generated by spinal-formed CFTGs. These are similar to the automata that recognise the tree sets of RTGs, but have a non-duplicable stack in operation as the automaton walks a path of a tree; that is, the stack can only be passed along a single branch of the tree.

This suggests that a similar mechanism might be appropriate for a tree query language that would allow limited counting. Using the notation of Section 3, we might have a query to find trees with (not necessarily balanced) NPs and Vs with the structure of the rightmost tree of Figure 5) as follows:

```
/child{0,inf}::VP[/child::NP]::{0,all}
/child::VP
/child{0,inf}::V[/child::V']::{0,all}
```

We would then extend this so that the number of nodes matching `/child{0,inf}::VP[/child::NP]::{0,all}` would be one less than the number matched by `/child{0,inf}::V[/child::V']::{0,all}` in order to match only those trees with appropriately paired NPs and Vs.[1] Notationally, this might be represented as:

```
/child{X=|{0,inf}|}::VP[/child::NP]::{0,all}
/child::VP
/child{X=|{0,inf}|}::V[/child::V']::{0,all}
```

where X is a variable containing the count of nodes matched by the particular components of the query, and $|Y|$ represents the number of steps actually matched for list of steps Y.

To restrict this to match only those trees describable by spinal-formed CFTGs, passing counts down through predicate expressions, which would in effect permit stack duplication, is disallowed (e.g. `VP[/child::NP]`).

However, this is just an indication of the form that an XPath-like query language based on a spinal-formed CFTG might take, and is intended only to be the starting point for future work.

6 Conclusion

The aim of this paper has been to examine what formal machinery is necessary for linguistic tree query languages. Existing tree query languages are typically related to regular tree grammars, although these query languages are almost exclusively for non-linguistic XML documents. The first contribution of the paper has been to show

[1] The topmost NP in the tree, under the S, is not matched by the query.

that regular tree grammars can be used as the basis for a range of proposed extensions to XPath motivated by linguistic considerations, for very typical sorts of queries such as those representing a search for a regular expression over axes. The paper's second contribution has been to demonstrate that regular tree grammars cannot, however, be a basis for some queries of linguistic interest. We have shown that machines with at least the power of spinal-formed context-free tree grammars, with their limited ability to count, can describe those constructions that are beyond RTGs, and made initial suggestions on how this ability to count could be incorporated into a query language.

There is much scope for future work in this direction. To deal with XPath return values we are interested in transducers based on (subtypes of) context-free tree grammars which have not yet been defined; many of the properties of the formal mechanisms remain to be investigated; and, most relevant to this paper, it is an open question as to how precisely a query language can be defined based on this mechanism.

References

Serge Abiteboul. 1997. Querying Semi-Structured Data. In Foto Afrati and Phokion Kolaitis, editors, *Database Theory—ICDT'97*, pages 1–18. Springer-Verlag.

Joan Bresnan, Ronald Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial Dependencies in Dutch. *Linguistic Inquiry*, 13(4).

Anne Brüggemann-Klein, Makoto Murata, and Derick Wood. 2001. Regular tree and regular hedge languages over unranked alphabets. Technical Report HKUST-TCSC-2001-05, Hong Kong University of Science and Technology.

Steve Cassidy and Jonathon Harrington. 2001. Multi-level annotation in the Emu speech database management system. *Speech Communication*, 33(1–2):61–77.

Steve Cassidy. 2003. Generalizing XPath for directed graphs. In *Proceedings of Extreme Markup Languages 2003*.

Boris Chidlovskii. 2000. Using Regular Tree Automata as XML schemas. In *Proceedings of IEEE Advances in Digital Libraries*, pages 89–104.

H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 1997. Tree Automata Techniques and Applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. Release October 1st 2002.

Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. 1999. A Query Language for XML. In *Proceedings of the 8th International World Wide Web Conference*, pages 77–91.

A. Fujiyoshi and T. Kasai. 2000. Spinal-Formed Context-Free Tree Grammars. *Theory of Computing Systems*, 33:59–83.

A. Fujiyoshi and I. Kawaharada. 2005. Deterministic Recognition of Trees Accepted by a Linear Pushdown Tree Automaton. In *Proceedings of the Tenth International Conference on Implementation and Application of Automata*.

Esther König and Wolfgang Lezius. 2001. The TIGER language—a description language for syntax graphs. Part 1: User's guidelines. Technical report, IMS, University of Stuttgart.

Catherine Lai and Steven Bird. 2004. Querying and Updating Treebanks: A Critical Survey and Requirements Analysis. In *Proceedings of the Australasian Language Technology Workshop 2004*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Makoto Murata, Dongwon Lee, and Murali Mani. 2000. Taxonomy of XML Schema Languages using Formal Language Theory. In *Proceedings of Extreme Markup Languages 2000*.

Geoffrey Pullum and Gerald Gazdar. 1982. Natural languages and context-free languages. *Linguistics and Philosophy*, 4:471–504.

Douglas Rohde, 2001. *TGrep2 User Manual*.

William Rounds. 1970. Mappings and Grammars on Trees. *Mathematical Systems Theory*, 4:257–287.

Stuart Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Peter Wood. 2003. Containment for XPath Fragments under DTD Constraints. In *Proceedings of the 9th International Conference on Database Theory*, pages 300–314.

XPath. 1999. XML Path Language (XPath), Version 1.0. Available on: `http://www.iw3.org/TR/xpath`.

Extracting Exact Answers using a Meta Question Answering System

Luiz Augusto Sangoi Pizzato and **Diego Mollá-Aliod**
Centre for Language Technology
Macquarie University
2109 Sydney, Australia
{pizzato, diego}@ics.mq.edu.au
http://www.clt.mq.edu.au/

Abstract

This work concerns a question answering tool that uses multiple Web search engines and Web question answering systems to retrieve snippets of text that may contain an exact answer for a natural language question. The method described here treats each Web information retrieval system in a unique manner in order to extract the best results they can provide. The results obtained suggest that our method is comparable with some of today's state-of-the-art systems.

1 Introduction

Text-based Question Answering (QA) focuses on finding answers for natural language questions by searching collections of textual documents. This area of research has become especially active after the introduction of a question answering task in TREC-8 (Voorhees, 1999), which was based on open-domain question answering. The result of this research is a number of systems and QA methodologies not only for generic domains (Moldovan et al., 2003), but also for restricted domains (Mollá et al., 2003) and Web-based systems (Zheng, 2002).

Each type of QA system has specific issues and methodologies. Thus, open-domain QA can rely on generic tools and resources such as parsers, named-entity recognisers, and lexical resources like WordNet (Fellbaum, 1998). This can be seen in recent TREC conferences (Voorhees, 2004b) where some of the participants used readily available third-party resources to quickly build systems that obtained satisfactory results for the amount of effort invested.

On the other hand, restricted domain QA can take advantage of deep knowledge of the covered area by using resources that are specific to the domain such as terminology lists and ontologies, for example in the domain of biomedicine (Zweigenbaum, 2003).

Finally, web-based QA can take advantage of the enormous amount of data available on the World Wide Web and use data-intensive approaches that exploit the inherent redundancy to find answers (Brill et al., 2001). Our system belongs to this category.

The satisfaction of the user with a certain answer will depend on various factors. For instance, someone who wants to find some specific fact would be satisfied with a short and brief answer while someone else may require a more detailed answer. These kind of differences between casual users of generic domain QA systems make the establishment of personalized answer models difficult.

One way that may satisfy both types of users is by providing an exact answer while at the same time showing a snippet of the original text from where the answer was extracted. This kind of response was required from the participant systems of the main task of the QA-track of TREC-2003 (Voorhees, 2004a). We have adopted this approach by providing the exact answer, a summary, and a link to the source document.

According to Voorhees (2003), an exact answer is defined as a string that does not contain any extraneous information but the answer in it. For instance *Brasilia* is the answer for *What is the capital of Brazil?*, but *the city of Brasilia* or *Brazilian capital Brasilia* are not.

In order to find the exact amount of text containing an answer, we used an approach that combines the results of several Web search engines and Web QA systems. Our system works in a similar way of those known as metasearch engines (Metacrawler[1], Mamma[2] and Profusion[3] just to name a few), however we do differentiate between the search engines used

[1] `http://www.metacrawler.com`
[2] `http://www.mamma.com`
[3] `http://www.profusion.com`

Proceedings of the Australasian Language Technology Workshop 2005, pages 105–112,
Sydney, Australia, December 2005.

in order to extract the best information they may provide. Although finding more information for a question helps to retrieve their answers, we believe that the assistance of several search engines can cause improvement when the best information of each one are extracted and weighted.

The common framework for question answering systems consists of three main phases:

Question Analysis: The question is classified into several types, possibly forming a classification hierarchy such as (Moldovan et al., 1999). The question type is typically related to the type of the expected answer, which in turn is typically related to the named-entity types available to the system. The question classification can be based on regular expressions (Mollá-Aliod, 2004; Chen et al., 2002; Hovy et al., 2000) or machine learning (Li and Roth, 2002; Zhang and Lee, 2003). Apart from the question type and expected answer type, this phase may return the question focus and other important words or concepts found in the question.

Information Retrieval: The question and/or the question features obtained by the question analysis are fed to an information retrieval system that returns the documents or document fragments that may contain the answer. Typically a generic document retrieval system is used or even a web search engine, though there are suggestions that the type of information retrieval required for this phase is different from the generic one. This phase is crucial, since relevant documents that fail to be retrieved will be ignored in the next phase.

Answer Extraction: The retrieved documents or passages are analysed in order to find the exact answers. Techniques to find the answer range from the selection of named-entities that are compatible with the expected answer type to the use of logical methods to find and/or validate the answer.

As it can be observed in Figure 1, our system structure is very similar to the common framework, however the approaches for performing each of the tasks are different. The question analysis is performed using the Trie-based question classifier (Pizzato, 2004; Zaanen et al., 2005) trained over the set of near 5500 questions prepared by Li and Roth (2002). As already stated, the information retrieval stage is a combination of several Web search engine results, and the answer extraction combines named-entity, n-grams and lexico-semantic information from WordNet (Fellbaum, 1998).

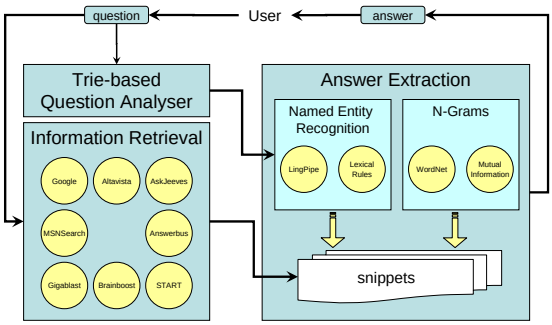


Figure 1: Overview of the system's architecture

In the next section we show some of the characteristics of the Web search engine we explored. Then, in Section 3, we address our method of combining the results and how we used named-entities and n-grams to pinpoint the answer location. In Section 4 we show an evaluation of our technique, while in the last section we present the concluding remarks and future work.

2 Web search results combined

According to Oztekin et al. (2002), the combination of search engine results is not a new approach for improving information retrieval. Many meta Web search engines provide a better retrieval by combining results of several search engines and re-ranking their results according to techniques such as the linear combination of scores described by Vogt and Cottrell (1998). However it seems that most approaches do not consider the differences between search engines. In this work, we take into account the best of each search engine used and, since our goal is to find exact answers to a question, we explored the characteristics of these search engines in order to answer questions.

Because of their availability on the Web, we also used the results of three Web QA systems

(Start[4], Answerbus[5] and Brainboost[6]). These systems perform their jobs using very different approaches and they do not provide exact answers (in the same sense of Voorhees (2003)), but only snippets of text where the answers are expected to be.

As stated we extracted the best of several search engines. Let's list some of their characteristics.

Start: Combines predefined Web databases to provide answers to Geography; Science and Reference; Arts and Entertainment; and History and Culture. The answers are normally structured as tables, sentences or even images and graphics.

Answerbus: Question Answering system that provide answers in a snippet-like format.

Brainboost: Provides answers to natural language questions in a similar way to Answerbus.

Altavista[7]: Well established search engine with a large amount of indexed Web pages.

AskJeeves[8]: It provides very useful information regarding specific questions on famous people, movies, definitions of words, and current weather.

Gigablast[9]: It has the feature, referred to as Gigabits, that presents related concepts to the search results. There is no disclosure on how this information is calculated (we would guess n-grams computation from every search result), but it is possible to notice that the answer for a question is likely to appear in the list of Gigabits.

MSN Search[10]: The Microsoft search engine has the ability to answer some NL questions by using encyclopedia information, as well as providing definitions for words, and a way to make measurement conversion.

Google[11]: It is considered one of the best Web search engines available. It also provides some information on definitions questions like: *What is a platypus?* or *define:platypus*. Following MSN Search, Google has recently acquired the ability to answer encyclopedia questions. We understand that this is a good feature to be used in our system, and we are planning to incorporate this. However the version and results we describe in this paper does not yet consider the QA ability for Google.

The results obtained from the search engine were combined into four different sets:

1. Answers from MSN Search;
2. Answer summaries from Start, Answerbus and Brainboost;
3. Definitions from Google, MSN Search and AskJeeves;
4. Summaries of the results from every Web search/QA system used;

The exact answer was extracted using these sets in a slightly different manner. For instance we observed that, the encyclopedia answers from MSN Search are of a high quality and they are easily pinpointed due to the fixed format and the short size of the passage used. The not-yet incorporated QA feature of Google will fit into this first set when implemented.

The answer snippets from Start, Answerbus and Brainboost do not have the high quality of MSN Search, but they normally contain the right answer within their results.

For definition questions we checked the definition results of Google, MSN Search and AskJeeves. If they are not present and the question asks for a definition, we rephrase the question to the search engine submitting a query in the format *define [question_focus]* or *define:[question_focus]* (on Google) in order to obtain a definition if available.

Because it is not possible to delimit an exact answer in definitions, we state that for these type of questions an exact answer is a brief description of the question subject (focus).

The last set involves all the snippets of documents obtained from the Web search engines. We used the top-50 documents provided by every search engine appended to each other. We did not merge common documents, since we believe that the process of finding the correct answer will take advantage of the several instances of the same information.

[4] http://www.ai.mit.edu/projects/infolab
[5] http://www.answerbus.com
[6] http://www.brainboost.com
[7] http://www.altavista.com
[8] http://www.ask.com
[9] http://www.gigablast.com
[10] http://search.msn.com
[11] http://www.google.com

3 Exact answer extraction

A good approach for answering questions is to provide an exact answer combined with a snippet of text supporting the answer. This may boost the satisfaction of the users of a QA system since the validation of the answers is fast and straightforward. The approach used for pinpointing the exact answer location uses named-entity recognition combined with n-grams extraction and word overlap. We also make use of the semantic classification of terms in WordNet (Fellbaum, 1998).

We first established some priorities in the sets of answers retrieved. If the answer requires a definition, the set of definition answers is evaluated, if this set is empty we try to rephrase the question forcing the search engines to provide a definition if one exists. In case a definition could still not be found, we give up this approach since the information may not be available in a dictionary or even the question analyser may have made a mistake when defining the expected answer category. Giving up the approach means that we will try to find the answer as if the question did not require a definition.

For exact answers, we found that in the rare cases when MSN Search answers questions, they are normally correct. Because of this, we first consider the summary of MSN Search answers if present to extract the exact answer. Otherwise we evaluate the set of answers from the QA systems. If no answer could be found, the set of all search engine responses is analysed.

If still no answer can be found, we relax the expected answer category to all the fine grained categories that the question classification returned. If by this time still no answers are found, the coarse grained categories are used.

3.1 Pinpointing an exact answer

Given the preferences explained above, we define the exact answer by extracting all the named-entities that match the expected answer category provided by our question analyser. The answer categories follow Li and Roth (2002) classification. They are divided into coarse and fine grained categories as shown on Table 1.

We used a large collection of gazetteer files, involving most types of named-entities, along with the LingPipe named-entity recognizer[12] for the definition of persons, organization and location names. In the spirit of Mikheev et al. (1999), we developed a set of internal and external lexical patterns in order to define the remaining types of named-entities.

Table 1: Answer classification and examples

Coarse	Fine	Example
HUM	IND	Who killed JFK?
HUM	GR	What ISPs exist in the NYC?
LOC	CITY	What is the capital of Brazil?
NUM	SPEED	How fast is light?
NUM	MONEY	How much does the President get paid?
DESC	DEF	What is ethology?
ENTY	ANIMAL	What is a female rabbit called?
ENTY	FOOD	What was the first Lifesaver flavor?
ENTY	SUBSTANCE	What is a golf ball made of?
ENTY	DISMED	What is a fear of disease?
ENTY	TERMEQ	What is the name of the Jewish alphabet?

HUM (human)	IND (individual)	GR (group)
LOC (location)	NUM (number)	DESC (description)
DEF (definition)	ENTY (entity)	DISMED (disease)
TERMEQ (equivalent term)		

For every named-entity found we calculate a score according to their average distance from all question words. Consider $F = \{f_1, f_2, \ldots, f_n\}$ to be the sequence of words in the question focus, and $\delta(a, b)$ the distance in words between two terms a and b in the summaries retrieved by the search engines. The score $S(E)$ of a named-entity E is computed as follows:

$$S(E) = \sum_{i=1}^{n} \frac{\delta(E, f_i)^{-1}}{n}$$

The $S(E)$ scoring assumes that possible answers are more likely to be close to the question focus words (Pasca, 2003; Kwok et al., 2001).

Although this provides a measure showing if a named-entity is likely to be the answer in a certain piece of text, we consider that the presence of the same answer string in different passages provides more hints that the answer string is the answer. In order to take advantage of the redundancy the Web provides (Brill et al., 2001), we sum the scores that a named-entity receives for every passage found.

We also have two extra processes that help to improve the answer extraction. The first one uses the Gigabits from Gigablast search engine. If an identified named-entity is in the Gigabits set, the score $S(E)$ is summed to a percentage value given by Gigablast.

The last ranking process uses n-grams information. Unigrams, bigrams and trigrams are extracted from all the responses from the search engines and the mutual information of the n-grams are extracted. The mutual information

[12] http://www.alias-i.com/lingpipe/

$I(a,b)$ is calculated as follows:

$$I(a,b) = \log \frac{P(a,b)}{P(a)P(b)}$$

Observe that $P(a,b)$ is the probability of occurrence of the unigram a followed by unigram b (bigram (a,b)) and $P(x)$ is the probability of occurrence of unigram x.

Since there is no mutual information for unigrams, we calculated a similar measure by the natural logarithm of the product of the probability of finding a certain unigram ($P(u) = freq(u)$/corpus size in unigrams) by the number of different unigrams in the collection of all retrieved summaries.

Those n-grams representing question words and stopwords are discarded and the values from the mutual information that are larger than one[13] are tested on the upper hypernym hierarchy of WordNet. We assumed that if an n-gram is a hyponym of a question word, it may increase the chance that this hyponym is the answer of a question. This helps to answer questions like *Which breed of dog has a blue tongue?*. By using this technique we increase the score of any breed of dog found within the search engine results.

We may still use n-grams that are not in WordNet, however we assign a very low score to them. If the n-gram was also identified as a named-entity, their scores are summed, otherwise it becomes the score for the n-gram alone. The score used by the n-grams is only a fraction of the mutual information calculation. We empirically defined the added value for n-grams found in WordNet hyponyms as a tenth of the mutual information value, while the added value for non-WordNet n-grams was defined as a 20th part.

This seems useful in two distinct cases. First, when the question analysis module fails we are still able to retrieve the correct answer; Second, it gives perspectives on answering multilingual questions. The second case is feasible, however it needs a list of the language stopwords and if possible a lexico-semantic database like WordNet.

After these scoring procedures take place, we collapse and sum scores of different numeric named-entities if their values are the same. For instance 'Two million' is the same as '2 million' or even '2,000,000'. However, though this idea is promising we haven't had the time to implement more information clusters. In the same manner of Kwok at al. (2001), information cluster would help to improve the system precision by grouping similar strings of text. Other information cluster could also include measurement conversions (i.e. 1 km = 1000 meters) and synonyms.

[13] A mutual information value larger than one means that the n-gram occurs more often than its probability of random occurrence.

```
<RANKED_ANSWERS>
 <QUESTION str="What is the capital of Brazil?"/>
  <ANSWER str="Brasilia" id="1" score="3.22">
   <DOC url="http://www.brazzil.com/p35nov95.htm">
    <TITLE>BRAZZIL - News from Brazil - FOOD -
           BRASILIA'S RECIPES
    </TITLE>
    <SUMMARY>
      Brasilia, the capital of Brazil, is better
      known for its prize-winning ultramodern
      design and for the unfriendliness of the
      city to the people who live there
    </SUMMARY>
   </DOC>
   <DOC url="http://gosouthamerica.about.com/b/a/
   065069.htm">
    <TITLE>Brasilia, Capital of Brazil</TITLE>
    <SUMMARY>
       Brasilia, Capital of Brazil. South America
       for Visitors Blog. ... Brasilia, Capital
       of Brazil Brasilia is a monument to what
       Brazilians can do and have done.
    </SUMMARY>
   </DOC>
  </ANSWER>
</RANKED_ANSWERS>
```

Figure 2: System current output

With this final score, every exact answer is ranked and then presented to the user with their source passages. Current results are shown into a XML-like structure containing all the answers and their passages. The idea is to develop a Web interface that will allow users to find answers with a minimum effort and also to provide feedback on the answer quality. An example of the current system output is shown in Figure 2.

The evaluation of the method was performed in a similar way as the main task of the QA track of TREC-2003 (Voorhees, 2004a) as we describe in details in Section 4.

4 Evaluation

The QA track of TREC conferences (Voorhees, 2004a) provides a common environment where QA systems can be tested and evaluated under the same measures. The main task for QA required exact answers from the participant's systems. They were required to provide the answer

and indicate one document in the AQUAINT corpus supporting that answer. The systems' results were manually evaluated from NIST personnel and the answers/systems scores were calculated in a different way for factoid, list and definition questions.

We performed the evaluation of our method only for factoid questions. The system was evaluated using the Mean Reciprocal Rank (MRR) measure of previous TREC QA tracks (Voorhees, 2002). This measure is the average of the precision for every question on it's first correct answer. The MRR is calculated as $\sum_{q=1}^{k} \frac{1}{r_1}$, where r is the ranking of the first correct answer and k is the number of questions.

To obtain the accuracy of our system for fact-based questions we ran the 413 questions of this type from the QA track of TREC-2003 using our system and performed an automatic evaluation using the answer patterns provided by NIST.

As expected these answers did not reflect every answer found in the Internet. Many questions had a correct answer that could not be identified by the patterns. The reasons for this may vary from the lack of answers representations to different or updated answers. For instance our system identified *6000 degrees Celsius* as the answer for *How hot is the sun?*, but the automatic evaluation could only validate answers that follow the patterns of Table 2.

Table 2: TREC-2003 patterns for automatic evaluation of answers.

6500 Celsius
6,?000 degrees Centigrade
two million degrees centigrade

Since the patterns are based on the answers supported by the AQUAINT corpus, some may contain outdated information, for instance *How many NFL teams are there?* requires *31* as its answer according to the patterns, however today's NFL is played with *32* teams.

Because of the reasons listed above, we pre-evaluated our system using the TREC answer pattern in order to adjust some of its parameters, and then we performed a manual evaluation in a slightly different way than the NIST guidelines. Due to time limitations we could not verify all the information sources for unsupported answers. Therefore the answers were assigned only as right or wrong.

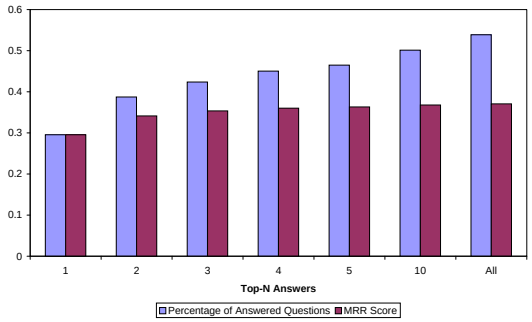


Figure 3: Results for the Top-n answers returned

As observed in Figure 3 using this approach for factoid questions we obtained an accuracy of 30% for the first answer. This result is reasonable considering that it is of the same value of the average results of TREC-2003 systems. This result places our system among the top-5 competitors of the main task for factoid questions.

We should stress that our system does not provides answers using the AQUAINT corpus, nor indicates a document to support the answers in that corpus. We also did not compute the NIL recall and precision since some NIL answers (answer that AQUAINT corpus did not provide an answer) could be found by using the Internet.

We can observe from these results that the exact answer could be found almost half of the times by considering up to 5 answers for every question, giving a reasonable MRR score of 0.36. With these results we may say that the performance obtained by our system could be compared with some of the best systems in TREC.

5 Concluding Remarks

In this work, we develop a meta-QA system that combines the results of different Web search/QA systems in order to provide exact answers for natural language questions. By using a trie-based question analysis, named-entity recognition, n-gram computation and lexico-semantic information from WordNet, we were able to achieve results comparable to some best state-of-the-art QA systems.

Even though our system regards heavily in third-part systems for information retrieval, we showed that it is possible to use and combine the

results from these systems in order to extract exact answers.

Since the developed system is highly modularized, it is possible to remove and add search engines, making the use of those here cited just the first trial for this approach. Further work is needed in order to identify the gain in performance by adding, replacing, removing and promoting search engines. There is also a need for the evaluation of the best weights for the features used to pinpoint the location of the answers, and the feasibility of using language independent methods such as n-grams and mutual information to perform a multilingual QA.

Other interesting aspect of this approach is the capacity of taking advantage of certain features provided by search engines. For instance, by restricting the search domain by Web site, language, country or even neighborhoods, it is possible to restrict the QA domain as well. We already performed some minor tests asking questions in the Macquarie University Web site showing promising results.

We may say that the success of a Web question answering system may not only depend on the precision of its answer. We believe that an effort has to be made in the user interface allowing them to easily verify the answer provided. Further work is needed to be done on developing such a user interface.

References

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC 2001*, number 500-250 in NIST Special Publication. NIST.

J. Chen, A.R. Diekema, M.D. Taffet, N. McCracken, N. Ercan Ozgencil, O. Yilmazel, and E.D. Liddy. 2002. Question answering: CNLP at the TREC-10 question answering track. In *Proceedings of TREC-2001*, pages 485–494.

C. Fellbaum, editor. 1998. *WordNet: An electronic Lexical Database*. MIT Press.

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

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the web. *ACM Trans. Inf. Syst.*, 19(3):242–262.

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

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

D. Moldovan, S. Harabagiu, M. Paşca, R. Mihalcea, R. Goodrum, Roxana Gîrju, and Vasile Rus. 1999. LASSO: A tool for surfing the answer net. In Voorhees and Harman (Voorhees and Harman, 1999).

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

D. Mollá-Aliod. 2004. Answerfinder in TREC 2003. In *Proceedings of TREC-2003*.

D. Mollá, F. Rinaldi, R. Schwitter, J. Dowdall, and M. Hess. 2003. Extrans: Extracting answers from technical texts. *IEEE Intelligent Systems*, 18(4):12–17.

B. Uygar Oztekin, George Karypis, and Vipin Kumar. 2002. Expert agreement and content based reranking in a meta search environment using Mearf. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 333–344. ACM Press.

Marius Pasca. 2003. *Open-Domain Question Answering from Large Text Collections*. CSLI Publications, Stanford California, USA.

Luiz Augusto Sangoi Pizzato. 2004. Using a trie-based structure for question analysis. In Ash Asudeh, Cécile Paris, and Stephen Wan, editors, *Proceedings of the Australasian Language Technology Workshop 2004*, pages 25–31, Macquarie University, Sydney, Australia, December. ASSTA. ISBN: 0 9581946 1 0.

Christopher C. Vogt and Garrison W. Cottrell. 1998. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, October.

Ellen M. Voorhees and Donna K. Harman, editors. 1999. *The Eighth Text REtrieval Conference (TREC-8)*, number 500-246 in NIST Special Publication. NIST.

Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In Voorhees and Harman (Voorhees and Harman, 1999).

Ellen M. Voorhees. 2002. Overview of the

TREC 2001 question answering track. In *Proceedings of The Tenth Text REtrieval Conference (TREC 2001)*.

Ellen M. Voorhees. 2003. Overview of the TREC 2002 question answering track. In *Proceedings of TREC-2002*.

Ellen M. Voorhees. 2004a. Overview of the TREC 2003 question answering track. In *Proceedings of TREC-2003*.

Ellen M. Voorhees. 2004b. Overview of TREC 2003. In *Proceedings of TREC-2003*.

Menno Van Zaanen, Luiz Augusto Pizzato, and Diego Molla. 2005. Question classification by structure induction. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-2005).*, Edinburgh, Scotland, August.

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

Zhiping Zheng. 2002. Answerbus question answering system. In *Human Language Technology Conference (HLT 2002)*, San Diego, CA, March 24-27.

Pierre Zweigenbaum. 2003. Question answering in biomedicine. In *Proc. EACL2003, workshop on NLP for Question Answering*, Budapest.

Multimedia presentation of grammatical description: design issues

Simon MUSGRAVE
School of Languages, Cultures and Linguistics
Monash University
Victoria 3800 Australia
Simon.Musgrave@arts.monash.edu.au

Abstract

In this paper, I argue that grammatical description of language is a type of information which is ideally suited to presentation as a multimedia object structured with hypertext. I examine three existing language resources, constructed for different audiences, and discuss various features of each which bear on the design issues relevant to grammatical description. From my examination of these exemplars, I argue for four guidelines in the design of a multimedia grammar: data centricity, multiple linking, exhaustive coding in data structures, and user control of the amount of information accessed.

1.Introduction

Any reasonably complete description of a language is a complex object. Traditionally, such works are divided into various components: a grammar, a dictionary and a text collection. But of course these are really highly inter-related. For example, a single entry in the dictionary is of little value without the general information about words of that class which can be found in the grammar, and any point made in the grammar may be hard to grasp without extensive exemplification from texts..

An impression of the complexity involved can be gauged from the following comments by two reviewers of a description published as three separate volumes (Heath 1980, 1982, 1984): "Unfortunately, F[unctional] G[rammar of] N[unggubuyu] is a very demanding work, both because of the inherent complexity of the language and because it requires the reader to make constant reference to the text volume." (Blake 1985: 310); "the work is particularly difficult to read. H[eath] makes no pedagogical concessions to the reader. One must look up the attestations for every major grammatical point in another volume." (Haiman 1986: 654-655).

The interrelatedness of the various components discussed above immediately suggests that hypertext would be a better means of presentation and additional benefits could come from making the grammatical description a multimedia object, rather than a text object. Examples could be heard in the original sound recorded by the researcher, or even seen as video clips where such presentation would aid the consumer (for example, where gesture added an important element of meaning to the utterance). In addition to the improved accessibility of the descriptive information, such presentation would bring the consumer much closer to the primary data, actual language in use, and therefore multimedia language description would increase substantially the standard of accountability in linguistics.

However, the standard paper and ink presentation of grammatical description has an established linear format which is not suitable for the new medium. In this paper, I examine three existing presentations of language data as multimedia: an online documentation, a documentation published as CD-ROM, and an online language learning site. I suggest that each of these exemplars can provide important hints to the most appropriate structure for multimedia grammatical description.

1. The organization of grammatical description

Most grammatical descriptions published in book format follow more or less closely a standard format. The presentation begins with background information on the language and its speakers, the relationship of the language to other languages, and a survey of previous research. The description proper then follows, moving through phonetics and phonology (the sounds of the language and how they are organized into a system), morphology (word-formation processes), and clausal syntax. Some discussion of syntax above the level of the individual clause and of textual organization may follow. If example texts are included in the volume, as is common, they will come after this, with word lists after them.

The organization of a grammar in this style is linear, that is, one sort of information is presented before another. And the linearity is to a large extent well-motivated. It is generally not easy to understand the morphological processes of a

Proceedings of the Australasian Language Technology Workshop 2005, pages 113–119,
Sydney, Australia, December 2005.

language before one understands the phonology; it is hard to understand syntax (combinations of words) before one understands morphology (word-formation).

Linearity of presentation is also a consequence of the medium. Paper and ink objects are read normally in sequence; even if one reads only a short section of a larger work, one starts at a particular place and reads on in sequence for as long as necessary. The reservations of the reviewers of Heath's work quoted previously are reflections of their frustration at an organization which attempted to subvert this linearity.

Hypertext, on the other hand, is a non-linear medium and the metaphor of a web is entirely appropriate for such presentation. As already mentioned, hypertext has clear benefits for the presentation of grammatical description, but it is desirable that at least some of the linear logic of the paper and ink model should be accessible in the new medium. Various questions at the conceptual level must therefore be addressed in order to exploit the technological possibilities to their full. These include:

- what should be the starting point for the consumer's navigation of the multimedia object?
- how much linear ordering can or should be built into the interlinking of individual objects?
- how rich can the interlinking be before it becomes confusing for the consumer to navigate?

Underlying all of these questions, is the assumption that all of the coding necessary for full interlinking of information in the description is present in the data structures on which the presentation object is based. I return to this issue in section 3.

2.Three exemplars

2.1.Online documentation of Kolyma Yukaghir (Nikolaeva and Mayer 2004)

2.1.1.Description

This resource (afterwards ODKY) is a documentation of an endangered language (see Himmelmann 1998 for discussion of the notion of documentation). It contains introductory material, texts, dictionaries and images (pictures and maps). The texts are presented so that an entire text can be heard as audio in one track, while a translation is viewed, or the text can be viewed broken into units, with morpheme-by-morpheme glossing and links to audio for each unit. For some text units, notes on grammatical or cultural matters are provided. In this view, many morphemes and words are also linked to the dictionaries. There are two dictionaries, one a listing of Yukaghir stems, the other a listing of affixes. The dictionaries give rather limited information: a short definition, a note if a word is a loan from another language, and a concordance of occurrences of the morpheme in the text collection. All entries in the concordance are links to the relevant text unit. The concordance lists are exhaustive, and this is unwieldy in the case of common items. For example, the verb 'to be' has a concordance list of around 500 occurrences. Where a morpheme has more than one form, the various possibilities are linked to the form considered basic. Links between words in texts and images are used in a limited way, the most noticeable being that a picture of the speaker who produced a text is often available.

2.1.2.Discussion

The nature of a documentation has influenced the design of ODKY. The aim is to provide a record of the language and of the linguistic behaviour of its speakers. Detailed grammatical description is not a part of the intention, rather the documentation is intended to serve as the basis for description by future scholars. Nevertheless certain features are of interest.

Firstly, the presentation of the material is not complex. There is very little annotation added to the data, and therefore linking paths through the material are straightforward. There are no instances where multiple links lead from a single location, and therefore no design is imposed at the level of hypertext linking.

Secondly, the presentation of the material is centred on actual data. Texts, as audio and as transcriptions constitute the greater part of the documentation. Supporting annotation is minimal; for example, the information given in dictionary entries includes only a single word gloss and a concordance. Not even word class labels are given, let alone more detailed definitions or encyclopaedic information.

Thirdly, concordances are used in the dictionaries. It is a very significant advantage of presenting language data via computer that all of the data relevant to some particular question can be accessed quickly (assuming the underlying annotation is rich enough – see discussion in section 3). However, the way that this feature is exploited in ODKY raises the question of whether it is necessary or desirable to always present such information exhaustively. For example, in a resource based on English data, it may not be useful to have every occurrence of the definite

article listed in the dictionary entry for that word, although in order to answer certain questions one would like to be able to access the information when it is needed. We return to discussion of this question in section 3 below.

2.2.Spoken Karaim (Csato & Nathan 2003)

2.2.1.Description

This source (afterwards SK) embodies a rather more sophisticated approach to the possibilities of multimedia than that discussed in section 2.1. To some extent, this is a result of delivery via CD rather than online. But in many cases, the design features are not dependent on the delivery medium. For example, the SK environment uses multiple windows to present various types of information on a single screen. Such presentation is an option available in web browsers, but one not exploited by ODKY.

This resource presents a variety of information about the Karaim people of Lithuania, including general cultural information along with information about the Karaim language. The package was developed in close collaboration with the community (Csato and Nathan 2004), and therefore the intended audience is different from that for ODKY. The specifically linguistic information is viewed across three windows: one large window contains a text unit in Karaim with an English translation, with a button on a control bar beside the screen allowing access to audio; a second smaller window displays a single item from the lexicon; and a third window, also small, displays the lexicon as a list. In the last two windows, various options are offered. The lexicon list can be viewed in either Karaim or English, or a third option labelled 'Grammar' can be selected. This third option does not give access to descriptive material, but only to a list of the grammatical categories which are represented in the language, for example names of nominal cases. Selecting from any of the three possible views in the lexicon list window results in the chosen item being displayed in the lexicon item window. The division between the dictionary of words and the dictionary of grammatical categories is in effect almost identical to the division between the word dictionary and the affix dictionary used by ODKY.

The information displayed in the dictionary entry window is similar to that given in the ODKY dictionary, that is, simple translations and no word class information. In some cases, some more encyclopaedic information is provided (e.g. *kibin* 'kibin (Karaim national dish, pirog filled with meat or cabbage)'). Additional options offered in this window include links to pictures in some cases, links to related words, and a morphology demonstration module. For nouns, identified by a graphic signal beside the word, it is possible to display various inflected forms by clicking and dragging on buttons at the top of the window. This feature is intended as an aid to language learners; it does not provide exhaustive information even for nominal morphology.

A short section of grammar notes can be viewed in the main window, that in which text is viewed. These notes are very brief and are not linked at all to text examples. They are also not complete even at the level of detail provided. The section on morphology discusses only some nominal morphology, while the list of grammatical categories, read in conjunction with the dictionary entries, makes it clear that the language also has a considerable amount of verbal morphology.

2.2.2.Discussion

SK shares with ODKY the first two features discussed above, simple annotation and therefore no complex paths of links through the material, and being data-centred. There is no possibility of extracting groups of data in this resource, such as is offered by the dictionary concordance in ODKY. The main feature of interest in SK is that mentioned already at the start of this section: the presentation of various types of information in different areas of the screen. The value of this technique in presenting linguistic data has also been demonstrated for the Shoebox/Toolbox software package (distributed by SIL: <http://www.sil.org/computing>) by Austin (2002).

One other issue concerning SK should be mentioned. This resource is presented in a specially designed environment, it does not use a standard web browser as does ODKY and the Nahuatl Learning Environment (see section 2.3). With dissemination via a CD, this poses no problems for the user, however it does raise questions about the portability of the design. Use of open source tools throughout the design, implementation and delivery of any resource is clearly desirable.

2.3.Nahuatl Learning Environment (Amith n.d.)

2.3.1.Description

The Nahuatl Learning Environment (afterwards NLE) is an even more ambitious project which aims to present online a corpus of texts in the Nahuatl language, along with a reference grammar and a comprehensive dictionary. Various possibilities for linking between these modules are planned, although only some are currently implemented. In its current form, this resource

provides a rich dictionary of the Nahuatl language, a corpus of texts which are linked to varying degrees to other information, and grammatical description which is only available as downloadable text files.

Texts are presented with an entire text on a single page with a single link to audio provided for each text. No morpheme-by-morpheme glossing is given for texts, but some complex words are active links with parses displayed as a pop-up on rollover. If the link is clicked, a new window opens with the dictionary entry for the root of the complex word. In comparison to ODKY and SK, these dictionary entries are very detailed with word class information, definitions for multiple senses and supporting examples, and links to related entries and audio. It is also possible to follow links from texts, or from an open dictionary entry, which automatically generate queries to the lexical database.

Some specific notes on grammar are present as footnotes in the texts, accessed via links. These are currently the only links to grammatical description in NLE, although in documentation, it is claimed that it is possible to view for example sets of verbs of the same subcategory via links which automatically query the database.

The dictionary module of NLE is based on a database application (Hyperlex2), and offers powerful query facilities (regular expression searches etc.). Other features include the possibility to switch between English and Spanish as the metalanguage, and detailed encyclopaedia information on topics such as botanical knowledge among the Nahuatl people.

2.3.2.Discussion

Of the three resources discussed here, NLE shows the greatest development of the possibilities of linking between various parts of the available information. As the available information is significantly richer than that in ODKY and SK, for example in the dictionary entries, the presentation does become complex. However, this complexity is currently restricted to the amount of information available on the screen at a single time. Complex paths through the information and multiple linking are not offered. Thus, a link from a complex word form in a text shows a parse of the word and leads to a dictionary entry for the root morpheme. But there is, for example, no path to the other morphemes which occur in the complex form. The implementation of the linking is also less than optimal, with the dictionary entry appearing as a pop-up area which overlays the text from which the user has started. The text is therefore no longer fully visible. The use of separate screen areas, as in SK, seems a more satisfactory solution.

2.4. Summary

Table 1 summarises the features of the three language resources discussed in this section.

3. Design guidelines for multimedia grammar

Various guidelines for the design of a multimedia grammar emerge from a consideration of the characteristics of the three resources surveyed in section 2, and additional ones can be inferred from the features which are lacking in those resources. Here, I concentrate on four of these: the data-centric nature of such a grammar, the multiple pathways between data and annotation, the exhaustive coding of properties needed in underlying data structures, and the possibility that the user should have some control over the degree to which information is presented exhaustively.

As noted in section 2, all three of the resources surveyed are centred on data. Actual language data, transcribed text or the original audio recording, is the point from which the user gains access to other information. In some cases, descriptive and analytic information can be accessed without viewing or hearing a text, but this is not the preferred mode of use. In these resources, any descriptive annotation has value in relation to the concrete examples provided by real data. This can be seen as both a practical and a philosophical decision. Practically, the combination of description with data provides a richer and more rapid understanding to the user. Philosophically, the great advantage of multimedia as the means for presenting linguistic material is that it allows for easy access to large amounts of data, and this imposes accountability on the analyst. Therefore, I suggest that these resources are following the correct approach and that being data-centric is a desideratum for the design of multimedia grammatical description.

It is clear that there is a huge potential problem in the design of multimedia grammars which arises from the fact that any single piece of data can potentially be linked to multiple annotations. A single word in a text can be linked to a morphological analysis of the word, to dictionary entries for each individual morpheme, as well as (perhaps) a dictionary entry for the whole word, to information on phonology, on syntax and other possibilities. Only NLE has any kind of rich linking of various sources of information in its structure. In at least one case, the potential problem is handled by dividing the work up: rolling over some complex word forms gives a pop-up window

with a morphological parse of the word, while the word in the original text is a live link to dictionary information. In the case where three or more links are needed, such a solution will not work, but the use of pop-up menus on rollover is an elegant solution to making several choices available.

Multiple paths through the available information will also be necessary. For example, a word form in a text might be linked to annotations concerning both morphology and phonology, but the phonological process in question might be dependent on the morphological environment. In such a case, a direct link would exist between the word form and the phonological discussion, but an indirect link would also have to exist via the material on morphology. Note that such paths of linking will mimic some of the linear structure of a book grammar: if one phenomenon cannot be understood without knowledge of some other phenomenon, then the presentation of the one logically precede that of the other. Linear sequence handles this logic in a book, linking paths can handle it in hypertext. Note also that the cases where cross-references are used in a book to circumvent linear sequence are handled in exactly the same way in hypertext – linear and non-linear relationships between material are identical.

The type of rich linking just discussed can only be implemented if the underlying data structures contain all the information needed. Logically, this means that every item in every text has to be explicitly coded for its relationship to every subject covered in the grammatical description. This is an extremely onerous job, although there are undoubtedly some possibilities to automate the coding by triggering mark-up from, for example, word classes. There are additional problems to be faced in deciding how to deal with syntactic description. Is it necessary to represent syntactic structure in the data structures in order to ensure that the linking paths needed will exist? And how should links to syntactic units be implemented: should each word of the unit have a link, should the head word of the unit have a link, or should the link be attached to some abstract location in the text? These are complex questions which I leave for further research.

Following from the point made in the previous paragraph, I would like to suggest that a highly desirable feature in a multimedia grammar will be the possibility for the user to have some control over the amount of information recovered via certain links. ODKY uses concordance lists in its dictionary, and NLE has the possibility of generating lists of words sharing certain features. Such functionality for grouping and recovering data is obviously useful and desirable, but it has to be handled with care. It is not particularly useful (at least in most circumstances) to have to negotiate a list of all the occurrences of a plural marker, for example (as mentioned previously, in ODKY the list for a common item has several hundred entries). But such information will always be recoverable, given the nature of the coding that I have just argued is needed in underlying data structures, and it is certainly possible to imagine situations in which such exhaustive information will be exactly what the user wants. The ideal solution would therefore seem to be to allow the user to control the amount of information which is retrieved by some functions, rather than having exhaustive lists generated as a default.

4.Conclusion

The presentation of grammatical description as a multimedia object is potentially an extremely exciting development for linguists and others interested in language data. However, the design problems which must be faced, both conceptual and implementational, are complex.

Here, I have discussed some of the conceptual issues on the basis of an examination of three existing multimedia language resources. Four design guidelines have been identified from this process: making the presentation data-centric, allowing for complex and multiple paths of linking through the available information, the necessity for very detailed coding in underlying data, and the need for the user to control the level of detail presented in some cases. These guidelines are certainly not sufficient to give solutions to all the problems which will be encountered in constructing a multimedia grammar, but I believe that they will be of assistance to anyone who undertakes such a project.

I have touched on several problems for the implementation of a multimedia grammatical description above, such as the structure of the data storage to be used, the nature of the presentation software, and the importance of achieving a platform-independent solution, preferably using open-source software. I also consider that the relationship between a multimedia grammatical description and a printed version of some of the material is a problem of implementation. The ideal solution will be that the traditional book grammar can be easily derived from the multimedia product via some type of transformation process. That is, the textual parts of the description should be exportable into a format suitable for printing, with examples and internal cross-references generated as part of the export process. This goal is an additional goal which should be kept in mind in the

design phase of any attempt to construct a multimedia grammar.

Acknowledgements

This research is supported by a Monash University Grant under the Arts/IT Small Grants Scheme and is part of a collaborative project with John Hurst. School of Computer Science and Software Engineering, Monash University. I am grateful to Mark Donohue, Nick Thieberger and two anonymous reviewers for helpful comments.

,

References

Amith, Jonathan D. n.d. *Nahuatl Learning Environment* <http://nahuatl.ldc.upenn.edu/> (Login as 'guest', password 'nahuatl')

Austin, Peter K. 2002. Developing interactive knowledge bases for Australian Aboriginal languages – Malyangapa. MS, University of Melbourne, 14pp.

Blake, Barry J. 1985. Review of Heath 1984. *Australian Journal of Linguistics* 5:304-310

Csato, Eva A. and David Nathan. 2003. *Spoken Karaim*. (CD-ROM)

Csato, Eva A. and David Nathan. 2004. Multimedia and documentation of endangered languages. In Peter K. Austin (ed) *Language Description and Documentation* Vol.1, 73-84. London: SOAS.

Haiman, John. 1986. Review article on Heath 1980, 1982, 1894. *Language* 62: 654-663

Heath, Jeffrey. 1980. *Nunggubuyu Myths and Ethnographic Texts*. Canberra: Australian Institute of Aboriginal Studies.

Heath, Jeffrey. .1982. *Nunggubuyu Dictionary*. Canberra: Australian Institute of Aboriginal Studies.

Heath, Jeffrey. 1984. *Functional Grammar of Nunggubuyu*. Canberra: Australian Institute of Aboriginal Studies

Himmelmann, Nikolaus P. 1998. Documentary and descriptive linguistics. *Linguistics* 36:161-195

Nikolaeva, Irina and Thomas Mayer. 2004. Online Documentation of Kolyma Yukaghir. <http://ling.uni-konstanz.de/pages/home/nikolaeva/documentation/intro.html>

	Components	Delivery	Screen Presentation	Annotation	Data Grouping and Recovery
ODKY	Text (+ audio) Dictionary Images	Web browser	Single screen	Minimal	Concordance
SK	Text (+ audio) Dictionary Annotations Images	Custom environment	Split screen	Medium, more cultural notes rather than language material	None
NLE	Text (+ audio) Dictionary Annotations Images	Web browser	Single screen with pop-ups and roll-overs	Medium–high, language and general material	Query function to lexical database

Table 1 – Summary of features of three multimedia language resources

ODKY: Nikolaeva, Irina and Thomas Mayer. 2004. Online Documentation of Kolyma Yukaghir.
SK: Csato, Eva A. and David Nathan. 2003. *Spoken Karaim*. (CD-ROM)
NLE: Amith, Jonathan D. n.d. *Nahuatl Learning Environment*

Structuring Documents Efficiently

Robert Marshall, Steven Bird and Peter J. Stuckey*
Department of Computer Science and Software Engineering
University of Melbourne, Victoria 3010, Australia
{robertgm,sb,pjs}@csse.unimelb.edu.au
*NICTA Victoria Laboratory

Abstract

Documents are typically marked up to enable rendering and to facilitate reuse. However, retargetting a document often requires pervasive changes to the markup. Power et al. have proposed a new level of representation called *document structure* which captures just those aspects of graphical organisation that are significant for conveying meaning. These document structures can be generated automatically from *rhetorical structures*, abstract representations of the meaning of a text. The mapping is highly indeterminate, being governed by a large number of interacting constraints. We present a constraint programming approach to the problem, and report on early experiments with an implementation in Prolog.

1 Introduction

Documents are typically marked up to enable rendering and to facilitate reuse. Simple adjustments in layout and style can be implemented without touching the source document. However, retargetting a document often requires pervasive changes to the markup itself — e.g. changing a bulleted list to an inline list — a fact which suggests that the markup is not sufficiently abstract.

Recent research by Power et al. (2003) has identified a new level of representation called *document structure* which captures just those aspects of graphical organisation that are significant for conveying meaning. These representations can be generated automatically from *rhetorical structures*, abstract representations of the meaning of a text.

Different document structures corresponding to the same rhetorical structures represent different realizations of the text. We may want to consider different document structures for the same rhetorical structure for a number of reasons. One document structure may be easier to understand than another for the same rhetorical structure. For example a bulleted list usually provides a clearer separation of which items form part of the list than an inline comma separated list. Alternatively, some document structures may have a much more compact representation which may be essential for viewing on a PDA screen with limited size, a constraint that is irrelevant when viewing the same document on a large screen. In this paper we concentrate on finding document structures that minimize the number of "defects", a somewhat artificial measure of comprehensibility flaws from Power et al. (2003).

The mapping from rhetorical structure to document structure is highly indeterminate, being governed by a large number of interacting constraints. The existing implementation method is to generate all possible document structures corresponding to a given rhetorical structure, evaluate them against the constraints, and find the best solution (minimizing "defects"). However, this method does not scale since the search space is exponential in the size of the document.

We present a constraint programming approach to the problem, in which an objective function is stated in advance in order to greatly prune the search space. We report on early experiments with an implementation in SICStus Prolog.

This paper is organized as follows. First, in §2 we review the work of Power et al. (2003) on document structure, the mapping from rhetorical structure, and the scoring metrics. Next, in §3 we report on our constraint programming implementation, before reporting on the results of our experimental work in §4. We close by presenting our conclusions and identifying issues for future investigation.

2 Review of document structure

Natural language generation systems produce formatted text from abstract meaning representations. Power et al. (2003) have demonstrated that the graphical organization of this text – e.g. its headings, fonts, and linebreaks – can convey meaning. They propose to capture those aspects of graphical organisation which carry meaning using a new level

Proceedings of the Australasian Language Technology Workshop 2005, pages 120–126,
Sydney, Australia, December 2005.

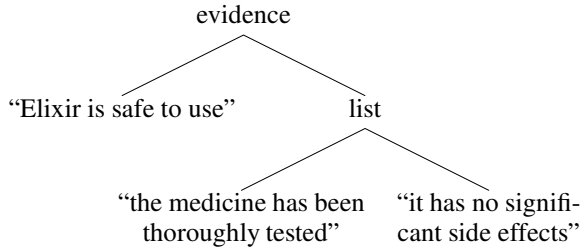
Elixir is safe to use since

- the medicine has been thoroughly tested,
- it has no significant side effects.

(a) Bulleted List Format

The medicine has been thoroughly tested; it has no significant side-effects. Therefore, Elixir is safe to use.

(b) Inline List Format



(c) Common Rhetorical Structure

Figure 1: Two formats for an excerpt from a patient information leaflet, and the underlying rhetorical structure, from (Power et al., 2003).

of representation called *document structure*. Power et al. (2003) define document structure as "the organization of a document into graphical constituents like sections, paragraphs, sentences, bulleted lists, and figures; it also covers some features within sentences, including quotation and emphasis." They argue that the same document structure can be rendered into formatted text in multiple ways, as illustrated in the patient information leaflet text in Figure 1(a) and 1(b).

Rhetorical structures represent the meaning of a text independently of its realization as a document (see Figure 1(c)). Power et al. (2003) use logic programming techniques to convert rhetorical structures into document structures which realise a given rhetorical structure. This generates a large number of candidates, and these are evaluated for conformance to a variety of heuristics such as "satellite precedes nucleus."

2.1 Rhetorical structure

Rhetorical structure is intended to represent the meaning of a text independently of its realisation as a document (Mann, 1999). Two documents with different formatting, using different words—or even written in different languages—could have the same rhetorical structure.

New World Guide to Wines	New World Guide to Wines

Figure 2: Example of how layout affects meaning (Power et al., 2003)

Rhetorical structure is expressed by rhetorical relations, which describe the relationship between facts, or between other rhetorical relations. A rhetorical relation consists of a type and some parameters, e.g. `justify(A, B)` has type `justify` and parameters `A` and `B`, and has the interpretation that we believe `A` to be true based on evidence `B`. Parameters may be other rhetorical relations, or they may be elementary propositions which are not dependent on any other information. Rhetorical relations are divided into two categories. Nucleus-satellite relations generally take two parameters: the nucleus is the central piece of information, and the satellite supports it (e.g. `justify`). Multinuclear relations can take many parameters, each of which is of equal importance to the others (e.g. `list`).

2.2 Document structure

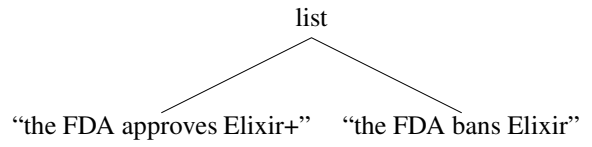
The theory of document structure was originally proposed by Power et al. (2003). The central insight is that the layout of a document affects its meaning. A simple illustration of this point appears in Figure 2 (Scott, pers. comm.). Power et al. (2003) contend that all texts have layout, even if it is very basic.

Document structure is related to markup languages such as HTML and LaTeX, which allow us to describe the structure of a document independently of its presentation. However, such markup languages are only suggestive of document structure, for they inconveniently blur the distinction between descriptive and presentational markup—c.f. (Coombs et al., 1987).

The formal theory of document structure is based around document units. *Document units* are elements such as phrases, sentences, paragraphs and chapters. Each unit can be made up of one or more sub-units, giving rise to a tree structure.

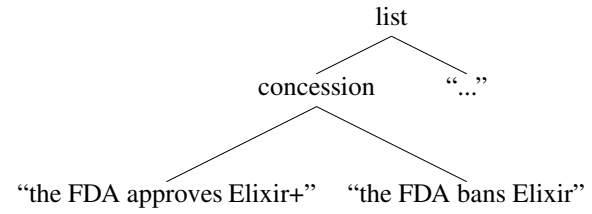
A document unit is represented by the following four variables: level, indentation, position, and connective.

Level: This represents the level of importance of the unit in the realised document. It is an integer from zero to five, corresponding to a realisation as



- The FDA approves Elixir+, and
- the FDA bans Elixir

(a) Bulleted List Format



- The FDA approves Elixir+, but the FDA bans Elixir.
- ...

(b) Inline List Format

Figure 3: Two different rhetorical structures that lead to level zero and indentation one for the elementary units.

a phrase (comma-terminated), clause (semicolon-terminated), sentence, paragraph, section, or chapter, respectively. In Figure 1, the children of the `list` relation are level zero in the first realisation, and level one in the second realisation.

Indentation: A document unit may be indented from its parent, allowing such items as bulleted lists. Power et al. (2003) represent indentation as an integer, indicating how many times a unit has been indented relative to the root node. In Figure 1, the list is indented in the first realisation, but not the second. However, we contend that the most important feature is not the total amount of indentation in an element, but rather whether it is indented relative to its parent.

If an element is indented relative to its parent, it must be realised on its own line and with its own bullet point, whereas if its parent is at the same level, it can be part of a larger structure, which is all indented. Consider the two rhetorical structures and realisations shown in Figures 3(a) and 3(b).

In both cases, the elementary units have level zero and indentation one, meaning that they are realised as phrases which are terminated with a comma, and are indented once from the base. However, in the first case they are rendered on separate lines, while in the second they are on the same line.

The reason for this is that in the first case, they are list elements, while in the second, they are both part of a concession. It is clear that when list elements are indented, they should each be realised on a separate line, but this is not actually spelled out in the document structure. While this can be inferred from the relative indentations of the elementary units to their parents, it adds unwanted complexity to the implementation, as well as violating the modularity of the document units.

Accordingly, we define `IndentationHere` as a binary variable, indicating whether a given document unit is indented relative to its parent. Note that the indentation at any given node is just the sum of the `IndentationHere` variables for it and each of its ancestors.

Position: The position variable indicates the order in which the nucleus and satellite occur. The two realisations in Figure 1 show alternate positionings of the nucleus and satellite of the `evidence` relation. Because all of the children of a multinuclear relation are of the same importance, we have the freedom to reorder them in any way. But for the purposes of this paper these reorderings will not change the evaluation of the realised document, hence they are always rendered in the same order that they occur in the rhetorical structure.

Connective: A discourse connective is always used to denote the rhetorical relationship between different document units. The only exception is for the leaf nodes of the document structure, which represent basic propositions. Figure 1 shows two different connectives for the `evidence` relation.

2.3 Discourse connectives

Discourse connectives indicate the rhetorical relationships between consecutive spans of text in a discourse.

Power et al. (1999) model discourse connectives with four attributes: relation, locus, phrase and syntactic type. These specify: the rhetorical relation which the connective represents; whether it should be attached to the nucleus or satellite; the text which realises it; and the syntactic restrictions on where the connective can be used. They side-step the question of whether a discourse connective should be realized overtly, insisting that there be a one-to-one relationship between rhetorical relationships in the rhetorical structure and discourse connectives in the generated text.

They define three syntactic types: parenthetical, coordinating or subordinating. Coordinating

The FDA bans Elixir; however, the FDA approves Elixir+.

(a) A parenthetical connective

Although the FDA bans Elixir, the FDA approves Elixir+.

(b) A coordinating connective

The FDA bans Elixir, but the FDA approves Elixir+.

(c) A subordinating connective

Figure 4: Examples of the different types of discourse connectives

connectives force the children to be of level zero. Subordinating and parenthetical connectives require that the satellite to appear before the nucleus, with parenthetical connectives additionally forcing the children to be of level greater than zero.

For example, the rhetorical relation *concession* can be realised by the discourse connectives *however*, *although* and *but*. These connectives are of parenthetical, coordinating and subordinating types, respectively. We give three different realisations of the same rhetorical structure in Figure 4. Note that the order of the two elements is reversed for the coordinating connective, and the different levels (sentences, phrases or clauses) used in each example.

2.4 Generating a document structure

Power et al. (2003) examine the task of determining a document structure from a given rhetorical structure, which they call "document structuring" (Power, 2000; Power et al., 2003), and implement it in a system called ICONOCLAST (Integrating Constraints on Layout and Style). The input consists of a collection of simple propositions organized into a rhetorical structure tree. The *document structurer* arranges these into a coherent collection of paragraphs, text-sentences and the like. This is then converted into an actual document by a *syntactic realiser*.

Each node on the rhetorical structure tree corresponds to a node on the document structure tree. The four variables associated with the node are constrained in various ways. First, the level of a child unit must be less than or equal to that of its parent, and equal to that of its siblings. The only exception is when the child unit is indented, in which case its level is independent of its parent, although it must still be equal to that of its siblings. Second, the indentation of a child unit must be either equal to or one greater than that of its parent. The positions of siblings must obviously be distinct. Third, the connective must realise the rhetorical relation. Finally, the type of the connective places additional constraints on the level and position of the children of the current unit, which must be satisfied.

Power et al. (2003) implement this process using logic programming. The different choices in mapping the rhetorical structure to the document structure are represented by constraints in the logic program, with the exception of discourse connectives. These set choicepoints, as in a standard logic program.

Moreover, they evaluate the defects of a structure (as described in the following section) after it has been completely generated. These factors force them to generate all possible document structures for a given rhetorical structure, using both a branch-and-bound and standard Prolog backtracking, before they can choose the best structure.

There are exponentially many (in the number of nodes in the rhetorical structure) candidate documents for any rhetorical structure. The number of candidates quickly blows up so that the approach is impractical for rhetorical structures with more than 10 nodes.

2.5 Scoring document structures

In order to choose between different document structures, Power et al. (2003) generate all valid document structures, then score each one by counting its undesirable features. The larger the score, the worse the structure. They give an example of a rhetorical structure with one relation between two facts, with seven renderings, and one with three relations and four facts, which has 58 renderings. They identify six kinds of undesirable features, which we describe below.

Nucleus before satellite: The nucleus appears before the satellite. This is undesirable, according to Power et al. (2003), because of psycholinguistic evidence which suggests that the more important information should be placed at the end of a sentence (and this is the common practise in English). The first realisation in Figure 1 shows a nucleus before satellite defect.

Left-branching structure: The left side of the document structure tree branches, while the right side does not. The second realisation in Figure 1 contains a left-branching structure, as the `list` child of the root node is realised before the `elementary` child.

Lost rhetorical grouping: The document structure can conflate distinct levels of the rhetorical structure. That is, a child and parent (and possibly higher-level ancestor) nodes in the rhetorical structure can be realised at the same level in the document structure. This makes it more difficult to infer the underlying structure from the text. For example, in Figure 4 the second and third examples contain this defect, while the first does not.

Single-sentence paragraph: A paragraph contains only one sentence.

Oversimple text-clauses: A sentence is composed of two text-clauses (clauses separated by a semi-colon), each of which expresses a single elementary proposition. The first sentence of the second realisation in Figure 1 contains is an oversimple text-clause.

Repeated discourse connective: A single rhetorical structure is represented twice in the document structure, by the same connective, and in such a way that one of the occurrences is on a descendent node of the other.

2.6 Summary

Rhetorical structures represent the meaning of a text independently of its realisation. Document structures include those realisation details which are relevant to its meaning. A document structure can be generated from a rhetorical structure using a constraint-based approach as described in (Power et al., 2003), but the current implementation is too inefficient to be used on large rhetorical structures.

3 Constraint Programming Implementation

Constraint programming (Marriott and Stuckey, 1998) allows us to specify relationships between variables, without having to actually calculate the values that the variables may take. In this work we use constraint logic programming over finite domains (Van Hentenryck, 1989), which augments a traditional logic programming language with the capacity to apply mathematical constraints over Boolean and integer variables over fixed ranges.

In order to express the complex constraints that arise in defining document structure and defects we make use of *reified constraints*. These allow us to attach a Boolean variable to the result of a constraint. For example $B \Leftrightarrow X > 3$ is a constraint that holds if $B = 1$ and X takes a value greater than 3, or $B = 0$ and X takes a value less than or equal to 3.

We implemented a document structurer in SICStus Prolog, using the same constraint model as Power et al. (2003). However, our program differs in that it simultaneously evaluates both the required constraints to create the document structure, and the constraints required to find the defects. It produces as output both a document structure and a count of its defects. By contrast, the document structurer of Power et al. (2003) produces a document structure which must then be evaluated. Moreover, our document structure contains positioning information for each word in the output.

Mapping a rhetorical structure to a document structure involves a large number of independent choices. As the rhetorical structure grows, the number of corresponding document structures grows exponentially. We would like to choose only those with less than a fixed number of defects, or perhaps the one having the fewest defects. Generating all possible document structures, to only choose one, is both unnecessary and impractical as the number of candidates grows so rapidly.

Our constraint programming model constrains the count of defects while we are generating document structures. This allows us to stop generating a structure as soon as it has more defects than the upper limit (for minimization this limit is defined by the number of defects in the best answer so far). Moreover, because a partially generated document structure may in fact lead to several document structures, each of which will have at least as many defects as the partially generated structure, we can prune entire branches from the search tree.

We accomplish this by expressing the rules for the defects as constraints. Each constraint is evaluated at every node on the document structure tree, indicating whether or not the defect occurs at that point. We simply sum them all to obtain the total count of defects found so far. The constraints are stored for each node, along with the other document structure parameters. The rules for the nucleus before satellite and left-branching defects are given below. In the following, PN and PS are the positions of the nucleus and satellite nodes, while EN and ES are Boolean variables indicating whether the nucleus and satellite are elementary.

$$PN < PS \;\Rightarrow\; NucleusBeforeSatellite$$

$$\begin{array}{l}(EN \wedge \neg ES \wedge PS < PN) \\ \vee (ES \wedge \neg EN \wedge PN < PS)\end{array} \Rightarrow LeftBranching$$

Because the total number of solutions to be checked is exponential in the size of the structure, the problem can easily become intractable for large

structures. Therefore, we set a maximum number of assignments which can be made, and simply fail to find any solutions after this point, and return the best solution which has been found thus far.

3.1 Improving the searching efficiency

Expressing the entire model in a constraint-based form allows us to search for a solution with the fewest defects more quickly than a non constraint-based implementation. However, the search space is still very large, so we have implemented several other techniques to improve the searching.

Labelling order: Because labelling one type of variable will affect the domain of others, the order in which variables are labelled can make a significant difference. For example, if we set the connective to a coordinating type, then the levels of any child nodes must be zero, while the converse does not necessarily hold. Therefore, at least in this case, it will be more efficient to label the connective before the level. The relationships between the variables are relatively complicated, so it is unclear what the best labelling order is without conducting some experiments. We test two methods for ordering variables. One approach is to label the document structure by traversing the tree, depth-first. We make one pass for each variable type. For example, we might first label all the `Indentation` variables, then all the level variables, and so on.

The other method we use is known as first fail labelling (Haralick and Elliott, 1980). Using this method, we label the variables in order of domain size, starting with the smallest. This is intended to reduce the amount of branching which occurs, and prune large regions of the search space as early as possible. Either of these methods can be used in conjunction with any of the following strategies.

Iterating through the goal variable: One technique which can improve efficiency is to iterate through the variable to be minimised, starting from zero, attempting to find a solution at each value.

While it may seem inefficient, this procedure can sometimes be faster than a simple search, because the goal variable is constrained to just one value for each call. Moreover, as soon as one of the calls succeeds, we are guaranteed to have the minimum value of the goal variable. Using a simple minimisation search, it is generally not immediately clear if a given solution is in fact the minimum, requiring further search.

Limited-discrepancy search: Limited-discrepancy searching (Harvey and Ginsberg, 1995) requires a heuristic, which guesses the best value for each labelling choice. Any choice which differs from the heuristic is called a discrepancy. Using this method, we search as before, but with the number of discrepancies limited to some upper bound. This has the effect of reducing the search space, potentially rejecting many possible solutions, but also allowing for a much faster search.

Our heuristic works as follows. If the level variable is not already set, we choose the second largest possible value for it. We are attempting to avoid two nodes having the same level, thereby incurring a lost rhetorical grouping defect. The larger the value, the more values will be available for descendant nodes to use, but the largest possible value will generally be the same as its parent.

We always choose to indent multinuclear relations, thereby allowing children to take any available level, and place the satellite first, avoiding a nucleus before satellite defect.

Subordinating connectives place the least restrictive constraints on the connected nodes, followed by parenthetical and then coordinating connectives. In particular, we do not wish to use a coordinating connective, as it forces the nucleus and satellite (and hence any child nodes they may have) to be of level zero, and will incur a lost rhetorical grouping defect on all further child nodes. Hence, we choose a subordinating connective if one is available; otherwise a parenthetical one, and finally a coordinating connective.

Unfortunately, it is unclear how many discrepancies to allow when using limited-discrepancy searching. We have tested our implementation using a maximum of both 3 and 10 discrepancies, but these choices are quite arbitrary.

Optimistic partitioning: Optimistic partitioning (Prestwich and Mudambi, 1995) requires no further information except the total range of the search space, which is from zero to the total number of nodes multiplied by 6 (the number of defect types). We split the search space in two and search for a solution in the first half of the space. If one is found, we partition again, using this solution as the new upper bound. If there is no solution in the first half, we search again, in the upper half of the search space. If there is a solution in the upper half, we partition again, using the midpoint and the solution in the second half of the search space as the lower and upper bounds, respectively.

In either case, we store the current solution as the best one thus far, and if at any time we fail to find a solution in the given range, then we know that the previous best solution is the overall minimum.

Restricting the number of assignments: A final, crude method of improving performance is to simply restrict the total number of assignments which can be made. Once we have reached this limit, we can simply return the best solution found up to that point. While it is preferable to find the best solution, the search space is exponential, and there may be cases in which this problem is intractable.

4 Results

Our testing data comes from Marcu's rhetorical structure corpus (Marcu, 2000). We have chosen a particular article from his corpus, and rendered the individual rhetorical structures which make up this article.

The rhetorical relations used in this corpus generally do not specify corresponding discourse connectives, so we used random placeholders. While this produces rather ugly output, it is sufficient for the purposes of testing the efficiency of the various labeling techniques.

As described in §3.1, we iterate through the document structure tree several times, once for each variable type. We have tested all 24 possible variable orderings. The numbers given in Table 1 represent the number of variable assignments required to find the optimal solution, divided by the number of variable assignments required by the most efficient variable ordering for the same structure.

The order in which variables are labelled has a large impact on the efficiency of the search, by a factor of about 4 on our testing data. The fastest method seems to be to label the $Level$ and $Position$ variables first, and then either of the other two. For the next table, we used the "Level,Position,IndentationHere,ConnectiveIndex" labelling strategy.

We have tested all of the search strategies discussed in the previous section, using both the tree-traversal and first-fail methods of choosing which variable to label. We forced all searches to terminate after 100,000 assignments and return the best solution found at that point, in order to prevent inordinately long execution times. Results are shown in Tables 2 and 3. We show the number of nodes in the rhetorical structure, **Size**, as well as the minimal number of defects possible (if known), **Min Defects**. The remaining columns give the number of assignments required to find the best solution, except the column **First** which gives the number of assignments to find the first solution with simple search. **Basic** is the simple minimization search, **Iterating** iterates the defect count upwards from 0, **OP** uses optimistic partitioning, **FF** is first fail search, and **LD(n)** is limited discrepancy search with a max discrepancy of n. If the best solution found is not optimal it is shown in parentheses after the number of assignments figure. A dash indicates no solution was found with 100,000 assignments. Note that we have not normalised these results, as it is sometimes unclear which is the best solution; some strategies may perform faster than others but return a non-optimal solution, and some are terminated early for the sake of tractability.

For the search strategy, limited-discrepancy search and iterating through the defect variable (using tree-traversal labelling) seem to require the least number of assignments in order to find a solution. However, the best search strategy varies considerably, depending on the structure. However, if we reach the maximum number of assignments without finding an optimal solution, iteration cannot provide a sub-optimal solution, because by definition, the first solution it finds is the optimal one. For this reason, limited-discrepancy may be a better choice when realising large structures.

Limited-discrepancy search often does not find a solution when used with first-fail labelling. We believe that this is because first-fail labelling will choose labelling variables from all over the document structure. Therefore, once a discrepancy has been incurred, the next variable to be labelled may be from an entirely different part of the document structure. This may happen several times, causing several discrepancies to occur, before the choices which have been made can affect new ones.

By contrast, when using tree-traversal labelling, once a discrepancy has been incurred, the next variables to be labelled will come from the same node, or its children, which will be constrained by the choice which has just been made.

Tree-traversal labelling seems to outperform first-fail labelling in most other cases too, but much less dramatically.

Basic search performs more poorly than the other methods in most cases, as we might expect. Optimistic partitioning represents an improvement on this, but not as much as limited-discrepancy or iterative searching.

Finding the first solution is generally quite fast, but the first solution is almost never optimal. It is also interesting to note that even merely finding the first solution is often slower than finding the optimal limited-discrepancy solution, indicating that our heuristic is improving performance considerably.

Overall, the iterating approach followed by a limited discrepancy search when this fails to find the

Ordering	Max	Min	Average
IndentationHere,Level,ConnectiveIndex,Position	57.1	1.00	7.52
IndentationHere,Level,Position,ConnectiveIndex	1.78	1.00	1.28
IndentationHere,ConnectiveIndex,Level,Position	46.9	1.00	6.24
IndentationHere,ConnectiveIndex,Position,Level	19.0	1.00	3.15
IndentationHere,Position,Level,ConnectiveIndex	2.79	1.00	1.38
IndentationHere,Position,ConnectiveIndex,Level	2.65	1.00	1.47
Level,IndentationHere,ConnectiveIndex,Position	57.2	1.00	7.52
Level,IndentationHere,Position,ConnectiveIndex	1.75	1.00	1.29
Level,ConnectiveIndex,IndentationHere,Position	116	1.00	14.04
Level,ConnectiveIndex,Position,IndentationHere	57.1	1.00	7.50
Level,Position,IndentationHere,ConnectiveIndex	1.75	1.00	1.27
Level,Position,ConnectiveIndex,IndentationHere	1.75	1.00	1.27
ConnectiveIndex,IndentationHere,Level,Position	65.0	1.00	8.38
ConnectiveIndex,IndentationHere,Position,Level	37.1	1.00	5.38
ConnectiveIndex,Level,IndentationHere,Position	74.1	1.00	9.20
ConnectiveIndex,Level,Position,IndentationHere	46.9	1.00	6.18
ConnectiveIndex,Position,IndentationHere,Level	19.0	1.00	3.24
ConnectiveIndex,Position,Level,IndentationHere	19.0	1.00	3.08
Position,IndentationHere,Level,ConnectiveIndex	4.56	1.00	1.65
Position,IndentationHere,ConnectiveIndex,Level	3.82	1.00	1.73
Position,Level,IndentationHere,ConnectiveIndex	2.40	1.00	1.31
Position,Level,ConnectiveIndex,IndentationHere	2.40	1.00	1.31
Position,ConnectiveIndex,IndentationHere,Level	3.08	1.00	1.47
Position,ConnectiveIndex,Level,IndentationHere	3.08	1.00	1.39

Table 1: Comparative number of variable assignments required to find optimal solution for various labelling strategies.

Size	Min Defects	First	Basic	Iterating	OP	LD(3)	LD(10)
3	0	15 (2)	29	15	30	14	14
28	2	699 (20)	100000 (3)	—	100000 (4)	21008 (9)	100000
45	?	—	—	—	—	20666 (17)	100000 (11)
17	1	220 (12)	1167	392	1062	1677 (5)	1115
10	0	50 (6)	197	50	150	209	232
13	6	194 (9)	2626	2617	2953	455	1504
31	3	2310 (26)	100000 (8)	—	100000 (12)	100000 (6)	100000 (6)
3	0	15 (1)	20	15	30	16	16
13	0	136 (9)	520	65	436	350	345
8	0	40 (5)	143	40	120	91	91
5	3	25 (4)	53	49	74	28	28
7	5	35 (6)	85	117	151	46	50
9	1	45 (7)	189	45	135	93	93
21	1	596 (16)	42410	40363	41634	9155	39894
3	0	15 (2)	29	15	30	14	14
15	0	214 (13)	1137	75	697	92	92

Table 2: Comparison of number of valuations required by different searching strategies, using a tree-traversal system for choosing labelling variable.

solution appears to be the most robust combination. By comparison, the prototype system demonstrating the work in Power et al. (2003) generates all solutions and then chooses the best one. This approach will not scale to large rhetorical structures, which is why we have not attempted to compare the systems directly.

5 Conclusion

We have examined Power et al's theory of document structure, and implemented their document

Size	Min Defects	First	Basic	Iterating	OP	LD(3)	LD(10)
3	0	12 (2)	14	12	24	17	17
28	2	— —	35691	—	—	—	
45	?	189 (21)	100000 (9)	—	100000 (9)	—	—
17	1	68 (7)	196	74	220	—	—
10	0	40 (4)	70	40	86	—	48
13	6	5079 (7)	9170	5923	10895	—	1009 (7)
31	3	1490 (14)	80592	40318	72914	—	—
3	0	12 (1)	14	12	24	14	16
13	0	52 (7)	110	52	216	—	105
8	0	32 (5)	65	32	96	—	35
5	3	20	84	120	120	—	87
7	5	49	297	481	393	—	287
9	1	38 (7)	160	67	168	—	55
21	1	30357 (13)	46594	250	38881	—	—
3	0	12 (2)	22	12	24	18 (1)	18
15	0	56087 (10)	91411	60	77427	—	232

Table 3: Comparison of number of valuations required by different searching strategies, using a first-fail method for choosing the labelling variable.

structuring algorithm with a superior constraint handling system. Limited-discrepancy search using tree-traversal labelling is generally the fastest of the search strategies we have tested, but may return sub-optimal solutions at times. Other techniques represent various trade-offs between completeness, likelihood of finding a solution, and the score of the solution. Any of these represent a significant improvement in the performance of document structuring for large inputs.

There are several avenues for further work in this area: constraining output to fit within a specified rectangle (e.g. for displaying text on a PDA); summarisation in which subparts of the rhetorical structure are omitted, and defects are scored depending on the nature of the omission; rendering of text together with diagrams; and larger-scale empirical testing. Ideally, we would like to obtain a larger corpus containing rhetorical structures, relations and discourse connectives.

It would also be interesting to use a more expressive formalism than rhetorical structure. Rhetorical structure has problems expressing many constructs used in everyday language, such as questions and tense, and accordingly more complicated formalisms have been developed.

Acknowledgements

We are grateful to Donia Scott for early discussions which inspired us to undertake this project, and for providing software and data. We are also grateful for the support of an Australian Postgraduate Award for Robert Marshall.

References

James H. Coombs, Allen H. Renear, and Steven J. DeRose. 1987. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947.

R. M. Haralick and G. L. Elliott. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313.

William D. Harvey and Matthew L. Ginsberg. 1995. Limited discrepancy search. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95); Vol. 1*, pages 607–615, Montréal, Québec, Canada, August 20-25. Morgan Kaufmann, 1995.

Bill Mann. 1999. An introduction to rhetorical structure theory (rst). `http://www.sil.org/~mannb/rst/rintro99.htm`.

Daniel Marcu. 2000. *The Theory and Practise of Discourse Parsing*. MIT Press.

K. Marriott and P.J. Stuckey. 1998. *Programming with Constraints: an Introduction*. MIT Press.

R. Power, C. Doran, and D. Scott. 1999. Generating embedded discourse markers from rhetorical structure. In *Proceedings of the European Workshop on Natural Language Generation, Toulouse, France.*

Richard Power, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document structure. *Computational Linguistics*, 29(2):211–260, June.

Richard Power. 2000. Planning texts by constraint satisfaction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000), Saarbruecken, Germany*, pages 642–648.

Steven Prestwich and Shyam Mudambi, 1995. *Improved branch and bound in constraint logic programming*, pages 533–548.

Pascal Van Hentenryck. 1989. *Constraint Satisfaction in Logic Programming*. MIT Press.

Round-Trip Translation: What Is It Good For?

Harold Somers
School of Informatics
Manchester University
Manchester, UK
Harold.Somers@manchester.ac.uk

Abstract

This paper considers the popular but questionable technique of 'round-trip translation' (RTT) as a means of evaluating free on-line Machine Translation systems. Two experiments are reported, both relating to common requirements of lay-users of MT on the web. In the first we see whether RTT can accurately predict the overall quality of the MT system. In the second, we ask whether RTT can predict the translatability of a given text. In both cases, we find RTT to be a poor predictor of quality, with high BLEU and F-scores for RTTs when the forward translation was poor. We discuss why this is the case, and conclude that, even if it seemed obvious that RTT was good for nothing, at least we now have some tangible evidence.

1 Introduction

Macklovitch (2001:27) talks of "the spectacular growth and pervasiveness of the World Wide Web" leading to a "democratization" of Machine Translation (MT) which has "profoundly transformed the MT business". The availability of free on-line MT systems, since CompuServe's initial experiments in 1994 (Flanagan, 1996) and then more significantly AltaVista's collaboration with Systran from 1997 onwards (Yang and Lange, 1998), has indeed revolutionized the MT world, creating a whole new and significantly large community of users, mostly with little or no knowledge or understanding of how MT works or even, in some cases, how *language* works. Such users are, nevertheless, keen to know how good MT output is, and frequently resort to the intuitive technique of 'round-trip translation' (RTT), or 'back-and-forth translation', in which they take a given text or sentence, have it translated into some foreign language by the MT system (the 'forward translation', henceforth FT), then have it translated back into the original language by the same system (the 'back translation', BT).

Popular articles on MT by journalists and other lay-users all too frequently use this technique to 'evaluate' MT, with results which are, depending on your predisposition, hilarious or infuriating. A recent example is from the *Biomedical Translations* website (Anon, 2003), where the author explains the technique, and suggests that "In theory, the back translated English should match the original English." Several garbled examples are then given, and the article concludes "Would you trust your surgeon using these instructions?" Another website recognizes the problem "Machine translations can produce text that is garbled or hilariously inaccurate", and suggests as a resolution "Test the precision of your translated text by sending a phrase on a round trip through the translation engine." (Anon, 2005).

The dangers of this approach have long been appreciated: for example Huang (1990), addressing the problem of evaluating output when you do not know the target language, describes it as the "seemingly most natural way" to evaluate a translation, but quickly warns that the results are not reliable. More recently, O'Connell (2001) gives the following sound advice on an IBM website:

> "A common misunderstanding about MT evaluation is the belief that back translation can disclose a system's usability. […] The theory is that if back translation returns [the source language] input exactly, the system performs well for this language pair. In reality, evaluators cannot tell if errors occurred during the passage to [the target language] or during the return passage to [the source language]. In addition,

Proceedings of the Australasian Language Technology Workshop 2005, pages 127–133,
Sydney, Australia, December 2005.

> any errors that occur in the first translation […] cause more problems in the back translation."

So, although it is widely agreed in the MT community that RTT is a bad technique, and equally widely suggested in the lay community that it is an effective way to evaluate systems, there has been little or no work to demonstrate empirically whether RTT is in fact as misleading as it is claimed.

In the next section we will briefly review the reasons why one might be wary of RTT as an indicator of MT quality: while one can cite anecdotal evidence of bad round trips, we can ask whether on a larger scale RTT might after all be indicative at least of general trends. In this regard, given the situation that lay users finds themselves in, we will consider two issues of concern to them: Which is the best MT system to use? And how machine-translatable is my text? In Sections 3 and 4 we will present two experiments that take this user's need, and explore whether RTT can meet it, or not.

2 Two perspectives on RTT

2.1 Why RTT might not work

As O'Connell (2001) states in the earlier quote, and as other commentators have pointed out, RTT could be misleading for three reasons:

First, if the round trip is bad, you cannot tell whether it was the outward journey or the return trip where things went wrong. For example, (1) shows an RTT from English to Italian and back again using Babelfish. The resulting BT (1c) is garbled, but in fact apart from a possible gender error (loan words usually take the gender of their literal translation, so *Home Page* should probably be feminine) the forward translation into Italian is really quite acceptable.

(1) a. Select this link to look at our home page.
b. *Selezioni questo collegamento per guardare il nostro Home Page.*
c. Selections this connection in order to watch our Home Page.

Of course, if it is the outward journey that is bad, then the return trip could be bad, but it might be disproportionately so, because of the old maxim 'garbage in garbage out'.

However, and this is the second point, a bad FT can nevertheless lead to a quite reasonable BT. So the fact that the round trip gives a good result does not necessarily tell you anything about the outward journey. This can be illustrated in (2), again using Babelfish, where the idiomatic phrase is translated literally into meaningless Portuguese (2a) and then 'perfectly' back into Enlgish (2c).

(2) a. tit for tat
b. *melharuco para o tat*
c. tit for tat

The third point is that of course the basic premise of RTT is flawed: even a pair of human translators would not be expected to complete a perfect RTT, in the sense that the return translation would be word-for-word identical to the original source text.

2.2 Why RTT might appear to work

So, it is easy to show RTT not working. But equally we should acknowledge that sometimes, RTT *does* appear to work, producing a quite understandable paraphrase and, if only we knew it, a reasonable translation on the way. Examples (3) and (4), translated by Freetranslation, illustrate this.

(3) a. The spirit is willing but the flesh is weak.
b. *Дух желает, но плоть слаба.*
c. The spirit wishes, but the flesh is weak.

(4) a. Once, a jolly swagman camped beside a billabong.
b. *Однажды, веселая бродяга разбила лагерь около устья реки.*
c. Once, the cheerful tramp has broken camp about a mouth of the river.

The conclusion is that *for a single given sentence*, we cannot know for sure if a good (or bad) RTT indicates that the FT was good (or bad) or vice versa. But it is a not unreasonable hypothesis that over the length of a longer text, average RTT quality might reflect the general quality of the system used. This will be the subject of our first experiment, reported in Section 3.

2.3 Lay-users and MT

For some time now, observers of MT have identified two distinct uses of MT, labelled 'for assimilation' and 'for dissemination'. The differences between the two are neatly summarized in Table 1. Until now, use of free on-line web-based MT services has been assumed to lie more or less firmly on the 'assimilation' side. However, as the availability of the service has become well known, we now see a lot of web pages with explicit links to

Assimilation	Dissemination
many SLs, one TL	one SL, many TLs
any style	controlled style
any topic	restricted topic(s)
poor quality OK	good quality required
post-editing if needed	no post-editing
user is reader	user is author

Table 1. Differences between MT for assimilation and MT for dissemination

MT services, which means that web-page designers now see on-line MT as a way of getting their message translated. They have become users of MT for dissemination, although typically they do not fit the profile outlined in Table 1, and it has been argued (Gaspari, 2004; Somers and Gaspari, 2005) that web-page designers need to be better educated about what MT can and cannot do.

It is not unreasonable therefore for web-page designers to seek some way of knowing how well their web pages will be translated well by free on-line MT services. There has been a fair amount of research recently on 'translatability'. (Gdaniec, 1994; Bernth, 1999a,b; Bernth and McCord, 2000; Underwood and Jongejan, 2001; Bernth and Gdaniec, 2002; O'Brien, in press). Research has focussed on identifying 'translatability indicators', stylistic or grammatical linguistic features that are known to be problematic for MT (so a more transparent name would perhaps be 'translation difficulty indicators'). For example, mid-sentence parenthetical statements or the use of the passive voice could respectively be classified as stylistic and grammatical indicators. While such measures are of use to linguists, and to designers of controlled languages, they mean little or nothing to the average lay-user. Even if RTT is not reliable on an individual sentence-by-sentence basis, might it be reliable enough to show whether an entire document is *by and large* machine translatable?

3 Experiment 1: Can RTT tell us which system is best?

Even if RTT does not always work, we might hope that the quality of the RTT will reflect the quality of the FT: if this is true, then at least RTT could be used to help lay-users to decide which system to use, when they are faced with a large number to choose from. In order to explore this hypothesis, we set up a first experiment in which we took four texts representing various language pairs, translated them each using five free on-line MT systems,[1] then translated the resulting FT back into the original language using the same system. We used two standard measures to evaluate the results, the familiar BLEU metric (Papineni et al., 2002), and Turian et al.'s (2003) F-score metric. A number of researchers have commented on the fact that BLEU scores do not always agree with the F-score, based on precision and recall, nor sometimes with human judgments, especially for shorter stretches of text (e.g. Way and Gough, in press). In addition, alternative packages available on the Web offering implementations of BLEU give significantly different results. Accordingly, we used our own implementations of BLEU and F-score,[2] and show results with both metrics; while they sometimes rank the translations differently, they both tell the same overall 'story', as we shall see.

The texts were as follows: extracts from the French web pages of the Tourist Offices of Marseilles and Barèges (a skiing resort) for translation into English, and two passages from the Europarl corpus of European Parliament Proceedings 1996–2003, one in English, for translation into German, and one in French, for translation into English. All the texts were around 100 sentences long.

Figures 1 and 2 show the BLEU and F-scores for the 20 pairs of translations, FT and BT, grouped by text, and ordered within each group. The order of the systems is different for each text, but since, for our purposes, we are only interested in seeing whether the scores for the FTs and BTs correlate, the identity of the individual systems is unimportant. The first thing to notice is that both BLEU and F-score show little correlation between the FT and BT scores. Figure 3 shows this more strikingly, as does a Pearson's coefficient of $r = -0.04$ for these scores. The F-scores (not shown here) correlate somewhat better, at $r = 0.61$, but this is nowhere near useful for our purposes.

However, we should note that, as Figures 1 and 2 show, the difference in scores for some of these systems is really quite small. Also, for two of the texts, the system with the top-ranking score for FT is actually ranked 4th or 5th for the BT.

[1] Babelfish, Freetranslation, Systran, ProMT, and Worldlingo.

[2] Thanks to Simon Zwaarts for these, and also for Python scripts used to translate on-line large amounts of material, overcoming the text-length limits imposed by many of the systems.

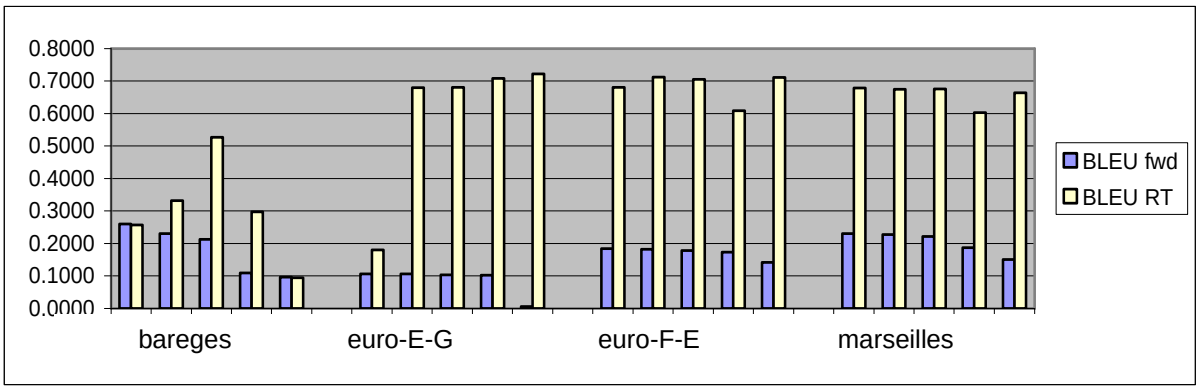


Figure 1. BLEU scores for the 20 forward and round-trip translations.

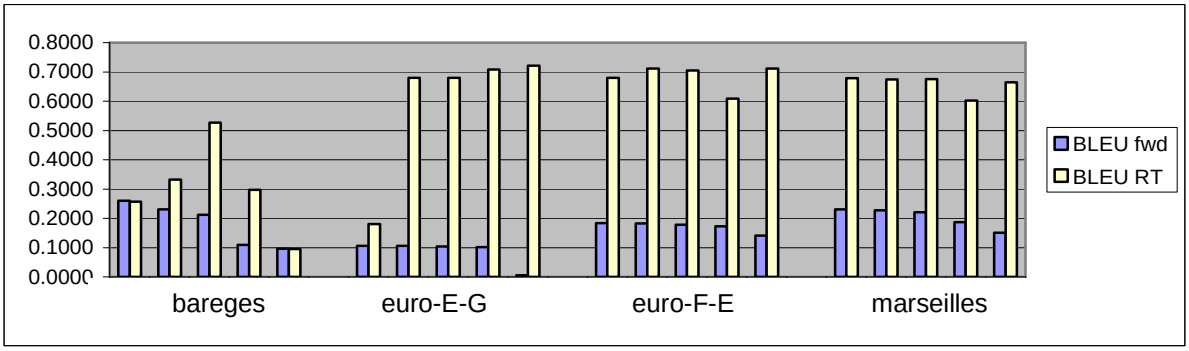


Figure 2. F-scores for the 20 forward and round-trip translations.

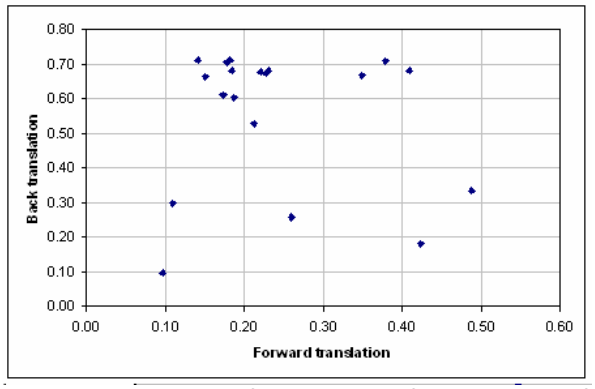


Figure 3. Correlation of BLEU scores for FT and BT. If the scores correlated well, they would cluster around the diagonal.

Our first conclusion then is that RTT is *not* a particularly good way to identify which system is better: if anything, a high-scoring BT indicates either the best or the worst system, but even this is not systematic.

What is also striking is that the BT score is often better than the FT score, and the difference is greatest when the FT score is low. Although the results do not show a consistent pattern, what is clear is that a good score for the BT generally does not necessarily 'predict' a good score for the FT; rather more often the opposite.

The reason for this is fairly easy to explain, considering how these MT systems in general work. Although systems perform source-text analysis to a certain extent, when all else fails they resort to word-for-word translation, and where there is a choice of target word they will go for the most general translation. Clearly, when the input to the process is difficult to analyse, the word-for-word translation will deliver pretty much the same words in the BT as featured in the original text. A further point to make is that the BLEU metric (and to a lesser extent the F-score) 'reward' word matches, even if the word order is somewhat scrambled. This can be illustrated with (5) which shows the source text (5a) and model translation (5b), the FT (5c) and the BT (5d). About two thirds of the words in (5a) appear in (5d); this pair of sentences alone would merit a BLEU score of 0.5882.

(5) a. All of us here are pleased that the courts have acquitted him and made it clear that in Russia, too, access to environmental information is a constitutional right.

b. *Wir freuen uns hier alle, daß das Gericht ihn freigesprochen und deutlich gemacht hat, daß auch in Rußland der Zugang zu Umweltinformationen konstitutionelles Recht ist.*

c. *Alle, von den uns hier erfreut werden, dass die Gerichtshöfe ihn freigelassen haben, und hat es reinigt gemacht, daß in Russland auch auf zu Umweltinformationen zugreift ist ein verfassungsmäßiges Recht.*

d. Everyone of which here delighted become us that the courts it released have has cleans made, and it that in Russia also on to environment information take action, is a constitutional right.

4 Experiment 2: Can RTT tell us how well our text will be translated?

In our second experiment, we wanted to see if the scores for the RTT would correlate with scores for FT when we compare texts that the MT systems translate well with texts that prove difficult. Based on the BLEU and F-scores, we took three of the texts from the first experiment, and the scores for their RTTs using Freetranslation, neither the best nor the worst of the MT systems. These 'hard' texts were the Marseilles web-page and the two Europarl examples. Against these we constructed three 'easy' texts: a children's story (*Goldilocks and the Three Bears*), some text from Canadian weather forecasts, and some typical entries from a tourist's phrase-book. We constructed the parallel *Three Bears* text and the tourist phrases from various websites. The weather bulletins come from RALI's Météo website;[3] we 'helped' the MT system by pre-editing the texts, converting the all-uppercase text to mixed case, inserting accents, and also changing *moins* and *minus* in temperature read-outs to a minus-sign. Like the 'hard' texts, the 'easy'texts were all roughly 100 lines long.

In (6) we see some examples of BTs that show that the easy texts were indeed generally well translated back and forth.

(6) a. Therefore she went in top in the bedroom where the three Bears slept, and there was the three beds.

b. Today. Cloudy with the clear periods and some snow. High close to –9. The winds of the west 15 to 30 km/h. Tonight. Cloudy with the clear periods and 30% probability of flurries.

c. Do you speak the English ? I do not speak the French. I do not understand. Please to speak slowly. I hope that you understand my English

The comparison of the BLEU scores for the FT and BT of these six texts is shown in Figure 4. The figure shows quite dramatically that, at least as far as the BLEU scores go, the easy texts are somewhat easier to translate than the hard texts; and it shows equally clearly that the score for the RTT does not reflect this at all: in fact according to the RTT score, all the texts are of about the same difficulty. The correlation between BLEU scores for the FT and BT is $r = -0.31$, while for the F-scores it is $r = 0.59$.

5 Conclusions

In this paper we have tried to demonstrate explicitly what most MT researchers already assumed: to paraphrase the words of Edwin Starr's 1970s anti-war song suggested by the title of this paper, "RTT (grunt), what is it good for? Absolutely nothing (say it again)...". This may have seemed like an obvious result, but we would like to restate that until now no one as far as we know has published results demonstrating this.

Before we leave the topic however, it may be appropriate to cast one small shadow of doubt over the result: throughout this work we have relied on the BLEU and F-score metrics to judge the translations. So our conclusion is really that RTT cannot tell good MT systems from bad ones, or easy-to-translate texts from hard ones, *based on automatic evaluation methods*. To be really sure of our results, we should like to replicate the experiments evaluating the translations using a more old-fashioned method involving human ratings of intelligibility. The reason for this is that both these metrics reward translations that are lexically close to the oracle translation, without taking into account whether they are grammatical or make sense. If we look at our high-scoring BTs, we can see that often they do indeed match the vocabulary of the model translations, without making much sense: contrast examples (5) and (6), both high scoring, but differing in qualities of grammaticality and intelligibil-

[3] http://rali.iro.umontreal.ca/meteo, as described in Langlais et al. (in press).

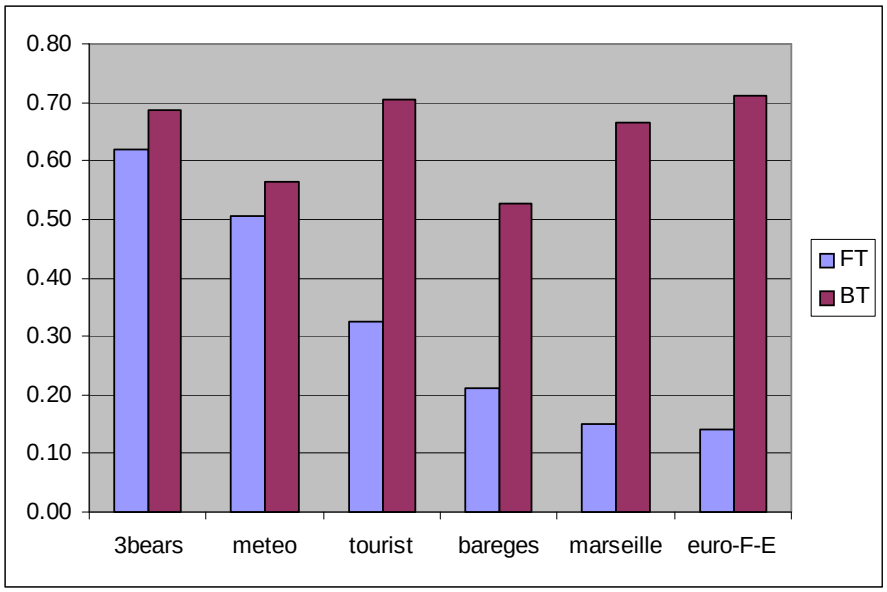


Figure 4. BLEU scores for the forward and back translations of three 'easy' and three 'hard' texts.

ity. So perhaps this is not quite the end of the story after all.

Acknowledgments

This work was completed while the author was on study leave at the Centre for Language Technology, Macquarie University. He is most grateful to all his colleagues there for their warm welcome, and especially Simon Zwaarts and Menno van Zaanen, whose programs and scripts were used to translate on-line large amounts of material, overcoming the text-length limits imposed by many of the systems, and to implement the BLEU and F-score measures. Thanks also to Elena Akhmanova for advice on the Russian examples in this paper, and to Steve Cassidy for suggesting the experiments reported here.

References

Anon. 2003. More Machine Translation: Fun with computer generated translation!, *Biomedical Translations*, News, October 2003. www.biomedical.com/news.html.

Anon. 2005, Gotcha!: Translation software. Software that translates text from one language to another may be a big help—or hindrance—to businesses and relief agencies alike. *Baseline*, May 2, 2005. www.baselinemag.com/article2/0,1397,1791588,00.asp.

Bernth, A. 1999a. EasyEnglish: A confidence index for MT. *Proceedings of the 8th International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '99*, Chester, England, pp. 120–127.

Bernth, A. 1999b. Controlling input and output of MT for greater user acceptance. *Translating and the Computer 21*, London, [pages not numbered].

Bernth, A. and C. Gdaniec. 2002. MTranslatability. *Machine Translation* **16**:175–218.

Bernth, A. and M. McCord. 2000. The effect of source analysis on translation confidence. In J.S. White (ed.) *Envisioning Machine Translation in the Information Future: 4th Conference of the Association for Machine Translation in the Americas, AMTA 2000, Cuernavaca, Mexico, ...*, Berlin: Springer, pp. 89–99.

Flanagan, M. 1996. Two years online: experiences, challenges and trends. *Expanding MT Horizons: Proceedings of the Second Conference of the Association for Machine Translation in the Americas*, Montreal, Canada, 192–197.

Gaspari, F. 2004. Integrating on-line MT services into monolingual web-sites for dissemination purposes: An evaluation perspective. *9th EAMT Workshop Broadening Horizons of Machine Translation and its Applications*, Valletta, Malta, pp. 62–72.

Gdaniec, C. 1994. The Logos translatability index. *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, Columbia, Maryland, pp. 97–105.

Huang, X. 1990. A Machine Translation system for the target language inexpert. *Papers presented to the 13th International Conference on Computational*

Linguistics, COLING-90, Vol. 3, Helsinki, pp. 364–367.

Langlais, P., S. Gandrabur, T. Leplus and G. Lapalme. In press. The long-term forecast for weather bulletin translation. To appear in *Machine Translation.*

Macklovitch, E. 2001. Recent trends in translation technology. *Proceedings of the 2nd International Conference, The Translation Industry Today: Multilingual Documentation, Technology, Market*, Bologna, Italy, pp. 23–47.

O'Brien, S. In press. Methodologies for measuring the correlations between post-editing effort and machine translatability. To appear in *Machine Translation*.

O'Connell, T.A. 2001. Preparing your web site for machine translation: how to avoid losing (or gaining) something in the translation. IBM website, www-128.ibm.com/developerworks/web/library/us-mt/.

Papineni, K., S. Roukos, T. Ward and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. *ACL-02: 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA, pp. 311–318.

Somers, H. and F. Gaspari. 2005. The impact of free web-based Machine Translation services on internationalisation. Paper presented at APWSI 2005: The Asia Pacific Workshop on Software Internationalisation, at ASWEC 2005, The Australasian Software Engineering Conference, Brisbane, Qld.

Turian, J.P., L. Shen, and I.D. Melamed. 2003. Evaluation of machine translation and its evaluation. *MT Summit IX: Proceedings of the Ninth Machine Translation Summit*, New Orleans, LA, pp. 23–28.

Underwood, N. and B. Jongejan. 2001. Translatability checker: A tool to help decide whether to use MT. In *Proceedings of MT Summit VIII: Machine Translation in the Information Age*, Santiago de Compostela, Spain, pp. 363–368.

Way, A. and N. Gough. In press. Controlled translation in an example-based environment: What do automatic evaluation metrics tell us? To appear in *Machine Translation*.

Yang, J. and Lange, E.D. 1998. Systran on AltaVista: a user study on real-time machine translation on the Internet. In: D. Farwell, L. Gerber and E. Hovy (eds) *Machine Translation and the Information Soup*: *Third Conference of the Association for Machine Translation in the Americas, AMTA'98, Langhorne, PA,*Berlin: Springer, 275–285.

Evaluating the Utility of Appraisal Hierarchies as a Method for Sentiment Classification

Jeremy FLETCHER and Jon PATRICK
Sydney Language Technology Research Group
The University of Sydney
Sydney, Australia, 2006
{jeremy, jonpat}@it.usyd.edu.au

Abstract

Recent studies of sentiment classification (determining whether a text is "positive" or "negative") using Appraisal theory have provided mixed results. While some good results have been obtained, it is difficult to tell what aspects of Appraisal are particularly useful for this task. In this paper, we present a series of experiments to isolate features of Appraisal, in order to compare which parts aid the task of sentiment classification on movie reviews. We report results which on the surface challenge the utility of Appraisal Hierarchies for this task, when modelled using systemic features. However in the context of making a trade-off between coverage and scale of feature space, our results appear promising. We hence discuss the need for a balance between the size of a classifier's structure and the overall accuracy.

1. Introduction

Sentiment classification is a field of growing interest in the computational linguistics world, as researchers see the need for what has been termed non-topical text analysis. Sentiment classification deals with the problem of determining whether a document is *positive* or *negative*. This task has wide-ranging applications, notably market research, and customer feedback. This paper sets about to determine the usefulness of the linguistic theory of *Appraisal* for Sentiment Classification.

Appraisal theory describes how opinion is expressed in text. Its description is in the form of system networks denoted by a taxonomy of expressions. In this work we rely on the description of these taxonomies in Martin and White's *The Language of Evaluation: Appraisal in English* (2005), for both our linguistically guided hierarchies and realisations of features[1]. Figure 1 shows a visualisation of the system network we use from appraisal theory.

Intuitively, it would seem that appraisal, if it could be modelled effectively using computational methods, would be a useful tool for sentiment analysis. A linguistic theory which gives us insight into the underlying construction of the opinion of the author of a piece of text should, in theory, allow us to compute such opinion more effectively. Previous work on using Appraisal for Sentiment Analysis, however, has been unconvincing and somewhat inconclusive.

In this paper, we set about trying to isolate the areas of appraisal theory which are useful and applicable to sentiment analysis. Subsequently, we wish to determine where efforts in the automatic extraction of a document's appraisal profile should be focussed.

2. Previous Work

There has been some work done on the use of Appraisal for sentiment analysis, including the work of Taboada and Grieve (2004) in which different categories of product reviews were analysed for different types of Attitude (the three sub-systems being Affect, Judgment and Appreciation), using adjectives which had been assigned particular proportions of each of these systems.

Perhaps the most relevant work though is that of Whitelaw, Argamon and Garg (2005), in which movie reviews (data set from (Pang and Lee, 2004)) are classified over a positive/negative dichotomy, using what they term *appraisal groups*. The frequencies of expressions within a text which bear opinion in the appraisal groups are counted. These counts are normalised against the total counts of appraisal groups within the

[1] In this study, we use only the ATTITUDE system from Martin and White's Appraisal structure, and append a simultaneous ORIENTATION system (cf. Whitelaw, Argamon and Garg, 2005). We omit the ENGAGEMENT and GRADUATION systems as they are not suited to the computational methods used in this study.

Proceedings of the Australasian Language Technology Workshop 2005, pages 134–142,
Sydney, Australia, December 2005.

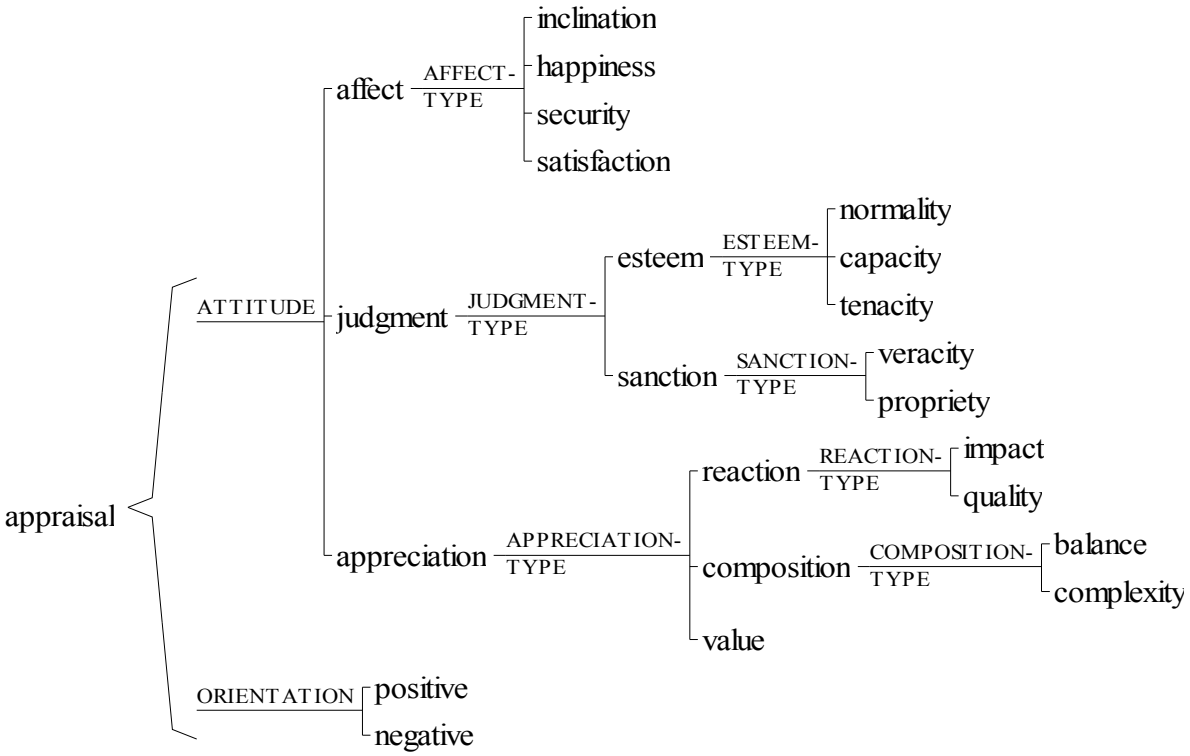


Figure 1: The Appraisal system network, comprising ATTITUDE and ORIENTATION sub-systems.

document.

They note, however, that work on similar taxonomies for other tasks such as the analysis of interpersonal distance (Whitelaw, Herke-Couchman and Patrick, 2004; Whitelaw and Patrick, 2004) and genre classification (Argamon and Dodick, 2004) use relative features within the hierarchies to model the "choice" made by the author about the way a particular structure is expressed.

Whitelaw and Patrick argue convincingly that this modelling of choice as it relates to meaning is an effective realisation of the tenets of the Systemic Functional Linguistic theory (see Halliday, 1994) which is the basis for the Appraisal model. Despite this, Whitelaw et al acknowledge that the use of this type of modelling of Appraisal for sentiment analysis gives inferior results to the simpler, non-theory conformant procedure they adopt.

3. Motivation

The results of Whitelaw et al using Appraisal for Sentiment Analysis were promising but unconvincing. Using their model of Appraisal theory, they were able to beat a baseline of simple bag-of-words analysis, and also improved on the then state of the art (Pang and Lee, 2004).

However, as we have already noted, to do this they removed the notion of modelling choice in the document, and used a simpler model of relative frequencies. This, then, raises questions about whether their Appraisal model does in fact match the true notion of Appraisal in Systemic Functional Linguistics.

Here, we adopt their methodology for populating the lexical realisations in the system network, in order to ascertain the areas of their use of Appraisal which are useful. However, in order to more closely model the linguistic phenomena of Appraisal, we revert to the use of relative systemic features (Whitelaw and Patrick, 2004).

We are hence attempting to approximate a computational method for linguistic analysis of Appraisal, in order to determine how useful such a method is for this task.

We distinguish between three key operations in the computational processing of system networks. Firstly, there is the system network design, in which the structure of the hierarchy is created (at this time, this process is a manual process, and the hierarchies used are those created by linguists). Secondly, there is the realisation of the system network, in which the concepts represented by nodes in the network are mapped to identifiable text features. And finally, there is the instantiation of a particular document as a representation of a system network. This final process involves some kind of abstraction of the text of a document using the realisations from the second process.

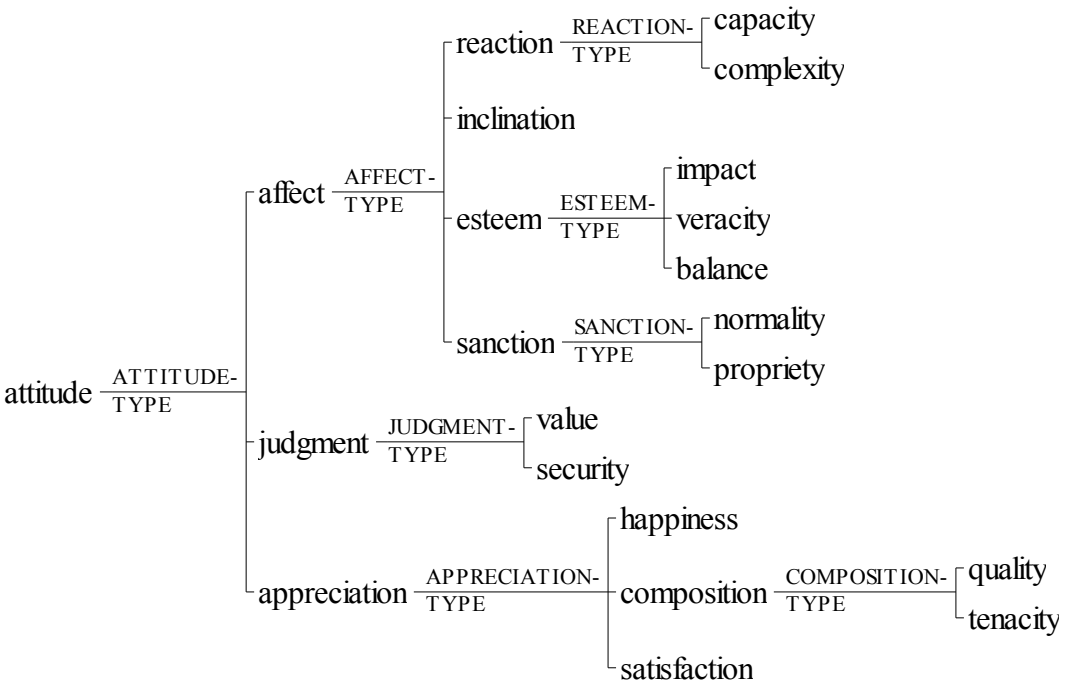


Figure 2: The ATTITUDE system from the *RelationshipsShuffled* system network. Nodes appear at the same depth as they did in the original tree (Figure 1), but their relationships to the next level are modified. (see note)

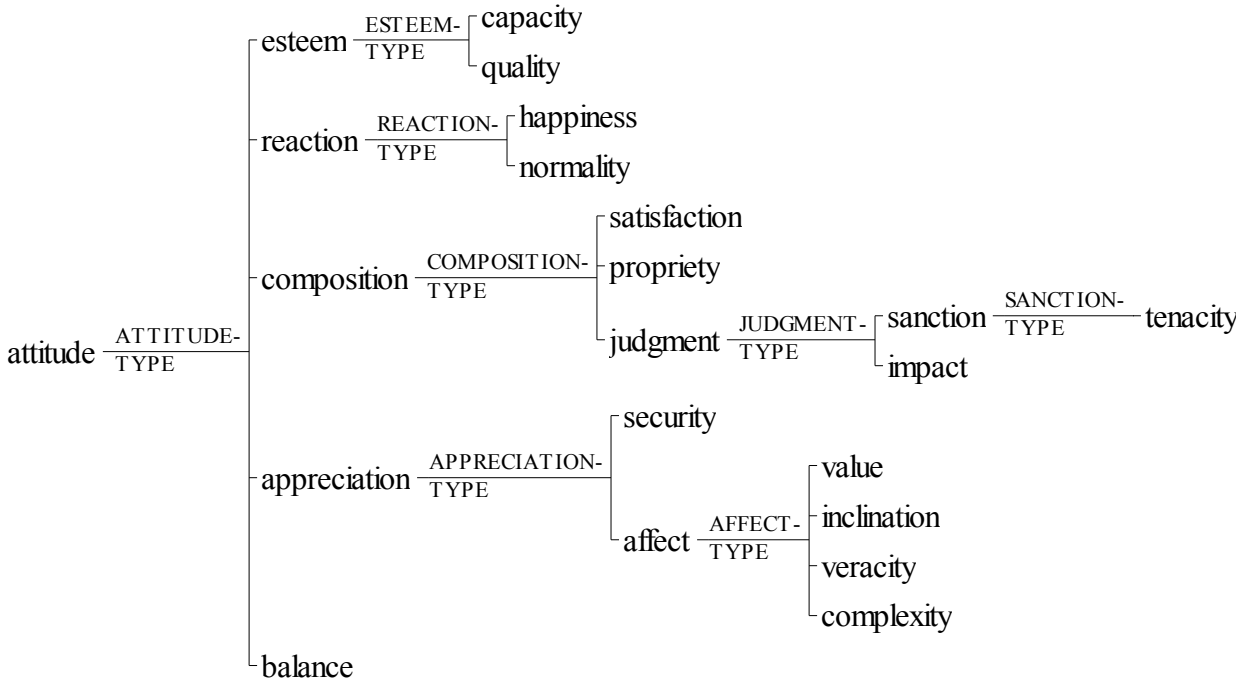


Figure 3: The ATTITUDE system from the *Hierarchy Shuffled* system network. Leaf nodes stay as leaf nodes, but other nodes are randomly assigned to places in the hierarchy. (see note)

NOTE: These systems each occur simultaneously with the ORIENTATION system within APPRAISAL. However, because of the nature of the shuffling, the ORIENTATION system remains the same in both cases.

Experiment	Raw Counts	Systemic Features	Original System Network	Relationship Shuffled (see Figure 2)	Hierarchy Shuffled (see Figure 3)	Attitude Realisations Shuffled	Orientation Realisations Shuffled
Baseline – ABOW	X						
Baseline – System Counts	X		X				
1		X	X				
2		X		X			
3		X			X		
4		X		X		X	X
5		X			X	X	X
6		X		X		X	
7		X			X	X	

Table 1: Feature types and networks used in the experiments

Our hypothesis is that there are particular elements of value in having the Appraisal in a document according to its conformance to the systemic network structure.

4. Realising the Appraisal System Network

In order to compute the appraisal profile of a document, we must be able to relate the content words in the document to the Appraisal system network of the theory.

The most common way of doing this is to attach to appropriate concept nodes in the tree a set of unigram features which are leaf-level "realisations". The system network is then instantiated for each document by counting all the realisation features within a document, and aggregating these counts up the tree.

However, one of the problems of this method is how to create a set of these realisations. For Appraisal, there are some small example texts from the Systemic Functional Linguistics literature, but not enough to allow for reasonable coverage within a computational framework.

To circumvent this lack of coverage of realisation, we took the example text from Martin and White (2005) as seed terms, and using the method of Whitelaw, Argamon and Garg (2005), expand the lexicon by generating synonyms from WordNet and two online thesauri[2]. From this, we also get a measure of the "confidence" of each expanded term, by counting the number of times a particular term is encountered from thesaural expansion in a particular node in the system network. For example, we may encounter "joyous" as a synonym of two different realisations of HAPPINESS ("happy" and "jubilant"), indicating that it is perhaps a stronger indication of that node than something which only occurs as a synonym once.

Note also that a particular unigram realisation may occur at numerous places within the system network. A particular unigram does not necessarily have a unique location within the system network. For example, the adjective "good" may be used in different contexts to realise SATISFACTION, PROPRIETY, QUALITY or VALUE, to name a few. Thus, each instance of "good" in a document increases the counts at each of these positions in the network.

While this method in no way guarantees complete coverage for the corpus, it does increase the coverage significantly, while still assuring Appraisal items can be identified computationally.

5. Experiments

We ran a set of experiments to classify Pang and Lee's (2004) movie review corpus as containing positive or negative sentiment[3].

In order to test our hypothesis, we developed a set of experiments to isolate particular attributes of the structure of the system network. In order to make the results comparable, we performed a process of randomising or shuffling the nodes in the network, thereby eliminating some of the linguistic information contained within the

[2] http://m-w.com and http://thesaurus.reference.com

[3] This dataset is freely available at http://www.cs.cornell.edu/people/pabo/movie-review-data/

particular representation of the tree.

There are two ways in which we randomise the network, and two levels of intensity with which we do it.

The first method of randomising the network involves keeping each node at the same depth as it was in the original, and simply randomly assigning a parent node to each, then continuing this process up the tree. The (shuffled) system network which results from this process is given in Figure 2 (*RelationshipsShuffled Network*).

The second method involves complete random assignment of nodes within the hierarchy. This means that any node can appear at any point within the hierarchy, with only the leaf nodes (which contain the realisations) staying as leaf nodes within the system. This system is shown in Figure 3 (*HierarchyShuffled Network*).

Of course, once this process has been executed, there is no longer a relationship between the labelling of particular nodes in the original and shuffled networks. For example, in Figure 2 AFFECT no longer encompasses HAPPINESS, SECURITY or SATISFACTION, and thus bears little resemblance to its function in the original network.

Running experiments using these shuffled networks and comparing the results to those we get on the original tree gives us some measure of the utility of the arrangement of systems within the original tree. If there is something of particular use to sentiment analysis that can be gleaned from the structure of the original tree, it should be reflected in the results on the different networks.

The second level of intensity involves shuffling the realisations between leaf nodes. Hence, word-level realisations are no longer grouped together as they were discovered in the process of thesaural expansion.

5.1 Methodology

We use the confidence measure we attain from the thesaural expansion to weight each realisation placed in the tree. That is, if a particular unigram has a high confidence measure for a particular node, the count value for a document will be increased more than if the confidence was low. This has two implications: firstly, words which are included as realisations from thesaural expansions of peripheral unigrams (and thus are less likely to be accurate realisations of appraisal) have little impact when found in a document, while increasing the impact of those unigrams which we are confident have some semantic similarity to our hand-crafted seed terms. Secondly, it means that individual unigrams are weighted for each node they realise. That is, a unigram may be a strong indicator of a particular node, but a weak indicator for another (due perhaps to having another, less common sense). This weighting then accounts for this case, rather than assigning the same weight for each realisation of each node.

We acknowledge that this confidence measure is a heuristic, and lacks manual crafting, but it increases the confidence about the decisions which have to be made in a computational process.

Once system network instance counts have been accumulated for a particular document, we calculate proportions of systems to their parents and siblings, using System Percentage (SYSPERC) and System Contribution (SYSCON) (Whitelaw, Herke-Couchman and Patrick, 2004; Whitelaw and Patrick, 2004).

SYSPERC: The proportion of the total system usage made up by this particular sub-system.

SYSCON: The proportion of a super-system's usage made up by a particular sub-system.

These features, once calculated were used as data for WEKA's (Witten and Frank, 1999) implementation of the SMO (Platt, 1998) support vector machine learning algorithm. We used a linear kernel and default parameters. Evaluation was done using 10-fold cross validation.

5.2 Experimental Results

We ran a series of experiments to evaluate the accuracy of classification of movie reviews created using the features of the linguistically modelled system network, and the same features throughoutt the shuffled system networks.

The results from the shuffled system networks were compared to the results on the hand-crafted hierarchies, as well as two baselines. Our experiments (summarised in Table 1) are as follows:

Baseline 1: *Appraisal-Bag-of-Words* (ABOW) – relative frequencies of all words which appear as realisations of systems in the Appraisal system network, normalised by document length. Omitted from the experiment are the realisations which do not appear in any document in the corpus, leaving 4,381 features.

Baseline 2: *Bag of Nodes* – the relative frequencies of the raw counts of each node in the system hierarchy, normalised by the total number of appraisal counts in the document (i.e. the aggregated count at the root of the hierarchy)

Experiment 1: SYSPERC and SYSCON measures at all levels in the hierarchy, using the original linguistically created system network.

Experiment 2: SYSPERC and SYSCON measures at all levels in the hierarchy, using the *RelationshipsShuffled* system network.

Experiment 3: SYSPERC and SYSCON measures at all levels in the hierarchy, using the *HierarchyShuffled* system network.

Experiment 4: SYSPERC and SYSCON measures at all levels in the hierarchy, using the *RelationshipsShuffled* system network, and realisations randomly assigned in both the ATTITUDE and ORIENTATION networks.

Experiment 5: SYSPERC and SYSCON measures at all levels in the hierarchy, using the *HierarchyShuffled* system network, and realisations randomly assigned in both the ATTITUDE and ORIENTATION networks.

Experiment 6: SYSPERC and SYSCON measures at all levels in the hierarchy, using the *RelationshipsShuffled* system network, and realisations randomly assigned in just the ATTITUDE network.

Experiment 7: SYSPERC and SYSCON measures at all levels in the hierarchy, using the *HierarchyShuffled* system network, and realisations randomly assigned in just the ATTITUDE network.

The results of these experiments are shown in Table 2.

Experiment	Acc. (%)
Baseline (ABOW)	83.7
Baseline (System counts)	71.8
1	72.4
2	72.6
3	72.8
4	67.8
5	69.5
6	68.5
7	70.4

Table 2: 10-fold cross validation results for different system networks and feature set configurations. (See Table 1 for details of experiments)

6. Analysis of Results

Immediately apparent from these results is the degradation of accuracy when you move from the Appraisal-bag-of-words features to systemic features. This mimics the results of Whitelaw et al who report that the use of these systemic features produces inferior results to their simpler measures. The most likely cause for this discrepancy is the fact that the Appraisal tree is reasonably shallow, so the aggregative properties of these features do not have the scope of previous experiments on these networks.

Occam's Razor tells us to "not multiply entities without necessity" and BOW classifiers rampantly ignore this economy argument. Our real objective should be to produce the classifier that attains the highest accuracy with the least model complexity, and to this end we need to devise new metrics of performance that balance performance against classifier size. In this light, the "efficiency" of the Appraisal model can be seen as superior to BOW. Of course, the complexity of a classification system relies on more than feature set size.

Our objective with this set of experiments, however, is to draw comparisons between the results of the linguistically created network and our shuffled hierarchies.

What we note about these results, is that there is very little difference between whether the hierarchy used is the linguistically created network, or one of those which was randomised to some degree. We can see that the accuracy on our original tree is 72.4%, whereas the results of the same feature set using our *RelationshipsShuffled* and *HierarchyShuffled* trees were 72.6% and 72.8% respectively. This leads us to believe that there is no advantage for sentiment analysis in the use of the structure of the original Appraisal network when modelled computationally in the manner we have described[4].

However, what we do notice is the distinct drop in accuracy once the realisations are randomly assigned to the leaf nodes in our hierarchy. Our accuracies drop by approximately 5% once this shuffling of realisations has taken place.

Given that the results above show no benefit in the structure of the hierarchy, we can deduce that the benefit comes from having our unigram realisations grouped together in some semantic categories.

Experiments 6 and 7 attempted to isolate the shuffling of realisations within the ORIENTATION network, as we felt that this decrease in accuracy may be due simply to the fact that each realisation (in experiments 1-5) had been assigned either a "positive" or "negative" Orientation value. This type of processing of Semantic Orientation (SO) has been exploited for sentiment classification previously (Hatzivassiloglou and McKeown, 1997; Turney, 2002).

[4] The small *increases* in accuracy over experiments 2 and 3 are most probably not statistically significant.

However, leaving the ORIENTATION realisations unshuffled produces only a minor increase in accuracy (Exp 4/5 v Exp 6/7), and the results of these experiments are still well below the results on those where the ATTITUDE realisations are also unshuffled (Exp 2/3 v Exp 6/7). This indicates that it is not only the Semantic Orientation of our realisations which aid classification, but also the categories of ATTITUDE.

Despite this, when we compare the results achieved on this type of analysis to the simple Appraisal-bag-of-words classification, there is a very marked decrease in accuracy.

Most probably this is due to the additional granularity which can be achieved by looking at words on an individual level. What is important to note is that although in the ABOW experiment there is no preordained measure of sentiment attached to the words, the machine learner distinguishes words which have an intrinsic positive or negative connotation some of which are "bad", "mess", "waste", "worst", "stupid", along with "fun", "great", "terrific", "memorable" and "hilarious". Perhaps more interesting is that some word features which do not intrinsically contain a semantic orientation become strong word features in the ABOW experiments. Words such as "very", "also", "nowadays", "many" and "leave" are indicators of positive sentiment, and words such as "only", "have", "work", "plot" and "intended" seem to indicate negative sentiment.

This indicates that there is perhaps some value to analysing the structure of the text, and how rhetorical structure is realised differently in positive and negative reviews. Another reason for these strong word features is perhaps their collocation with other sentiment-bearing expressions. In this case, a process for identifying frequent collocations in the text may also be a useful tool for identifying better sentiment-bearing expressions, as well as increasing the number of realisations. This acknowledges the need for more complex realisations of the system network.

The peculiarities of particular words being indicators of a particular orientation of sentiment are worth exploring. For example, the fact that "plot" tends to be indicative of negative sentiment suggests that those movie reviews which make specific reference to the plot are more likely to be negative. This raises questions about different styles of reviewing; are there ways to extract information about how hard or leniently a reviewer gives his or her opinion? When dealing with a style of text which is opinion heavy, especially when resolving the opinion into a positive/negative dichotomy, issues of review style come into effect.

In fact, one of the largest problems with the use of these hierarchies, and perhaps the reason why accuracy using them is less than with ABOW features, is simply the lack of coverage. While we have isolated potential sites for sentiment within a text by collecting and expanding lists of Appraisal realisations, it is reasonable to expect that there are many more which are not captured. Moreover, those Appraisal expressions which we do have form only small proportion of the text as a whole[5].

Experiment	**# *features***	**Acc. (%)**
W:A[6]	1047	77.6%
ABOW	4318	83.7%
BOW[6]	48,314	87.0%

Table 3: Comparison of percentage accuracy and size of feature set.

Although the Appraisal results generally and hierarchical use in particular do not appear to be competitive this is not the whole story. An investigation of the size of the classifying indicates a strong efficiency in the Appraisal classifier. Using full bag-of-words features on the same set of documents, Whitelaw et al attain accuracy of 87.0%. However, to attain this accuracy, they used 48314 features, whereas our results on the Appraisal-bag-of-words (83.7%) are attained through the use of only 4381 features, less than ten percent of the size (see Table 3). To draw direct comparisons between the accuracies of these feature sets, however, it would be necessary to compare the different types of selected features over similar set sizes. How effective is the use of the 50 best appraisal features compared to the 50 best unigram BOW features? These questions are open for discussion.

Whitelaw et al's result on a similar feature set to our Appraisal-bag-of-words, using 1047 features is 77.6% accuracy. This continuum relationship between size of feature set and accuracy once again emphasises the need for some balance in a real working system. One of the problems which must be addressed within any type of text classification system is our ability to reach the accuracy of feature indiscriminate classifiers using classifiers with a much smaller internal structure.

[5] Even with the thesaural expansion, the Appraisal realisations only make up ~22% of the unique words within the text.

[6] These experiments are documented in Whitelaw, Argamon and Garg (2005)

7. Discussion and Future Work

We have presented here a set of experiments designed to test the utility of particular parts of Appraisal theory for sentiment analysis. While we are still far from a definitive answer as to whether this type of processing is useful in this domain, the results here show that there is little benefit to be gained from structure of the Appraisal network. Perhaps one reason for this is that the network itself is quite shallow. The deepest node in the hierarchy is only four links from the root, indicating that the type of aggregative statistics gained from System Contribution and System Percentage are ill-suited to this particular network.

Furthermore, it is also possible that we are not yet able to effectively approximate a model of Appraisal theory using computational methods. Linguists in particular would argue that our Appraisal-as-realisations methodology does not do justice to the complexities of the theory.

As we discussed, there is some merit in the level of distinction gained by the machine learner on the Appraisal-bag-of-words features. In actuality, in the process, the learner discovers patterns between the word-level realisations and the sentiment of the document as a whole.

A potentially useful extrapolation of this principle is having a machine learner craft an "optimal" hierarchy for Appraisal, for a particular task. While the Appraisal hierarchy we see in the linguistics literature is useful for general descriptions of linguistic phenomena, it is probably true that modifications to suit a particular task could amplify the delineation of some aspect of the text (for example sentiment analysis), thereby increasing the accuracy of computational processing.

Another method of testing the applicability of this theory to the computational process of sentiment analysis is to use the Appraisal network for a different type of text classification (for example, topic classification). If, in fact, there is a notable decrease in the accuracy of the Appraisal model on non-emotive texts, then we can see that there is a particular relationship between Appraisal theory and the computational process of sentiment analysis. However, if the only real utility provided by the network is some kind of smoothing process, or a benefit from the aggregative properties of the hierarchy, which would perhaps be attested by similar results on emotive and non-emotive texts, then we can no doubt create (or construct using automatic methods) better hierarchies for doing this.

Overall, there is still some question over whether Appraisal theory is useful for the computational process of sentiment classification. The results here suggest that there is some value to be gained from the grouping of words into Appraisal clusters, but it is also true that only doing this decreases the accuracy over using the words themselves.

However, before a definitive answer is given to this question, we would need to assess our computational model of Appraisal in mode detail. There is obviously some level of uncertainty in the model, due to the method of realising nodes in the system network. It is also likely that the instantiations of the networks need to be modelled in richer ways.

There is ongoing research into the way realisations of linguistic phenomena are modelled computationally, and is important that other methods of such realisation are explored before the use of Appraisal for sentiment classification is discarded.

On the criteria of efficiency, however, the Appraisal model appears to work very well, although a true comparison has not been achieved in this paper. That would require comparison to a result from the top 4318 features in a BOW experiment excluding the Appraisal terms.

References

S. Argamon and J. T. Dodick. 2004. "Conjunction and modal assessment in genre classification". In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text*. AAAI.

Michael A. K. Halliday. 1994. *Introduction to Functional Grammar*. Edward Arnold.

V. Hatzivassiloglou and K. R. McKeown. 1997. *Predicting the semantic orientation of adjectives*. In "Proceedings of the 35th ACL and 8th EACL", pages 174-181, Somerset, New Jersey. ACL.

J. R. Martin and P. R. R. White. 2005. *The Language of Evaluation: Appraisal in English*. Palgrave, London.

Bo Pang and Lillian Lee. 2004. "A Sentimental education: Sentiment analysis using subjectivity summarization base on minimum cuts". In *Proc. 42nd ACL*,. Pages 271-278. Barcelona, Spain.

M. Taboada and J. Grieve. 2004. "Analysing appraisal automatically". In *AAAI Sprint Symposium on Exploring Attitude and Affect in Text*. AAAI.

Peter D. Turney. 2002. "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews". In *Proc. 40th ACL,* pages 417-424, Philadelphia, Pennsylvania.

Casey Whitelaw, Shlomo Argamon and Navendu Garg. 2005. Using Appraisal Taxonomies for Sentiment Analysis. In *Proceedings of the First Computational Systemic Functional Grammar Conference*, University of Sydney, Sydney, Australia.

Casey Whitelaw, Maria Herke-Couchman and Jon Patrick. 2004. "Identifying Interpersonal distance using systemic features". In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text.*

Casey Whitelaw and Jon Patrick. 2004. "Selecting Systemic Features for Text Classification". In *Proc. Australasian Language Technology Workshop, 2004*. Macquarie University, Sydney Australia.

Janyce M. Wiebe, Theresa Wilson, Rebecca F. Bruce, Matthew Bell and Melanie Martin. 2004. "Learning Subjective Language". In *Computational Linguistics 30(3),* pages 277-308. MIT Press.

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

Efficient Grapheme-phoneme Alignment for Japanese

Lars Yencken and **Timothy Baldwin**

Dept. of Computer Science and Software Engineering
University of Melbourne
Victoria 3010 Australia

NICTA Victoria Research Lab
University of Melbourne
Victoria 3010 Australia

{lars,tim}@csse.unimelb.edu.au

Abstract

Current approaches to the grapheme-phoneme alignment problem for Japanese achieve good accuracy, but are extremely computationally expensive. In this paper we evaluate various modifications to previous algorithms for both the alignment and okurigana detection subtasks. The best algorithm achieved accuracy of 96.2% for the combined task on a limited data set, and was significantly more efficient than previous approaches.

1 Introduction

Alignment is the task of, for two streams of data which represent alternate construals of the same basic information content, identifying corresponding segments within the two streams. A common alignment task in computational linguistics is word alignment, whereby given an English sentence and its French translation, say, each English word n-gram is aligned with its French translation (Brown et al., 1993; Manning and Schütze, 2000). The combined set of such alignments derived from a parallel corpus is generally used to train the translation model in statistical machine translation systems. Other alignment tasks in computational linguistics include sentence alignment, structural alignment (e.g. as a means of grammar inference), and grapheme-phoneme alignment.

The grapheme-phoneme ("GP") alignment task aims to *maximally* segment the orthographic form of an utterance into morphophonemic units, and align these units to a phonetic transcription of the utterance. Maximal indicates the desire to segment grapheme strings into the smallest meaningful units possible. Taking the English example word *battleship* and its phonetic transcription /bætlʃɪp/, one possible alignment is:

b	a	tt	le	sh	i	p
b	æ	t	l	ʃ	ɪ	p

Note that alignment in general is many-to-many. In the example above, *tt* aligns to /t/, *le* aligns to /l/ and *sh* aligns to /ʃ/. Equally it might be possible for some letters to align to an empty string. This task is challenging for any language without a one-to-one correspondence between individual graphemes and phonemes, as is the case with English (Zhang et al., 1999), Japanese (considering graphemes as kanji characters), and indeed most languages with a pre-existing writing system.

GP alignment is a prerequisite for many applications. For example, the alignment process, and the resulting aligned GP tuples, are a precursor to achieving automated grapheme-to-phoneme mappings for several text-to-speech systems (Allen et al., 1987; Sejnowski and Rosenberg, 1987; Sloat, 1996; Black et al., 1998). Further uses include accented lexicon compression (Pagel et al., 1998), identification of cognates (Kondrak, 2003), Japanese-English back-transliteration (Knight and Graehl, 1998; Bilac and Tanaka, 2005a, 2005b) and finally the FOKS dictionary system for Japanese learners (Bilac, 2002; Bilac et al., 2002), which provides the context for our work.

There are several successful approaches to Japanese GP alignment, notably the iterative rule-based approach taken by Bilac et al. (1999), later followed by an unsupervised statistical model based on TF-IDF by Baldwin and Tanaka (1999a, 1999b). Although these models were found to have high accuracy, their iterative approach had a high computational cost, making them impractical for many real-world applications. For the statistical models, this is partially a consequence of their strongly unsupervised nature. We thus explore the use of the Edict and Kanjidic electronic dictionaries (Breen, 1995) as means of constraining the alignment search space and reducing computational complexity.

The goal of this paper is to compare sev-

Proceedings of the Australasian Language Technology Workshop 2005, pages 143–151,
Sydney, Australia, December 2005.

eral different GP alignment methods in order to achieve equivalent or better alignment accuracy to that for existing methods, at a much lower computational cost. To achieve this goal, we split the task of GP alignment into a pure alignment subtask and an okurigana detection subtask, and compare algorithm variants of pre-existing approaches for both. As our base model, we use the top performing statistical model from Baldwin and Tanaka (2000).

The remainder of this paper is structured as follows. First, we discuss the FOKS system, an important motivator for this work (Section 2). We then discuss the GP-alignment problem for Japanese in greater depth (Section 3), before giving details of the baseline statistical model and our modifications to it (Section 4). Finally, we discuss our results on a manually aligned test data set (Section 5).

2 FOKS dictionary system

GP alignment is an important step in the pipeline that drives the FOKS ("Forgiving Online Kanji Search") dictionary interface (Bilac, 2002), our particular research interest. Whereas normal electronic dictionaries will not provide the target word if an incorrect reading is looked up, FOKS is able to compensate for learner mistakes by dynamically predicting readings for compounds, and aims to direct the user to the correct word despite possible mistakes in the entered reading.

For example, suppose the user wishes to look up 風邪 [*ka-ze*] "common cold". He or she may know the kanji 風 [*ka-ze/fu-u*] "wind", and also 邪 [*yo-ko-shi-ma/jya*] "evil, wicked", and thus guess that the reading for 風邪 is *ka-ze-yo-ko-shi-ma*, one possible combination of readings. However, the correct reading *ka-ze* is non-compositional. Despite the incorrect guess, FOKS still lists the target word with the correct reading in its list of candidates for the guessed reading.

The back-end data that drives FOKS is constructed as follows. Firstly, all entries in the Edict dictionary are GP aligned. The subsequent GP tuples are counted to estimate the probability $P(r|k)$ of a given reading r for a given grapheme segment k. Composing segment probabilities together gives the probability $P(r|s)$ of an entire dictionary entry s taking reading r. Using Bayes rule, we finally calculate $P(s|r)$, the probability of a dictionary entry s being the target entry given the user provided reading r. The entries s with non-zero probabilities form our list of candidates for the user's query, and the probabilities $P(s|r)$ provide the basis of the ranking (Bilac et al., 2002). GP alignment allows us to calculate each $P(r|k)$ and is thus a pivotal supporting technology which underlies the FOKS system.

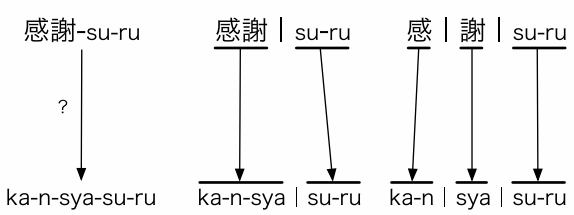


Figure 1: A typical dictionary entry requiring GP alignment, with two potential alignments shown

3 Grapheme-phoneme alignment in Japanese

In the context of Japanese, the GP-alignment task has a few peculiarities. Japanese has three scripts: *kanji*, *hiragana*, and *katakana*. Since hiragana and katakana (or *kana* collectively) are essentially phonetic, we can represent the phoneme string using either of these scripts directly. Kanji on the other hand are ideographic rather than phonetic. Each kanji may have many readings as a single unit, and may also form part of larger units which themselves take on one or more readings. To emphasize the difference between scripts, we shall use romanizations for the phonetic scripts. Figure 1 gives an example for 感謝する [*ka-n-sya-su-ru*] "to give thanks, be thankful".

Given that kana are phonetic, the main task is then reduced to determining how the kanji elements should be segmented, and what elements of the phoneme string they correspond to. Below, we outline four features of Japanese that impede this task.

3.1 Okurigana alternation

Individual kanji segments do not always correspond to minimal units in language. Often a hiragana suffix of some description (usually conjugational) is required, which we term *okurigana*. Verb and adjective conjugation fall under this category: for example 行-*ku* [*i-ku*] "go" in plain form changes to 行-*ke* [*i-ke*] in the imperative. Any useful segmentation should thus

include such suffixes along with their kanji stem in order to preserve the basic morpho-phonemic structure of the compound.

Although most cases of okurigana represent verb and adjective conjugation, there are many general cases such as that of the kanji 取, which occurs in compounds almost exclusively as 取-*ri* [*to-ri*], but also has an alternate where the suffix *ri* is conflated with the kanji stem (such as in 取分 [*to-ri-bu-n*] "one's share or portion"). With some lexemes, both alternants are possible, such as in 取-*ri*-分 [*to-ri-bu-n*]. It is desirable for systems to be able to capture such alternations, in order to achieve consistent segmentation behaviour and attain an accurate estimate of the frequency with which a given kanji occurs with a particular reading (independent of the exact lexical form of the word).

3.2 Sequential voicing

Sequential voicing occurs when a tailing segment has its initial consonant voiced. For example: 本 [*ho-n*] "book" + 棚 [*ta-na*] "shelf" → 本棚 [*ho-n-da-na*] "bookshelf". Although sequential voicing is notoriously unpredictable, its potential occurrence is constrained by Lyman's law, which states that sequential voicing will not occur where there are existing voiced obstruents in the tailing segment (Vance, 1987). It occurs in about 75% of cases where Lyman's law is not violated, with some systematic irregularities for noun-noun compounds as found in recent work by Rosen (2003).

Alignment methods based on precedence or frequency counts may be hindered by sequential voicing, since aligned grapheme/phoneme pairs may not be recognised as phonological variants of previously seen kanji–reading pairs. Fortunately, devoicing is relatively simple, so a common approach is to simply consider voiced and devoiced grapheme/phoneme pairs to be equivalent for counting or comparison.

3.3 Sound euphony

Sound euphony occurs when the last syllable of a leading segment is modified to match the sound of the tailing segment. This is marked uniquely by the っ kana character in Japanese. For example: 国 [*ko-ku*] "country" + 境 [*kyo-o*] "boundary" → 国境 [*ko-k-kyo-o*] "national border". Unlike sequential voicing, which imposes a reversible transformation, it is not clear from 国境 [*ko-k-kyo-o*] "national border" what the original kana ending for 国 was (possibilities include *ko-ki*, *ko-ku*, *ko-su* and *ko-tsu*).

3.4 Grapheme gapping

Occasionally a kana is omitted from the written form of a word, but does not constitute a component of the readings of the neighbouring kanji. Typically the kana can also be explicitly included in the written form of the word. For example: 山 [*ya-ma*] "mountain" + *no* [*GENITIVE*] + 手 [*te*] "hand" can be written as either 山手 or 山-*no*-手, both with reading [*ya-ma-no-te*].

Grapheme gapping is very rare, normally only occurs with the particles *ga* or *no*, and tends not to be productive, suggesting that even approaches aimed at open text are better off simply storing each case individually. The only productive case involving a kanji is 真 [*ma*]. For example: 真 [*ma*] "true/pure" + 暗闇 [*ku-ra-ya-mi*] "darkness" → 真暗闇 [*ma-ku-ra-ya-mi*] "pitch dark", or 真っ暗闇 [*ma-k-ku-ra-ya-mi*] for emphasis.

4 Multi-step alignment

In this section, we first describe the baseline algorithm of Baldwin and Tanaka (1999a, 1999b), before introducing the modifications we propose in this research.

4.1 Overview

A high-level depiction of the unsupervised alignment method of Baldwin and Tanaka (1999a, 1999b) is given in Figure 2. Firstly, all potential segmentations and alignments for input entries are created. Each entry will have potential segmentations and alignments per segmentation which number exponentially in the entry's orthographic length.

Some simple linguistic constraints used as forward constraints to reduce this number are strictly linear alignment, a minimum of one phoneme aligned to each grapheme, and a restriction that each alignment must successfully match any kana entry in the grapheme string with its equivalent phoneme entry. Further constraints used to prune entries include matching okurigana to pre-clustered variants and forcing script-boundaries (except kanji to hiragana boundaries) to correspond to segment boundaries.

Based on the linguistic constraints, we can reasonably expect to have uniquely determined some number of alignments for any sufficiently diverse data set.[1] The uniquely determined

[1] Notable exceptions to this are dictionaries of 4-kanji proverbs, such as the 4JWORDS electronic dictionary,

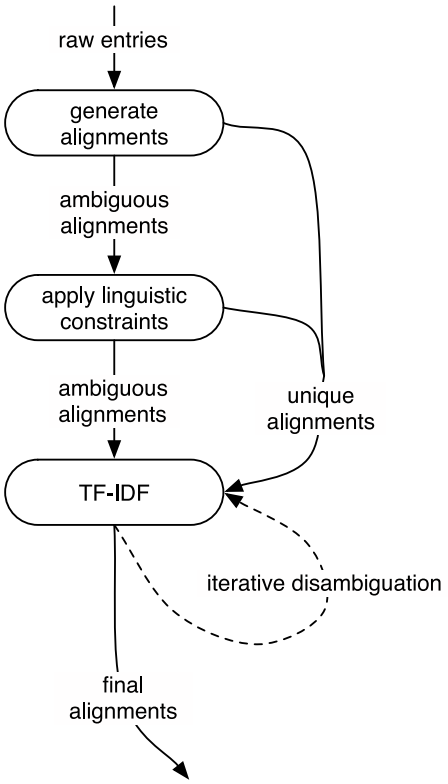


Figure 2: The TF-IDF based alignment algorithm

alignments and the remaining ambiguous alignments are both used separately to seed frequency counts for the TF-IDF model.

TF-IDF is a family of models originally developed for IR tasks, combining the TF (term frequency) and IDF (inverse document frequency) heuristics (Baeza-Yates and Ribiero-Neto, 1999). In the GP alignment task, they mediate the tension between oversegmenting and undersegmenting. The TF value is largest for the most frequently occurring GP pair given any grapheme; an oversegmented alignment produces rarer segments with lower frequency, penalizing the TF score. The IDF value on the other hand is largest for segments which occur in a wide variety of contexts, and penalises undersegmenting.

4.2 TF-IDF Alignment

We use a modified version of the TF-IDF model which takes into account the differing level of confidence we have in our frequency counts between solved ($freq_s$) and ambiguous ($freq_u$) alignments (Baldwin and Tanaka, 2000). For each alignment, we count the occurrence of each grapheme segment $\langle g \rangle$, of each aligned grapheme/phoneme segment pair $\langle g, p \rangle$, and of the same pair with one additional pair of context on either side $\langle g, p, ctxt \rangle$. For any frequency lookup, the w_s and w_u constants provide a weighting between information from solved and ambiguous alignments:

$$wtf(x) = w_s \times freq_s(x) + w_u \times freq_u(x) \quad (1)$$

To score a potential alignment, we calculate the *tf* and *idf* scores for each grapheme/phoneme segment pair and multiply them together as in Equations 2-4. The score for the whole alignment is the average of the scores for every pair which contains a kanji character, since these are the non-trivial pairs. The constant α is intended as a smoothing factor for the TF and IDF scores. It must be assigned such that $0 < \alpha < w_u \leq w_s$.

$$tf(g,p) = \frac{wtf(\langle g,p \rangle) - w_u + \alpha}{wtf(\langle g \rangle)} \quad (2)$$

$$idf(g,p,ctxt) = \log \frac{wtf(\langle g,p \rangle)}{wtf(\langle g,p,ctxt \rangle) - w_u + \alpha} \quad (3)$$

$$score(g,p,ctxt) = tf(g,p) \times idf(g,p,ctxt) \quad (4)$$

Once all potential alignments have been scored, the highest-scoring alignment is chosen to disambiguate its entry. Its counts are removed from the unsolved pool and added to the solved pool, and algorithm reiterates with updated counts. In this way entries are iteratively disambiguated until no more remain, and the algorithm is complete.

Although effective, the iterative algorithm is extremely expensive, with two main costs. Firstly, as with any alignment task where two strings of length l and m respectively need to be aligned, there are 2^{lm} possible alignments before applying constraints (Brown et al., 1993). In our task, kanji essentially form free variables in the alignment, whereas kana align to themselves, constraining the search space. Entries with many kanji and no kana to constrain them thus have prohibitively large numbers of possible alignments. These cases bloat the number of potential alignments to be rescored on each iteration so much that including them makes our main algorithm infeasibly expensive.

whose entries' grapheme forms lack kana to help eliminate possible alignments.

The second bottleneck is in the average case. Suppose there are n alignments pairs, each with p possible alignments. Then the cost of the iterative rescoring loop is $O((np)^2)$. Even having removed the problem cases above, if p is still high on average, the problem will prove intractable for suitably large n. As a comparison, the evaluation set we use has 5000 elements, yet the Edict dictionary has over 110,000 entries, representing a near 500 fold expected increase in computation time. Although this could be mitigated by simply breaking the input down into smaller subsets for processing, it is desirable to process all the data in the same iterative loop, since this gives greatest consistency of alignment.

Strategies to reduce the average case for p and to eliminate the worst case for p thus form the basis for our attempts at modifying the algorithm.

4.3 Modified algorithm

The modified algorithm diverges from the unsupervised algorithm in three main respects. Firstly, we separate out okurigana handling into a separate step after alignment, benefiting both efficiency and error measurement. Secondly, a reading model is introduced based on the Kanjidic electronic dictionary[2] and is used to disambiguate the majority of remaining cases before the TF-IDF model is reached. Thirdly, we provide a maximum alignment size cutoff above which we use a simplified non-iterative alignment algorithm which meets resource constraints for problem cases. We discuss these changes below.

4.3.1 Separating okurigana handling

The okurigana handling in the original algorithm involves pre-clustering okurigana alternates, and attempting to restrict alignments to match these alternates wherever possible. Whilst this constraint does help reduce potential alignments, it also limits the application of the stronger constraint that script boundaries in the grapheme string must correspond to segment boundaries (i.e. every occurrence of a kanji–hiragana script boundary must be considered as a potential okurigana site). If okurigana detection is left as a post-processing task, we can strengthen this constraint to include all script boundaries, instead of omitting kanji-to-hiragana boundaries. This in turn provides a larger gain than the original okurigana constraint, since more entries are fully disambiguated.

The GP-alignment task is then split into two parts: a pure alignment task, which can be carried out as per the original algorithm, and a separate okurigana detection task. This redesign also allows us to separately evaluate the error introduced during alignment, and that introduced during okurigana detection, and thus allows us to experiment more freely with possible models.

4.3.2 Short and long entries

Ultimately, any method which considers all possible alignments for a long entry will not scale well, since potential alignments increase exponentially with input length. We can however extend the applicability of the algorithms considered by simply disambiguating long entries in a non-iterative manner.

The number of potential alignments for an entry can be estimated directly from the number of consecutive kanji. Our approach is to simply to count the number of consecutive kanji in the grapheme string. If this number is above a given threshold, we delay alignment until all the short entries have been aligned. We then use the richer statistical model to align all the long entries in a single pass, without holding their potential alignments in memory.

Although long entries were not an issue in our evaluation set, a threshold set experimentally to 5 consecutive kanji worked well using the Edict dictionary as input, where such entries can prove difficult.

4.3.3 Reading model

For the pure alignment task, we added an additional reading model which disambiguates entries by eliminating alignments whose single kanji readings do not correspond to those in the Kanjidic and KANJD212 electronic dictionaries. These dictionaries list common readings for all kanji in the JIS X 0208-1990 and JIS X 0212-1990 standards respectively, covering 12154 kanji in total. Effectively, we are applying the closed world assumption and allowing only those alignment candidates for which each grapheme unit is associated with a known reading. Only in the instance of over-constraint, i.e. every GP alignment containing at least one unattested reading for a grapheme unit, do we relax this constraint over the overall alignment candidate space for the given grapheme string.

[2] `http://www.csse.monash.edu.au/~jwb/kanjidic.html`

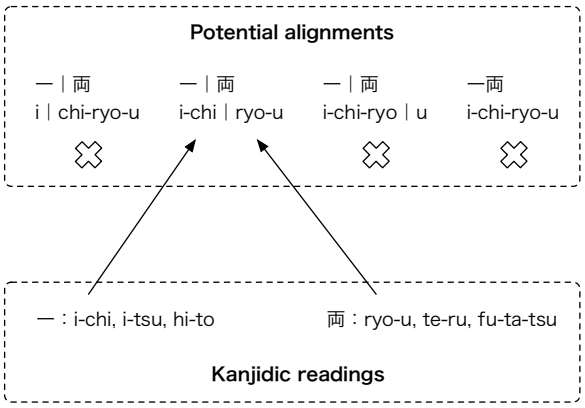


Figure 3: Disambiguation using the reading model

A simple example of disambiguation using the reading model is that of 一両 [*i-chi-ryo-u*] "one vehicle" as shown in Figure 3. Since only one of the potential alignments is compatible with the known readings, we then select it as the correct alignment. As an indication of the effectiveness of the reading model, our initial constraints uniquely determine 31.1% of the entries in the Edict dictionary.[3] The reading model disambiguates a further 60.6% of entries, effectively decreasing the input to the iterative alignment algorithm by an order of magnitude, to the remaining 8.3%.

4.3.4 Heuristic variants

We could continue to use the original TF-IDF model over the residue which is not disambiguated by the reading model, although the type of input has changed considerably after passing through the reading model. Since the reading model is likely to fully disambiguate any entry containing only single kanji segments, the only remaining ambiguous models are likely to be those with solutions containing multi-kanji segments (which do not occur in either Kanjidic or KANJD212); an instance of a multi-kanji segment is our earlier example 風邪 [*ka-ze*] "common cold". With this in mind, we compare the original TF-IDF model (our baseline) with similar models using TF only, IDF only, or random selection to choose which entry/alignment to disambiguate next.

[3] `http://www.csse.monash.edu.au/~jwb/edict.html`

4.3.5 Okurigana detection

We similarly wish to determine what form of okurigana detection and realignment model is most appropriate. Since the majority of entries in the Edict dictionary (our main experimental data set) which contain potential okurigana sites (i.e. kanji followed by hiragana) do contain okurigana in some form, we use as our baseline the simple assumption that every such site is an instance of okurigana. In this manner, the baseline simply removes every kanji-to-kana segment boundary. As a small enhancement, the boundary is not removed if the tailing kana segment is one of the hiragana particles *no*, *ga* or *ni*, which frequently occur alone.

We consider three alternative okurigana models to compare to our baseline, of increasing complexity and expected coverage. Firstly, the Kanjidic dictionary contains common okurigana suffixes for some kanji with conjugating entries. Thus our first model uses these suffixes verbatim for okurigana detection. The coverage of okurigana suffixes in Kanjidic is somewhat patchy, so in our second model, in addition to Kanjidic suffixes, we also perform a frequency count over all potential okurigana sites in the Edict dictionary, and include any occurrences above a set threshold as okurigana.

Finally, most instances of okurigana are due to verb conjugation. As well as taking straight suffixes from the previous models, this final model harvests verbs from Edict. Most verb entries in Edict have a tag marking them as *ichidan*, *godan* or *suru* verbs.[4] The verb type and stem allow us to conjugate regular verbs variously, giving us a large number of new okurigana suffixes not present in the previous models. In order to improve accuracy, all three methods fall back to the baseline method if they do not detect any okurigana.

5 Evaluation

Having teased apart the alignment and okurigana detection algorithms, we are in a position to separately evaluate their performance. Our test set for the combined task consists of 5000 randomly chosen and manually aligned examples from Edict, from which we then separated out an individual evaluation set for each subtask.

Since we are also interested in efficiency, we

[4] The tagset for Edict verbs is larger than this, but the additional tags largely mark subclasses and exceptions of the three main classes, which we ignore for the sake of simplicity.

provide execution time as measured by elapsed time on a standard Pentium 4 desktop PC. Our emphasis however is on the *relative* time taken by different algorithms rather than the exact time as measured.

In the following section we first evaluate alignment and okurigana detection separately, then we evaluate okurigana detection, and finally we assess performance over the combined task.

5.1 Alignment

We first compare the accuracy of the various alignment algorithm variants, as given in Table 1. After some experimentation, parameter values of 0.05 for α, and 2.5 for w_s and w_u were found to yield the best results, and were hence used to generate the results we discuss here.

For each of the non-random heuristics, we expect that the iterative version will achieve higher accuracy than the non-iterative version, since the statistical model is rebuilt each iteration adding the best example from the last. As such, this represents a time/accuracy trade-off, a fact confirmed by our data (see Table 2). The gain — 2% in the case of TF-IDF, 4% for IDF alone — comes at the cost of an order of magnitude larger execution time, which also increases exponentially with the number of input entries.

In contrast, the Kanjidic model consistently achieves a very high accuracy regardless of the heuristic chosen. A large number of entries are immediately disambiguated by the Kanjidic model, thus initially improving accuracy and then facilitating use of more accurate statistics in the iterative algorithm without significant penalty to efficiency. We also expect the Kanjidic model's execution time to scale more moderately with the number of input entries than the original iterative algorithm, since a far lesser proportion of the entries require iterative disambiguation.

Comparing the individual heuristics at this stage, a surprise is that the IDF heuristic attains equivalent results to the TF-IDF heuristic, suggesting that broad occurrence of $\langle g, p \rangle$ pairs is a good indicator of their alignment probability. The TF heuristic in comparison performs worse than simply choosing randomly, suggesting that the proportion of times a grapheme occurs as the current $\langle g, p \rangle$ pair is a very poor indication of its alignment probability.

Model	Accuracy
Simple	98.1%
Kanjidic	98.3%
Co-occurrence	97.7%
Verb conjugation	97.7%

Table 3: Okurigana detection accuracy across models

5.2 Okurigana detection

We now compare the performance of our okurigana detection algorithms. All the algorithms we compare are linear in the size of the input and thus run in much less time than the alignment phase, thus efficiency is not a significant criteria in choosing between them. The accuracy found by each model is shown in Table 3.

Interestingly, the simple model which assumes that every potential case of okurigana *is* okurigana performs extremely well, beaten only by the addition of the Kanjidic common okurigana stems. Adding more information to the model about valid okurigana occurrences even reduces the accuracy slightly over our test data.

Rather than indicating blanket properties of these models, the results suggest properties of our testing data. Since it consists entirely of dictionary entries without the common hiragana particles which would occur in open text, this greedy approach is very suitable, and suffers few of the shortcomings which it would normally face.

In open text, we would consistently expect additional language features between lexical items which would break the assumptions made by our simple model, and thus reduce its accuracy dramatically. In contrast, the full verb conjugation model would then be expected to perform best, since it has the most information to accurately detect cases of okurigana even in the presence of other features.

5.3 Combined task

Selecting the two models which performed best on our test data, we can now evaluate the pair on the combined task. For the alignment subtask, the IDF heuristic with Kanjidic was used. For the okurigana detection subtask, the simple algorithm is used. The results are shown in Table 4.

A final accuracy of 96.2% was achieved, with the errors caused mostly in the alignment subtask. As predicted, grapheme gapping was a

ACCURACY (%)	Random	TF	IDF	TF-IDF
Iterative	47.8	23.7	94.7	93.4
Single-pass	47.3	23.6	90.5	90.8
Kanjidic	94.4	92.9	98.0	97.9

Table 1: Alignment accuracy across models

TIME (M:S)	Random	TF	IDF	TF-IDF
Iterative	0:10	24:10	22:47	21:54
Single-pass	0:10	0:11	0:09	0:10
Kanjidic	0:12	0:27	0:24	0:24

Table 2: Alignment execution time across models

Status	Count	Percentage
Correct	4809	96.2%
Incorrect	191	3.8%
→ Gapping	6	0.1%
→ Alignment	163	3.3%
→ Okurigana	22	0.4%

Table 4: Best model accuracy for the combined task

source of errors only in a small percentage of cases, justifying its exclusion from our model. This level of accuracy if equivalent to that of earlier models, yet it has been achieved with a much lower computational cost. Examples of incorrect alignment are given in Figure 4 below.

a.	*Output*	真 | 盛	ma-s | sa-ka-ri
	Correct	真 | 盛	ma-(s) | sa-ka-ri
		"full bloom"	
b.	*Output*	挟 | 撃 | chi	ha-sa | mi-u | chi
	Correct	挟 | 撃 | chi	ha-sa-mi | u | chi
		"pincer attack"	
c.	*Output*	赤-n | 坊	a-ka-n | bo-u
	Correct	赤 | n | 坊	a-ka | n | bo-u
		"baby"	

Figure 4: Examples of incorrect alignment in the combined task

Example (a) shows a grapheme gapping error, where the output, although correctly segmented and aligned, attributes the additional *s* sound to the 真 kanji instead of detecting it as a gapped grapheme. In example (b) we see a typical alignment error, where one kanji has been attributed part of the reading of another. Finally, example (c) gives an error in okurigana detection, where the *n* kana is erroneously detected as an okurigana suffix of the 赤 kanji.

6 Extensions

Although current work is suitable for use with the FOKS system, it is still untested on open text. The lack of suitable aligned data is the main obstacle to creating a system with wider applicability. Of the two subtasks, alignment should remain relatively unchanged in the move to open text, and we expect the IDF algorithm with Kanjidic to continue to perform well.

Okurigana detection remains the harder problem, for tasks which require it. The verb-conjugation model, despite its relatively poor performance for dictionary entries, suggests itself as the most fruitful approach to accurate detection for open text, and could easily be extended. In particular, the addition of conjugation suffixes of high-frequency irregular verbs would be a straightforward way to boost accuracy.

7 Conclusion

We have decomposed the GP alignment task into an alignment subtask and an okurigana detection subtask, and explored various algorithm variants for use in both. In particular, the iterative IDF heuristic with a Kanjidic reading model provided the best accuracy in significantly less time than the original algorithm. For the okurigana detection subtask, a simple model outperformed more complicated models of conjugation due to peculiarities of dictionary entries as input to alignment.

References

Jonathan Allen, Sheri Hunnicut, and Dennis Klatt. 1987. *From Text To Speech, The MITTALK Sys-*

tem. Cambridge University Press, Cambridge, UK.

Ricardo Baeza-Yates and Berthier Ribiero-Neto. 1999. *Modern Information Retrieval.* Addison Wesley / ACM press.

Timothy Baldwin and Hozumi Tanaka. 1999a. The applications of unsupervised learning to Japanese grapheme-phoneme alignment. In *Proc. ACL Workshop on Unsupervised Learning in Natural Language*, College Park, USA.

Timothy Baldwin and Hozumi Tanaka. 1999b. Automated Japanese grapheme-phoneme alignment. In *Proc. International Conference on Cognitive Science*, pages 349–354, Tokyo, Japan.

Timothy Baldwin and Hozumi Tanaka. 2000. A comparative study of unsupervised grapheme-phoneme alignment methods. In *Proc. 22nd Annual Meeting of the Cognitive ScienceSociety*, pages 597–602, Philadelphia, USA.

Slaven Bilac and Hozumi Tanaka. 2005a. Direct combination of spelling and pronunciation information for robust back-transliteration. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 413–424. January.

Slaven Bilac and Hozumi Tanaka. 2005b. Improving back-transliteration by combining information sources. In Keh-Yih Su, Junichi Tsujii, Jong-Hyeok Lee, and Oi Yee Kwong, editors, *Proc. 1st International Joint Conference on Natural Language Processing*, pages 216–223, January.

Slaven Bilac, Timothy Baldwin, and Hozumi Tanaka. 1999. Incremental Japanese grapheme-phoneme alignment. In *Information Processing Society of Japan SIG Notes*, volume 99-NL-209, pages 47–54.

Slaven Bilac, Timothy Baldwin, and Hozumi Tanaka. 2002. Bringing the dictionary to the user: the FOKS system. In *Proc. 19th International Conference on Computational Linguistics*, pages 85–91, Taipei, Taiwan.

Slaven Bilac. 2002. Intelligent dictionary interface for learners of Japanese. Master's thesis, Tokyo Institute of Technology.

Alan W. Black, Kevin A. Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *Proc. 3rd ESCA Workshop on Speech Synthesis*, pages 77–80, Jenolan Caves, Australia.

Jim Breen. 1995. *Building an electronic Japanese-English dictionary.* Japanese Studies Association of Australia Conference (`http://www.csse.monash.edu.au/~jwb/jsaa_paper/hpaper.html`).

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Grzegorz Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In Alexander Gelbukh, editor, *Proc. 4th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 432–443, Berlin. Springer-Verlag.

Christopher D. Manning and Hinrich Schütze. 2000. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts.

V. Pagel, K. Lenzo, and A.W. Black. 1998. Letter to sound rules for accented lexicon compression. In *Proc. 5th International Conference on Spoken Language Processsing*, pages 252–255.

Eric Rosen. 2003. Systematic irregularity in Japanese rendaku: How the grammarmediates patterned lexical exceptions. *Canadian Journal of Linguistics*, (48):1–37.

T. Sejnowski and C. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Richard Sloat. 1996. Multilingual text analysis for text-to-speech synthesis. *Natural Language Engineering*, 4(2).

Timothy J. Vance. 1987. *An Introduction to Japanese Phonology.* SUNY Press, New York.

Jianna Jian Zhang, Howard J. Hamilton, and Nick J. Cercone. 1999. Learning english grapheme segmentation using the iterated version space algorithm. In Andrzej Skowron and Zbigniew W. Raś, editors, *Proc. 11th International Symposium on Methodologies for Intelligent Systems*, Warsaw, Poland. Springer-Verlag.

Statistical Interpretation of Compound Nominalisations

Jeremy Nicholson[†] and **Timothy Baldwin**[‡]

[†‡] Dept. of Computer Science and Software Engineering
University of Melbourne
Victoria 3010 Australia

[‡] NICTA Victoria Lab
University of Melbourne
Victoria 3010 Australia

{jeremymn,tim}@cs.mu.oz.au

Abstract

This paper presents a method for detecting compound nominalisations from open data, and providing a semantic intepretation. It uses a statistical model based on confidence intervals over frequencies extracted from a large, balanced corpus. Using three paraphrases of the given compound nominalisation, and interpretation preferences of its components, the algorithm achieves about 70% accuracy in classifying the semantic relationship as one of SUBJECT, and OBJECT, and 57% between SUBJECT, DIRECT OBJECT, and PREPOSITIONAL OBJECT.

1 Introduction

Compound nouns have been a thorn in the side of computational linguistics since its inception, as it is almost impossible to avoid the issue of compound noun interpretation in any language task with a semantic or lexical semantic dimension. For example, an information extraction task may need to predict the semantic divergences between the compound nouns *news print* ("cheap paper on which newspapers are printed"), *thumb print* ("impression of the pattern on a thumb"), and *colour print* ("printed matter in colour").

Interpreting these divergences has become yet another instance of disaccord between the empiricists and rationalists of theoretical and computational linguistics. While the rationalists contend that compound nouns can be semantically described by some small, hand-crafted set of relations, the empiricists point to discordant examples which defy such natural sets, and rely on data sets for the necessary description. The rationalists then call this approach biased and brittle.

Needless to say, the argument is not one which will be resolved here. What we do hope to shed some light on is the applicability of an empiricist approach to the interpretation of an important subclass of compound nouns: **compound nominalisations**, or those compound nouns whose head derives from a verb. One example is *pattern generation*, where *generation* is derived from *generate*. These compounds tend to have a clearer semantic definition and are well-suited to techniques based on corpus statistics.

We propose a method for taking English[1] raw text input, detecting the compound nouns within, and applying a semantic interpretation for the compound nominalisations.

Section 2 details some of the previous work performed in the task. Section 3 outlines the resources used in our study. Section 4 describes the method we applied, with an analysis of the results in Section 5, and a discussion in Section 6.

2 Background

The first notable work on interpreting compound nouns focussed on the development of discrete semantic classes with which to classify all compound nouns: Levi (1978) proposed a nine-way classification, Warren (1978) identified a basic set of twelve paraphrases, Leonard (1984) developed an eight-way typology, and Finin (1980) put forth a much larger number of possible roles. This early research established a basic dichotomy in classification approaches: semantic class-based approaches (e.g. *linguistics textbook* $\equiv$ TOPIC(*textbook*) = *linguistics*) and syntactic paraphrase-based approaches (e.g. *linguistics textbook* = *textbook* <u>*on*</u> *linguistics*). A basic assumption underlying both approaches is that all compound nouns can be classified according to a finite set of relations, although researchers rarely agree on the number and ele-

[1] Although this paper focusses on English compounds, the phenomenon occurs readily in other languages, such as German, Modern Greek, Japanese, and Welsh. With some caveats for morphology and syntax, our concepts still apply.

Proceedings of the Australasian Language Technology Workshop 2005, pages 152–159,
Sydney, Australia, December 2005.

ments. Many identify nominalisations as a subclass, or include subjective and objective relations, which imply deverbal forms.

Automatically interpreting compound nouns has usually taken one of two approaches: one conceptual, in that the modifier fills a slot according to the concept of the head (Finin, 1980; McDonald, 1982); the other rule-based, in that the relation is selected by the first applicable rule taken from a series (Leonard, 1984; Vanderwende, 1994). Few, however, include evaluation of their systems — notable exceptions are the rule-based systems of Leonard (1984), who achieves 76% accuracy over a training set, Vanderwende (1994), who reports 52% over a test set for thirteen relations, and the concept-based work of Rosario and Hearst (2001), scoring 70% in a specific domain.

The first notable work on statistically interpreting compound nouns was that of Lauer (1995), who used prepositional paraphrases as an interpretation model, similar to those of Leonard (1984). Another useful element of this work was that of automatic compound noun bracketing, which has allowed work since to dismiss ternary and higher-arity compounds as a solved problem, and reduce the task to considering interesting binary compounds only. Lauer's interpretations get 47% accuracy over a set of twelve paraphrases, explicitly excluding subjective and objective relations. Lapata (2002), performs much better using a combination of a probabilistic method with a decision tree learner, to achieve 86% accuracy, albeit on a far better-defined and far simpler two-way classification task. Moldovan et al. (2004) get an F-score of 43% using statistical techniques across a wide semantic space. Finally, Grover et al. (2005) use a technique similar to that of Lapata to achieve 77% accuracy in a domain-specific setting over a broader thirteen item set.

Lapata (2002) and Grover et al. (2005) provide the usual statistical interpretation of a given compound nominalisation: corpus frequencies are derived for the verb-argument pair of a given deverbal head and modifer across a list of semantic relations. The selected relation is that which has the most attested instances in the corpus. To counter the problem of data sparseness, they examine the influence of backing-off, class-based, and distance-weighted smoothing, which are not found to perform significantly differently.

Automatic detection of compound nouns and compound nominalisations has had much less analysis than their interpretation, partly because detecting simple compounds is usually considered trivial, and reliably detecting nominalisations requires a semantic interpretation. Consider *corner piece* (of a puzzle): a naive system can certainly identify this contiguous noun sequence in an NP as a compound noun, but correctly dismissing a nominalisation interpretation requires semantic analysis of the absurdity of "*corner pieces [ST]", "*[ST] pieces the corner", "??[SO] pieces together the corner", and so on. Nonetheless, examination performed by Leonard (1984) notes the increasing frequency of compound noun usage over the past 250 years, and Grover et al. (2005) note that 72% of a small sample of sentences contained one or more compound nouns in a domain in which they are prevalent. 35% of these are compound nominalisations.

3 Resources

3.1 Tools

Similarly to both Lapata (2002) and Grover et al. (2005), we use the British National Corpus (BNC: Burnard (2000)), but only the 90M-token written component. We parsed the corpus using RASP (Briscoe and Carroll, 2002), a tag sequence grammar-based stochastic parser, in order to extract the corpus frequencies for use in disambiguation. We use tagging and chunking tools built with fnTBL 1.0 (Ngai and Florian, 2001) over the BNC independently, and for use in the detection of compound nominalisations.

3.2 2-Way Classification

We attempt to replicate the experiment performed by Lapata (2002), who manually extracted and annotated a sample of 796 binary compound nominalisations out of about 170,000 candidates automatically identified in the BNC.

In the original Lapata data, the underlying verb form of the head noun was uniquely identified using a combination of CELEX (Burnage, 1990) and NOMLEX (Macleod et al., 1998) data, sometimes resulting in sub-optimal results (e.g. the base verb of *question* is given as *quest*). In order to ameliorate such quirks in morphological analysis and expand the coverage of our method, we mined CELEX and NOMLEX, and also the word clusters in CATVAR (Habash and Dorr, 2003) for morphologically-related noun–verb pairs. This culminated in a total of about 14,000 deverbal nouns, many of which are listed

with multiple base verb forms (e.g. *divination* is listed as all of *divine*, *divinise* and *divinize*).

To validate the data for consistency, we removed those nominalisations which were not associated with a context sentence in the data set, those which did not occur in the same chunk in that sentence, according to the chunker above, those for which we did not find a verbal form (e.g. *decision-maker*), and those consisting of one or more proper nouns. We were left with 695 items which were classified as one of SUBJ or OBJ interpretations.

3.3 3-Way Classification

We also attempt to replicate the experiment performed by Grover et al. (2005), but in a domain-inspecific environment. To do so, we extract 1000 sentences randomly from the BNC which are then examined for compound nouns. About 32% of these contained at least one compound, much lower than the number in the biomedical domain of Grover et al. (2005).

We overtly exclude compounds which were of higher arity than two, (e.g. *silk jersey halter-neck evening dress*) and those consisting at least in part of proper nouns, similarly to the two-way task. They represented about 7.5% and 25% of the total compounds in the sample space respectively. The rest we classified according to the relations in Table 1: that of subject, direct object, prepositional object, not verbal (where the head does not have a verbal form), and not applicable (where the modifier is not the argument of the verbal head in an acceptable paraphrase).

We thereby collated a small data set, that of 129 items which occurred in a nominalisation relationship. The kappa coefficient, where the raw agreement is corrected for chance agreement (Carletta, 1996), between three annotators was $\kappa = 0.83$ ($N = 129, k = 3$) in detecting noun compounds.[2] This corresponds to a unigram agreement between the judges of 98.4%. Compared to the gold standard, the annotators had a mean precision of 92.5% and recall of 84.8% in detection of compound nouns, and 78.8% mean accuracy in semantic classiification of the compound nominalisations within.

4 Proposed Method

We propose an algorithm to detect compound nominalisations based on the output of a chunker, and then interpret each detected compound nominalisation by way of corpus evidence.

[2] $\kappa \geq 0.8$ indicates good agreement.

4.1 Detection of Compound Nominalisations in Open Data

To detect compound nominalisations in open data, we examine sequences of nouns that occur with the same chunk. Hence, we chunk parse a given sentence, and check for noun chunks with common noun modifiers immediately preceding the chunk head.

Next, we perform a table lookup over the head of each compound noun to check if it is contained in the combined set of deverbal nouns mined from NOMLEX, CELEX and CATVAR. If the head noun is not found to be deverbal, we conclude that the compound noun is not a compound nominalisation.

While our combined set of deverbal nouns provides excellent coverage, it suffers from low precision, largely because of CATVAR lacking explicit word-to-word derivational information. That is, we are able to access clusters of words which share the same stem, but have no way of checking for direct derivational correspondence between a given noun and verb. As a result, the output of the filter tends to have excellent recall, but diminished precision. We combat this effect by additionally checking for the plausibility of a subject or object interpretation against corpus data and thresholding over the probability for each interpretation type.

We evaluated the detection method by attempting to re-extract the gold standard two-way classification data from the BNC, and over our annotated data set for the three-way classification.

4.2 Paraphrase Tests

Lapata (2002) and Grover et al. (2005) provide the usual statistical interpretation of a compound nominalisation: that of the most attested relation in corpus frequencies for the verb-argument pair.

Other paraphrases are also used, however: as Lauer (1995) notes, the system by Leonard (1984) has paraphrasing as a goal, whereby *mountain vista* is interpreted via paraphrasing as *vista of a mountain or mountains*. Lauer himself also attempts to paraphrase compounds based on corpus statistics.

We notice that the other direction is also productive: instances of *vista of mountains* occur in the corpus, and they supply evidence for the reading "view mountains". We can thus form paraphrase tests to influence our interpretations.

Table 1: Classes of Compounds in the Sample Data

Class	Example	Frequency	
SUBJ	*eyewitness report*	22	(6.4%)
DOBJ	*eye irritation*	63	(18.2%)
POBJ	*side show*	44	(12.8%)
NV	*scout hut*	58	(16.8%)
NA	*memory size*	158	(45.8%)

We can search plain text for instances where the head noun is separated from the modifier by the preposition, and count corpus frequencies from these. For the preposition *by*, we assume a subject interpretation, as samples like *passage by animals* for *animal passage* imply that it is the animals that are passing. For the preposition *of*, we assume an direct object interpretation, as samples like *speaker of language* for *language speaker* imply that there is someone or something that speaks the language. Other prepositions separating the head and modifier contribute to a prepositional object interpretation, as samples like *operation on leg* imply that someone or something operates on a leg.

A second, related paraphrase test is for adjectival participles of the verbal form of the head connected to the modifier noun. In this case, present participles like *[the] passing animals* contribute to the subject interpretation, and past participles like *[a] spoken language* contribute to the direct object interpretation. There are no sensible cases in this test for prepositional objects, as *?operating on leg* would almost certainly be termed an indirect object relation by RASP, and included in our standard frequency counts.

A possible drawback to the prepositional test is losing phrasal verbs which legitimately take *by* or *of*. As well, paraphrases in this form blur somewhat in current English. Consider *child behaviour*, where a child behaves. Instances of *?behaviour by child* are greatly overwhelmed by occurrences of *behaviour of child* and *child's behaviour*. Despite examples such as this, this test is indicative of most paraphrases in the language.

4.3 Statistical interpretation

We interpret compound nominalisations by considering pairwise SUBJ vs. DOBJ and SUBJ vs. POBJ. First, we make the null hypothesis that the probabilities of all relations are equal, i.e. $P(rel_A \mid (rel_A \cup rel_B)) = 0.5$. We then consider each occurrence of a verb–noun pair to be a normally-distributed binomial trial for the two relations under consideration.

We derive our selection preferences based on the largest confidence interval between that of the SUBJ-DOBJ comparison (as Lapata (2002)), and that of the SUBJ-POBJ comparison.

A **Confidence Interval** P is the region under a normal curve with mean μ and standard deviation σ between $[\mu - n\sigma, \mu + n\sigma]$, where n is the z-score of a trial. Kenney and Keeping (1962) show that:

$$P = \frac{2}{\sqrt{\pi}} \int_0^{n/\sqrt{2}} e^{-t^2} dt \qquad (1)$$

where $t \equiv (x - \mu)/\sqrt{2}\sigma$, so as to normalise the curve. We observe that P is strictly increasing on n, so choosing the largest confidence interval from a set is simply a matter of choosing the largest z-score.

For a large set, calculating the z-score exactly is very costly. Instead, we estimate the sample z-scores for our observed trial by way of the binomial approximation to the normal distribution. Considering two relations at a time, having equal probability from the null hypothesis, our sample mean is the arithmetic mean of the frequencies, and our sample standard deviation is half of the square root of twice this number. The z-scores are then: $Z = \frac{f-\mu}{\sigma}$.

For example, consider the compound nominalisation from the Lapata data set *adult provision* found in the BNC in the following context: *...protecting someone's rights in the justice system (for example, appropriate adult provision).* We attempt to interpret the compound nominalisation, relative to the verbal forms *provide* and *provision*. *provision adult* is not productive, while *provide adult* gives the counts seen in Table 2.

From this, the highest z-score is Z_{PS}, for the prepositional object interpretation, which coincides with the correct reading "provide for adults", and the gold-standard tag OBJECT. The correct reading here would not have been

Verb-noun	SUBJ	DOBJ	POBJ	Z_{SD}	Z_{DS}	Z_{SP}	Z_{PS}
adult provision	7	5	18	0.58	-0.58	-2.20	2.20

Table 2: Z-scores for sample verb–noun pairs extracted from the BNC

captured by a simple SUBJ-DOBJ comparison, as Lapata would perform.

It is, however, not the case that we wish to examine prepositional object interpretations in every instance. If a verb does not take any prepositional objects at all, they will not occur in the data, and calculating the SUBJ-POBJ comparison will not be meaningful, and may introduce incorrect interpretations if it has a higher z-score than the SUBJ-DOBJ interpretation. As such, we construct a list of prepositional verbs derived from WordNet, COMLEX, the ERG, and the Longman Phrasal Verb Dictionary, and we can choose to apply the SUBJ-POBJ z-scores if the verb in question coincides with one of these.

4.4 The Algorithm

For a given detected compound nominalisation, we perform a number of steps to attempt to arrive at an interpretation.

First, we derive a set of verbal forms for the head using the table lookup from NOMLEX, CELEX, and CATVAR, as mentioned above, and note whether any of the forms occur in our set of prepositional verbs. If NOMLEX indicates that the head absorbs one of the possible interpretations, we automatically assume that the opposite interpretation is correct. For example, in *license holder*, the head absorbs the SUBJ relation, so we are left with OBJ. In *business employee*, *employee* absorbs the DOBJ relation, so we consider only SUBJ or POBJ.

This occurs for 8.9% of our compounds in the binary set, with all but one of them accurate (*woman referee*, who does not referee women, but is a woman who referees). In the ternary set, 6.2% of the compounds have such an interpretation, again with all but one accurate (*immigrant worker*, who is an immigrant who works).

This is similar to the suffix indications used by Lapata (2002), and the affixes used by Grover et al. (2005). Lapata identifies 12.9% of her set as having one of *-er, -or, -ant* suffixes, leading to an OBJECT interpretation, or an *-ee* suffix, leading to a SUBJ interpretation. Grover et al. identify that *-er* affixes receive an DOBJ relation, and *-or, -our* a SUBJ. These are also features to a decision tree learner. NOMLEX only captures a portion of these, but a head can have one of the endings without demanding such an interpretation. For example, *transfer* ends with *-er*, but does not take a DOBJ relation in *bank transfer*.

Next, we normalise the lemmas and attempt an interpretation. We acquire subject, direct object, and prepositional object counts for the modifier and verbal head pair, for each individual verbal form, as well as counts for the prepositional and participial paraphrase tests. We then calculate each of the four z-scores $Z_{SD}, Z_{DS}, Z_{SP}, Z_{PS}$ for the three tests, and select the interpretation having the highest z-score from the set.

If the best z-score for two differing interpretations are equal, we employ the simplest smoothing method from Lapata (2002): backing-off. Lapata assumes that the ratio of the counts can be approximated by backing-off to the counts of the modifier noun:

$$P(rel \mid n_1, n_2) = \alpha \frac{f(rel, n_1)}{f(n_1)} \qquad (2)$$

The reason for this being superior to backing-off to the verb counts is not immediately clear, so we compare backing-off to those counts as well. We also examine another form of "backing-off" — that of the deverbal head counts, which cannot be directly examined from the corpus. Instead, we mine the BNC for sentences which contain the head in an instance which we can interpret using corpus frequencies, and count frequencies based on the number interpreted as SUBJ, DOBJ or POBJ.

Regardless of the chosen method, the need for backing-off occurs quite often in practice, as some 16% of the Lapata data set has no instances of the verb–noun pair attested to in the corpus, as well as 36% of the open data set.

We implement backing-off by examining the interpretation preferences, again using confidence intervals. The preference for the modifier noun or verbal head is the greatest z-score from the counts of all instances of that noun or verb occurring as or with a subject, direct object, or prepositional object. The preference for the deverbal head is the greatest z-score from the counts of all instances of that head occurring with a modifier for which we can provide

an interpretation of subject, direct object, or prepositional object using corpus frequencies.

5 Experimental Results

5.1 Detection

We evaluated the detection method first by running it over the contextual sentences and seeing how many of the compound nominalisations in the normalised Lapata data set were detected by our method. On this data set, we were able to detect 88.8% of the instances.

On the open text data set, our algorithm detected 69.8% of the SUBJ, DOBJ, and POBJ compounds. The more general compound nouns were detected with a precision of 86.6% and a recall of 77.0%, comparable to the human annotators.

The primary causes of data instances being missed by our method were that the head noun was not contained as a nominalisation in our combined lexicon (e.g. *decision-maker*), or the input had been misanalysed by the chunker. Many of the latter errors were caused by poorly punctuated sentences in the corpus (e.g. *citizen charter* in *Ministers ' views were set out in the citizens charter*), with some mistakes made by the POS tagger (e.g. calling *covers* a verb in *leopardskin seat covers*).

As for the various relations in our set above, the detection algorithm discovers NV relations with a precision of 58.4% and a recall of 77.6%, NA relations with a precision of 65.1% and a recall of 53.2%, and SUBJ/ DOBJ/ POBJ with a precision of 57.1% and a recall of 49.6%.

Recalling that CATVAR lacks derivational information, and therefore tends to broaden coverage at the cost of precision, we examine the detection procedure without CATVAR in the deverbal filter. This algorithm discovers NV with a precision of 22.2% and a recall of 82.8%, NA with a precision of 26.5% and a recall of 5.7%, and nominalised relations with a precision of 72.3% and a recall of 26.4%. Indeed, the precision in detecting nominalisations increases, but at a substantial cost of recall.

Errors cascade in this definition, so that a noun incorrectly given a verbal form causes a false negative in NV and a false positive in NA, and so on. This explains the loss of precision in the latter classifier, when a compound is classified as NV from not recognising that the head is verbal and is an NA relation.

These figures occur for a baseline classifier, where NV implies that the noun was not in our deverbal list, NA implies that the verb-noun pair was not attested in the corpus, and one of SUBJ, DOBJ, or POBJ otherwise.

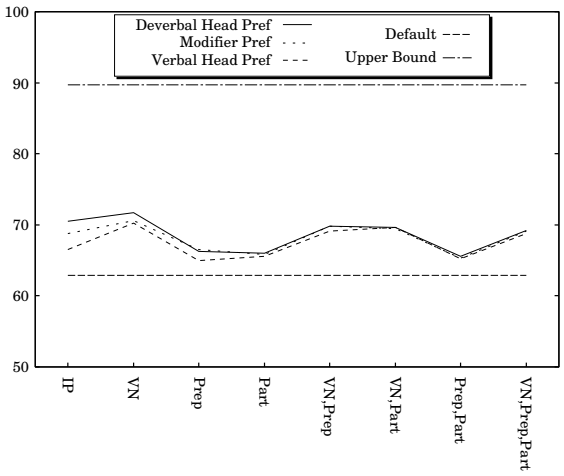


Figure 1: Disambiguation Accuracy for the 2-Way Classification Task

5.2 2-Way Classification

The data set from Lapata had 695 compound nominalisations: of these, 258 had a SUBJ interpretation and 437 had a OBJECT interpretation. So, the baseline of choosing the OBJECT relation each time has a performance of 62.9%.

Figure 1 shows the performance for the three paraphrase tests: using verb–noun pair counts (*VN*), using participial paraphrases (*Part*), and using prepositional paraphrases (*Prep*). We can back-off to the verbal head, modifier noun, or deverbal head in each case. We also contrast these with the performance of the baseline (*Default*) and the interpretation preferences when used without the paraphrase tests (*IP*).

The prepositional and participial paraphrases, when used on their own, do not perform significantly better than the baseline ($\chi^2 = 1.97, p \leq 0.2$). This is not overly surprising, as coverage over the data set is quite poor: only 40% could be given an interpretation using one test, and 58% for both tests — far lower than the 84% for the verb–noun pairs.

The verb–noun counts are significantly better than the baseline ($\chi^2 = 9.45, p \leq 0.01$), and also slightly improve upon the figures recorded by Lapata for backing-off — namely, 69.6% over the test set and 68.0% over the entire data set.

Interestingly, backing-off to the deverbal head is consistently slightly better than backing-off to the modifier noun or verbal head, at the cost of extra examination of the corpus. Also, the best

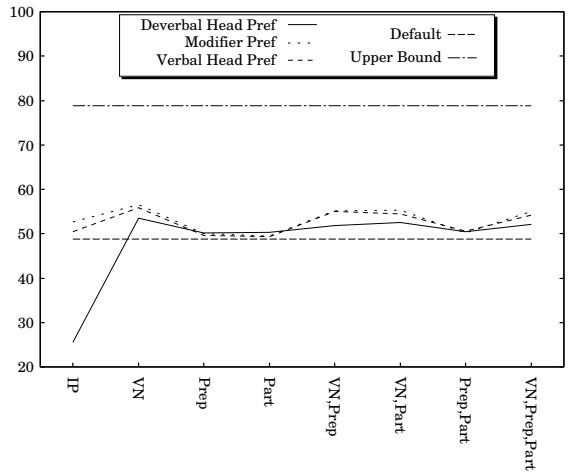


Figure 2: Disambiguation Accuracy for the 3-Way Classification Task

performance occurs for the verb–noun pairs using backing-off to the deverbal head, but including the paraphrases does not give results that are significantly different to these.

5.3 3-Way Classification

Our collated data set had a baseline of 48.8%, that of selecting DOBJ each time. Figure 2 shows the results of our experiments, similarly to the two-way classification.

Again, the later paraphrases do not perform significantly better than the baseline ($\chi^2 = 0.39, p \leq 1$), while the verb–counts do perform better ($\chi^2 = 4.01, p \leq 0.05$)), and slightly improve on the figures reported by Grover et al. (2005) using frequency counts and affixes.

In this case, backing-off to the modifier noun proves better than to either the verbal or deverbal head, and the further paraphrases do not improve the performance of the frequency counts. Also of note is the fact that the deverbal head preferences on their own perform quite poorly here, in stark contrast to their performance on the binary task.

6 Discussion

We presented a method for detecting compound nominalisations and deriving an interpretation in open data for a two-way classification task between an underlying subject or object semantic relation between a head noun and its modifier, and for a three-way task between subject, direct object, and prepositional object relations. This achieved about 70% accuracy in the two-way task, and 57% in the three-way task, using a statistical measure based on z-scores — the confidence interval — in selecting one the relations. We investigated the performance of three paraphrase tests across the BNC: the frequency of the modifier noun and verbal head pair, the frequency of prepositions separating instances of the head and modifier nouns, and the frequency of the verbal head occuring as a participial adjective connected to the modifier noun. We also examined the interpretation preferences of the modifier noun independent of the verbal head and the head as both verbal and deverbal, and used these for backing-off the paraphrase counts. Interestingly, the performance of the different tests was not altogether dissimilar: the best-performing set for the two-way classification task was the deverbal interpretation preferences for the verb–noun counts, and verb–noun counts backed-off to the modifier in the three-way task.

This method was useful in that it can act over open data to detect and interpret compound nominalisations. It performed comparably to the human annotators in detecting compound nominalisations, and the generosity of CATVAR in classifying a deverbal noun was avoided to some extent by thresholding over the probabilities of interpretation in the corpus frequencies.

Our method also extended the scope of the interpretation of nominalisations away from the need for pre-filtered data, such as was necessary for the two statistical works of interpretation using corpus frequencies, that of Lapata (2002) and Grover et al. (2005). Our method also does not presuppose hand-crafted parsed data, which is necessary for both of these investigations. It also can operate more or less independently of the domain in which it is used, as we demonstrated in sampling random sentences over a balanced corpus.

The utility of the two proposed paraphrases tests, using prepositions and participles, is that they do not require parsed data to acquire corpus frequencies. This allows us to take counts for these tests from the Web, which we believe will alleviate the data sparseness problem for these tests to some extent.

The fact that the performance of the algorithm (70% and 57% for the two tasks) does not match the state-of-the-art performance by these works (86% and 77% respectively) does not worry us too much, as they match the simple performance of the works, and these better figures included a variety of class-based smoothing tasks, contextual features, and machine learning

tools.

Acknowledgements

We would like to thank Mirella Lapata for generously contributing her data set to the study.

References

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

Gavin Burnage. 1990. CELEX: A guide for users. Technical report, University of Nijmegen.

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

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

Tim Finin. 1980. The semantic interpretation of nominal compounds. In *Proc. of the First National Conference on Artificial Intelligence*, pages 310–315, Stanford, California. AAAI Press.

Claire Grover, Mirella Lapata, and Alex Lascarides. 2005. A comparison of parsing technologies for the biomedical domain. *Journal of Natural Language Engineering*, 11(01):27–65.

Nizar Habash and Bonnie Dorr. 2003. A categorial variation database for English. In *Proc. of the 2003 Human Language Technology Conference of the North American Chapter of the ACL*, pages 17–23, Edmonton, Alberta.

John F. Kenney and E. S. Keeping, 1962. *Mathematics of Statistics, Pt. 1*, chapter 11.4, pages 167–9. Van Nostrand, Princeton, New Jersey, 3rd edition.

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

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

Rosemary Leonard. 1984. *The Interpretation of English Noun Sequences on the Computer*. Elsevier Science, Amsterdam.

Judith Levi. 1978. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. NOMLEX: A lexicon of nominalizations. In *Proc. of the 8th International Congress of the European Association for Lexicography*, pages 187–193, Liege, Belgium.

David McDonald. 1982. *Understanding Noun Compounds*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *Proc. of HLT-NAACL 2004 Workshop on Computational Lexical Semantics*, pages 60–7, Boston, USA.

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

Barbara Rosario and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proc. of the 6th Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, USA.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *Proc. of the 15th International Conference on Computational Linguistics*, pages 782–788, Kyoto, Japan.

Beatrice Warren. 1978. *Semantic Patterns of Noun-Noun Compounds*. Acta Universitatis Gothoburgensis, Göteborg, Sweden.

Paraphrase Identification by Text Canonicalization

Yitao Zhang and **Jon Patrick**
School of Information Technology
University of Sydney
Sydney, Australia, 2006
{yitao, jonpat}@it.usyd.edu.au

Abstract

This paper proposes an approach to sentence-level paraphrase identification by text canonicalization. The source sentence pairs are first converted into surface text that approximates canonical forms. A decision tree learning module which employs simple lexical matching features then takes the output canonicalized texts as its input for a supervised learning process. Experiments on the Microsoft Research (MSR) Paraphrase Corpus give comparable performance to other systems that are equipped with more sophisticated lexical semantic and syntactic matching components, with a Confidence-weighted Score of 0.791. An ancillary experiment using the occurrence of nominalizations suggests that the MSR Paraphrase Corpus might not be a rich source for learning paraphrasing patterns.

1 Introduction

Paraphrase identification is the task of recognizing text fragments with approximately the same meaning within a specific context. It has been recently proposed as an application-independent framework for measuring semantic equivalence in text, which is critical to many natural language systems like Question Answering, Information Extraction, Information Retrieval, Document Summarization, and Machine Translation.

This paper proposes an approach to identifying sentence-level paraphrase pairs by transforming source sentences into more canonicalized text forms. By "canonical form", we mean a transformed text which is more generic and simpler in someway than the original text, following the idea of restricted languages. For example, the sentence

> Remaining shares will be held by QVC's management.

is transformed into a more canonicalized form by changing it from the passive to active voice producing

> QVC 's management will hold Remaining shares.

which is more common in Subject-Verb-Object (SVO) languages like English, while the Passive Voice in the source sentence usually occurs in scientific and business text where a more formal writing style is used.

This approach is consistent with Chomsky's Transformational Grammar, in which syntactically different, but semantically equivalent sentences can be related by their identical deep semantic structures (Chomsky, 1957). However, it is generally difficult to efficiently analyze any corpus by using the Transformational Grammar due to its high complexity and computational overhead (Hausser, 2001). In our approach, we only attempt to transform parts of the surface structure into a more generic text representation within the context of the paraphrase identification problem. The underlying hypothesis of this approach is that if two sentences are paraphrases of each other, they have a higher chance of being transformed into similar surface texts than a pair of non-paraphrase sentences.

In this paper, only a set of limited canonicalization rules have been applied as a preliminary attempt to evaluate the effectiveness of the methodology. The objective is not to create grammatically correct text sequences from source sentences, but to enable the true paraphrases to share as much surface text, both lexically and syntactically, as possible. Despite this simple model, experiments on the MSR Paraphrase Corpus nevertheless show comparable results to those scores reported in the recent ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment (2005). They also show that this approach increases the Recall rate of the system quite significantly.

Proceedings of the Australasian Language Technology Workshop 2005, pages 160–166,
Sydney, Australia, December 2005.

2 Background

Recent work on sentence-level paraphrasing generally views the problem as one of identifying bidirectional entailment in text pairs. Given an entailment text T and a hypothesis text H, T entails H if H can be inferred from the contents of T (Dagan et al., 2005). A pair of sentences is therefore considered as a paraphrase pair if the entailment relationship holds from both directions.

However, this strict mutual entailment relationship does not hold in most naturally occurred sentence-level paraphrases. Recent attempts on extracting paraphrase pairs from the web, notably the MSR Paraphrase Corpus (Dolan et al., 2004), have shown a large quantity of "more or less semantically equivalent" paraphrase pairs, such as the examples in Table 1. In the paraphrase pair "913945-914112", "Dewhurst" in the first sentence cannot be inferred from the second without giving the specific context knowledge that this person is someone belongs to the "committee". In the pair "420631-420719", the first sentence does not include any information that the minister is Saudi which occurs in the second sentence. Human judges have generally shown little difficulty in identifying these loose semantically equivalent sentence pairs as paraphrases. A surprisingly high inter-rater agreement of 83% was reported in the construction of the MSR Paraphrase Corpus despite the rather vague guideline of identifying sentence-level paraphrases that was used. It suggests that human judges were only interested in the matching of main propositions in sentence pairs, while neglecting the existence of other non-entailed trivial contents.

Bar-Haim et al. (2005) decomposed the entailment task into two sub-levels, namely, lexical and lexical-syntactic. At the lexical level, for each word or phrase h in Hypothesis H, if h can be matched with a corresponding item t in Text T using either lexical matching, or a sequence of lexical transformations, then H and T are tagged as a true entailment pair. Lexical transformation rules include morphological derivations like nominalization (example "913945-914112" in Table 1, "proposal => propose"), ontological relations like synonym and hypernym, or world knowledge such as "Taliban => organization". At the lexical-syntactic level, entailment between H and T holds if both the lexical and syntactic relations in H are also found in T. The relations evaluated at the lexical-syntactic level include syntactic movement triggered by morphological derivation of words, passive to active voice transformation of verbs, co-reference in text, and the syntactic level paraphrases like "X was born in Y <=> X is Y man by birth". In an empirical analysis of the PASCAL Recognising Textual Entailment Challenge (RTE) corpus (Dagan et al., 2005), 240 sentence pairs were randomly chosen and tagged by human annotators based on the above criteria for semantic entailment. What they have found is that working on the lexical-syntactic level outperforms on the lexical level by a significant increase of the Precision score, namely, from 59% to 86%. However, the Recall rate shows only 6% improvement by switching from lexical to a lexical-syntactic level.

In a similar effort to evaluate the contribution of syntactic knowledge in the entailment task, Vanderwende et al. (2005) found that 37% of the RTE Entailment Corpus examples could be handled by syntax alone, assuming the existence of an ideal parser. With additional help from a thesaurus, this figure can be increased to 49%.

Corley and Mihalcea (2005) proposed a bag-of-words model for identifying entailments and paraphrases by measuring the semantic similarity of two texts. In their model, the semantic similarity of two text segments T_i and T_j is defined as a score function that combines the semantic similarities of nouns and verbs, the lexical similarities of other open class words, together with word specificities measured by the inverse document frequency metric derived from the British National Corpus. Experimental results on the MSR Paraphrase Corpus showed a 4.4% increase of system accuracy by incorporating semantic knowledge.

Inversion Transduction Grammars (ITG), which is previously proposed as a framework for machine translation, has also been applied in the context of the paraphrase and entailment task by Wu (2005). Without consulting any thesaurus, the Bracketing ITG model worked mainly on a syntactic matching level and achieved a Confidence-weighted Score of 0.761, which is 10% higher than the random baseline.

3 The Dataset

The Microsoft Research Paraphrase Corpus has been used throughout our experiments. It is the result of a recent effort to construct a large scale

Table 1: Examples of MSR Paraphrase Corpus

ID	Text1	Text2	Description
913945-914112	Dewhurst's proposal calls for an abrupt end to the controversial "Robin Hood" plan for school finance.	The committee would propose a replacement for the "Robin Hood" school finance system.	Nominalization
2484044-2483683	The tour plans to make stops in 103 cities before rallying in Washington on Oct. 1-2, and in New York City on Oct. 3-4.	The tour will stop in 103 cities before rallying in Washington on Oct. 1 and 2, and New York on Oct. 3 and 4.	Nominalization + Future Tense
420631-420719	Those reports were denied by the interior minister, Prince Nayef.	However, the Saudi interior minister, Prince Nayef, denied the reports.	Passive/Active Voice

paraphrase corpus for generic purposes (Dolan et al., 2004). It consists of 5,801 sentence pairs extracted from online newswire text, in which 3,900 are tagged as true paraphrases by human judges. This high proportion of occurrences of paraphrase pairs can be explained by the methodology used to create the corpus. In the construction of the corpus, edit distance is used as the only metric to filter out lexically unsimilar sentence pairs, which means the remaining instances have large lexical overlaps. As a consequence, although the MSR Paraphrase Corpus is rich in the number of paraphrase pairs, it is not enriched with a good variety of lexical and syntactic patterns. Weeds et al. (2005) argue that this "high overlap in words" makes it a poor source for studying the distributional similarity of syntactic paraphrasing patterns.

In an effort to substantiate this claim, we made an evaluation of the occurrence of nominalization, which is a classical linguistic device for paraphrasing, in both the MSR Paraphrase Corpus and the RTE Entailment Corpus. We used a semi-automatic method to calculate the occurrence of nominalizations. First we pos-tagged sentence pairs in the corpus and lemmatized all the verbs and nouns. If there was an exact string match between a lemmatized verb and a lemmatized noun in a sentence pair, we marked it as a candidate of nominalization, and asked human judges to verify it at a later stage. A walk-through example of finding nominalization is shown in Table 2.

This method gives a reliable lower bound on the occurrence of nominalizations in the corpora. The results are shown in Table 3. Notice that in the MSR training dataset only 60 true nominalizations exist in over 4,000 sentence pairs, compared to the number of 44 over 800 in the RTE testing dataset. This result suggests that the distribution of paraphrasing patterns in the MSR Paraphrase Corpus is likely to be below the normal distribution in natural text, or at least not that rich compared to a human constructed and balanced corpus. Therefore, it might not be a rich resource for studying the real distribution of features of naturally occurring paraphrases and Weeds et al.'s comments are justified.

Table 3: Occurrence of Nominalizations

	True Nominalizations	Corpus Size(sentence pairs)
RTE	44	800
MSR	60	4076

Despite these innate problems of the corpus, it is still by far the largest sample dataset of paraphrasing phenomenon, which provides a solid base for system testing. Therefore, we decided to focus our research on this corpus as the first stage of our experiments.

4 Experiments

This section describes the details of the two modules in the system, namely the text canonicalization module and the supervised learning module.

4.1 Text Canonicalization

The function of the text canonicalization module is to constrain the language choices, both at lexical and syntactic level, of any text that carries meanings. In this paper, only a set of limited canonicalization rules has been applied.

Number Entities Number entities include dates, times, monetary values, and other quan-

Table 2: An Example of Finding Nominalizations

ID	913945	914112
	Dewhurst/NNP 's/POS proposal/NN calls/VBZ for/IN an/DT abrupt/JJ end/NN to/TO the/DT controversial/JJ "/NNP Robin/NNP Hood/NNP "/NNP plan/NN for/IN school/NN finance/NN ./.	The/DT committee/NN would/MD propose/VB a/DT replacement/NN for/IN the/DT "/NNP Robin/NNP Hood/NNP "/NNP school/NN finance/NN system/NN ./.
Nouns	proposal=>propos, end, Robin, Hood, plan, school, finance=>financ	committee=>committe, replacement=>replac, Robin, Hood, school, finance=>financ, system
Verbs	calls=>call	propose=>propos
Candidate Nominalizations: (proposal, propose)		

Table 4: Passive to Active Voice

	id = 420631	id = 420719
Before transformation	Those reports were denied by the interior minister, Prince Nayef.	However, the Saudi interior minister, Prince Nayef, denied the reports.
After transformation	the interior minister, Prince Nayef denied Those reports.	unchanged

tities like percentages. In the experiments, the system will replace those number entities with generic tags in the text.

Passive/Active Voice In the passive to active voice transformation, the system first consults Minipar (Lin, 1998), which is a principle-base English parser, to get the parsed dependency tree structure of the text. Then it finds all the verbs in passive voice, together with their grammatical subjects and the objects. Finally, the system swaps the child nodes of the subjects and the objects of each verb. The canonicalized text is then created from the transformed syntactic tree. An example of passive to active voice transformation is shown in Table 4.

Future Tense The expression of future tense in text has also been canonicalized to constrain the lexical choices which refer to future action and willingness. An example of future tense usage in the MSR Paraphrase Corpus is given by the text pair "2484044-2483683" in Table 1. In the sentence, "plans to" and "will" both refer to the future actions the subject will be taking. They have to be canonicalized into the same surface text to create higher probabilities to be matched at a later stage. In the experiments, we compile a list of common words and phrase structures(like "plan to" and "be expected to") to be substituted by a single word "will", which the system defines as the generic expression of future actions.

4.2 Supervised Learning

At the supervised learning stage, the decision tree learning module of Weka (Witten and Frank, 1999) was used. The training dataset and the test dataset used in the experiments are the corresponding training and test dataset in MSR Paraphrase Corpus as described in Section 3.

Lexical Matching Features The features used in the supervised learning stage are

- *Longest Common Substring* measures the length of the longest common strings shared by two sentences. It is a consecutive sequence of words.
- *Longest Common Subsequence* measures the length of the longest common sequence of strings shared by two sentences. It does not require this sequence to be consecutive in the original text.
- *Edit Distance* describes how many edit operations (add, delete, or replace of a word token at a time) are required to convert a source text into a target text. The fewer edit operations needed, the less edit distance and the more lexical overlap of the two text segments.
- *Modified N-gram Precision* is also an important metric adopted from the BLEU

algorithm for evaluating machine translations (Papineni et al., 2001). It was originally proposed to capture both the accuracy and the fluency of a translated text with reference to a set of candidate translations. In the context of paraphrases, we try to calculate the modified n-gram precision from both directions of a sentence pair. For example, given the following sentence pair:

T_1: the the the the the the the.
T_2: The cat is on the mat.

we first define the modified count of an n-gram t in T_1 as the minimum between the occurrence of t in T_1 and the maximum occurrence of t in T_2. For instance, $Count_{\text{modified}}$("the") is 2 because the unigram "the" occurs only twice in the second sentence. The directional modified n-gram precisions from T_1 to T_2 is defined in Equation 1, in which m is the order of n-gram (up to trigram m=3 was used in our experiment), and $Count(k)$ simply counts the number of k in the source sentence T_1. We also calculated the directional modified n-gram precision score from T_2 to T_1, and used the average of the two directional precision as the modified n-gram precision of the sentence pair by Equation 2.

Moreover, our calculation of the above features is solely based on word token level. For instance, we use word n-gram instead of letter n-gram in calculating the modified n-gram precision.

$$mnp_{T_1} = \frac{1}{m}\sum_{i=1}^{m} -\log\left(\frac{\sum_{t\in\text{n-gram}_i} Count_{\text{modified}}(t)}{\sum_{k\in\text{n-gram}_i} Count(k)}\right) \quad (1)$$

$$mnp(T_1, T_2) = \frac{mnp_{T_1} + mnp_{T_2}}{2} \quad (2)$$

Evaluation Metrics To assess the system performance, we adopt the Confidence-weighted Score(CWS) as the main figure for our evaluations. CWS is defined in Equation 3

$$cws = \frac{1}{n}\sum_{i=1}^{n} \frac{\#\text{correct-up-to-i}}{i} \quad (3)$$

in which #correct-up-to-i is the number of correct tagging instances up to the current position i, and the test data samples are first ranked in decreasing order according to their confidence level of tagging judgments. The CWS metric generally rewards a system that assigns higher confidence values to correct tagging decisions than to those wrong ones (Dagan et al., 2005). Meanwhile, traditional machine learning metrics like accuracy, precision, recall, and F_1 values are also reported for better understanding of the system.

The Baselines Two baselines have been provided for the task. The first baseline system uniformly predicts true for paraphrase pairs. The second baseline system uses the lexical matching features in Section 4.2 on the original text pairs for the supervised learning stage.

5 Results

The experiment results are shown in Table 5. For comparison, scores of Wu (2005), and Corley and Mihalcea (2005)'s systems are also included in the table.[1]

For the two baseline systems, B2, which employs pure lexical matching features on the source text, outperforms B1, the system that uniformly predicts paraphrases, both in Accuracy by 6%, and in CWS by 12%. The B2 system also shows comparable results with respect to Wu, and Corley and Mihalcea's systems and sets a high standard as a baseline system. This further reveals the main characteristic of the MSR Paraphrase Corpus: paraphrase text pairs in the corpus share more lexical overlaps than non-paraphrase pairs.

Compared with B2, systems using canonicalized text, namely S1 - S7, generally suffer a slightly poorer performance in the Accuracy score. However, the Recall rate rises significantly in all systems except in S3 and S6. Interestingly, S3 and S6 also show the highest CWS score and the Precision score at the same time. This suggests that the canonicalization of future tense helps systems to make more precise and reliable tagging decisions. Canonicalization on Passive/Active voice (S2) also increases the Recall rate by almost 10% compared with B2. This suggests that a pure lexical matching system could be further improved by even some preliminary syntactic transformations. Number entity canonicalization helps to increase the Recall rate of the system. This could be explained

[1] Wu only reported the CWS score on MSR corpus in his paper, while Corley and Mihalcea did not report any CWS score in their paper.

Table 5: Experiment results on MSR Paraphrase Corpus

	CWS	Acc	Pre	Rec	F_1
Systems using Canonicalized Text					
S1: (a)number entities	0.740	0.692	0.713	0.898	0.795
S2: (b)passive/active	0.742	0.719	0.743	0.882	0.807
S3: (c)future tense	0.791	0.708	0.784	0.775	0.779
S4: (a)+(b)	0.739	0.697	0.716	0.900	0.798
S5: (a)+(c)	0.731	0.701	0.732	0.869	0.794
S6: (b)+(c)	0.791	0.709	0.784	0.776	0.780
S7: (a)+(b)+(c)	0.723	0.703	0.734	0.867	0.795
Baselines					
B1: Uniform	0.664	0.664	0.664	1	0.798
B2: LexicalMatch	0.783	0.723	0.788	0.798	0.793
Other Systems with Reported Scores					
Wu (2005)	0.761				
Corley and Mihalcea (2005)		0.715	0.723	0.925	0.812

by how the MSR Paraphrase Corpus was constructed. During the tagging process, source sentences were already pre-processed by replacing number entities with generic tags. Human judges then made their decisions based on the canonicalized text. While the dataset revealed to the public, the source text is provided instead of the data used by human judges.

In general, systems S1-S7 show competitive performance with respect to Wu, and Corley and Mihalcea's systems. Corley and Mihalcea's system gives a better Recall rate, which suggests the importance of introducing lexical semantics features in the system. Our approach currently does not model synonyms into any canonicalized form, therefore loses the possibility of capturing this lexical variance. On the other hand, neither Wu, nor Corley and Mihalcea's system outperforms the lexical matching system B2 in terms of CWS and Accuracy. This again suggests that the nature of the paraphrases in the corpus is that they share more lexical overlaps than non-paraphrases, rather than employing sophisticated syntactic paraphrasing patterns.

6 Conclusion

This paper proposes a text canonicalization approach to the paraphrase identification task. Our approach tries to tackle the problem on both the lexical and the grammatical level, as distinct from existing research which has concentrated on lexical analyses. Despite the simple transformation rules applied, this approach has shown competitive figures of system performance on the MSR Paraphrase Corpus with that reported in current state-of-the-art systems. Moreover, this method reports a significant increase in the recall rate of paraphrases compared with a system using non-canonicalized text. It clearly encourages the use of more conceptualized and more canonical syntax which tries to approximate the deeper semantic information of the original text.

However, further research is required to reveal how many transformation rules are needed for the task. It would also be interesting to develop an effective engineering method for managing the expanding canonicalization rule set. In the future, more work has also to be done to equip the system with lexical semantic knowledge from either manually constructed lexical databases like WordNet (Fellbaum, 1998), or other resources that automatically learned from corpora like VerbOcean (Chklovski and Pantel, 2004).

Acknowledgments

This paper was supported by the School of Information Technologies, University of Sydney. We would also like to thank the reviewers for their insightful comments, and all members of the Sydney Language Technology Group for their support.

References

Roy Bar-Haim, Idan Szpecktor, and Oren Glickman. 2005. Definition and analysis of intermediate entailment levels. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 55–60, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, Barcelona, Spain.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ido Dagan, Bernardo Magnini, and Oren Glickman. 2005. The PASCAL Recognising Textual Entailment Challenge. In *PASCAL Proceedings of the First Challenge Workshop, Recognizing Textual Entailment*, Southampton, U.K., April.

Bill Dolan and Ido Dagan, editors. 2005. *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. Association for Computational Linguistics, Ann Arbor, Michigan, June.

William B. Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of COLING 2004*.

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

Roland Hausser. 2001. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Springer.

Dekang Lin. 1998. Dependency-based Evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*, Granada, Spain, May.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation.

Lucy Vanderwende, Deborah Coughlin, and Bill Dolan. 2005. What Syntax can Contribute in Entailment Task. In *PASCAL Proceedings of the First Challenge Workshop, Recognizing Textual Entailment*, Southampton, U.K.

Julie Weeds, David Weir, and Bill Keller. 2005. The distributional similarity of sub-parses. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 7–12, Ann Arbor, Michigan, June. Association for Computational Linguistics.

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

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 25–30, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Words and Word Usage: Newspaper Text versus the Web

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

Abstract

This paper explores the differences in words and word usage in two corpora – one derived from newspaper text and the other from the web. A corpus of web pages is compiled from a controlled traversal of the web, producing a topic-diverse collection of 2 billion words of web text[1]. We compare this *Web Corpus* with the Gigaword Corpus, a 2 billion word collection of news articles. The Web Corpus is applied to the task of *automatic thesaurus extraction*, obtaining similar overall results to using the Gigaword. The quality of synonyms extracted for each target word is dependent on the word's usage in the corpus. With many more words available on the web, a much larger Web Corpus can be created to obtain better results in different NLP tasks.

1 Introduction

In corpus-based Natural Language Processing (NLP), the corpus is the primary representation of a language from which algorithms extract information and build linguistic models. Words and word usage in a corpus of newspaper text will differ from that of a corpus of web pages, due to the different genres of text and as an artifact of the corpus collection process.

Words are used differently across different medium and corpora. While newspaper text is usually written by experienced writers and carefully checked by editors for accuracy, few restrictions exist for web text. It has a wider range of writing styles and less adherence to formal grammar. Anyone with access to the web can create web pages and there is no restriction as to what topics are written about on the web. Thus, a web corpus will contain a wider range of topics than a newspaper corpus.

[1]In this paper, we report on the number of tokens after the corpus has been tokenised, counting both words and punctuation.

Some differences between corpora can be attributed to the collection process. A corpus of newspaper text is usually collected from a few publishers across a set period of time. This corpus will reflect the topics in the news over that period as well as the types of news targeted by the publishers (e.g. political, financial). A corpus of web pages can contain pages from any time before its creation. It will have a different distribution of dates than a newspaper corpus.

In this paper, we explore the difference between a corpus of newspaper text and a corpus of web text. We conduct three experiments to highlight some of the differences. First, the token types that exist within each corpus are examined and the vocabulary unique to each corpus is accounted for. We then analysed words frequent in one corpus that are infrequent in the other to reveal topic skew. Finally we extract synonyms for common nouns to show the word usage and topic coverage of the two corpora and to demonstrate the usefulness of web corpora.

2 Corpora

A *corpus* is a collection of text fulfilling some specified criteria. If a corpus is intended to be used for study of English or any other language, it must incorporate samples across all usage of the target language. For example, an English corpus should include both written and spoken English. Each major type of text, across topic and genre, should be represented in the corpus in proportion to their usage in the language.

While a corpus can be broadly representative of a language, no collection of text can definitively represent a language. There is no set percentages that can be specified across mode and genre. We cannot ask how much more or less is English written than spoken? Instead, a corpus is better defined by its composition. In this paper, we compare two corpora – the LDC's Gigaword Corpus of newspaper text and our own Web Corpus.

Proceedings of the Australasian Language Technology Workshop 2005, pages 167–175,
Sydney, Australia, December 2005.

2.1 Existing Corpora

One of the first machine-readable corpora was the Brown Corpus (Francis and Kucera, 1979), created in 1964 and consisting of 1 million words of American English. Another step forward in corpus development came with the Penn Treebank, which consists of 4.5 million words of American English manually annotated with part of speech (POS) tags and parse trees (Marcus et al., 1994). The British National Corpus (BNC) is a collection of British English, consisting of 90 million words of written text and 10 million words of transcribed speech (Burnard, 2000). At almost one hundred times the size of the Brown Corpus and more than twenty times the size of the Penn Treebank, it is too large to be manually annotated and so the BNC is automatically tagged with POS tags.

While languages such as English are rich with language resources, minority languages often resort to using freely available web text. One of the first web-collected corpora was the Hungarian Web Corpus (Halacsy et al., 2004), created by downloading pages from the `.hu` domain. It has about 1 billion words of text after removal of duplicates and non-Hungarian text.

2.2 The Gigaword Corpus

The *English Gigaword Corpus* consists of over 4 million documents and 1.75 billion words (Graff, 2003), with more than 2 billion tokens when the text is tokenised, including punctuation. It is the next progression up in size from the BNC. The Gigaword is the large single collection of English news text available to-date. It consists of newspaper text from the Associated Press, Agence France Press (English Service), the New York Times Newswire, and Xinhua News Agency (English Service) from the years 1994-2001. Parts of Gigaword have been released by the LDC in other collections. The data is skewed toward the New York Times ($\sim 50\%$) and the Associated Press ($\sim 25\%$). The Agence France Press and Xinhua News Agency articles together make up the last 25%.

3 The Web Corpus

We collected the Web Corpus by a controlled traversal of the web. If a web spider were to traverse the web starting from a single seed URL, many more pages of the seed URL topics would be visited than other topics. As some topics on the web are linked to by a larger number of pages than others, these topics also tend to be over-represented in such a sample of the web. Pages pertaining to these topics tend to have more incoming links than others, but this is not entirely reflective of the popularity of such websites. Gambling and adult websites, for example, are known to densely link to one another.

3.1 Uniform Web Sampling

The web is too large to be downloaded entirely or for a significant percentage to be collected by most research projects. Two primary approaches exist for obtaining a uniform sample of the web. IP address sampling techniques (Lawrence and Giles, 1999; O'Neill et al., 1997) obtain a uniform sample by randomly generating addresses and exploring the associated server. While the IP address sampling approach has been successfully implemented and used for extraction statistics of the web, it is costly in the resources required. Lawrence and Giles report that only 1 in 269 tries of a random IP address received a response.

Random walk techniques (e.g. Henzinger et al., 2000) attempt to create a regular undirected web graph on which a random traversal would produce a uniform sample. This is usually accomplished using search engines to calculate the number of backward links (making the web undirected) and creating self-loops to standardise the number of links (both incoming and outgoing) for each page.

3.2 USyd-NLP-Spider

Our *Web Corpus* is compiled from the web using a method based on link-to-link traversal, similar to the random walk approaches. It allows faster download of web pages than the IP sampling technique but does not produce a uniform sample. Web pages are collected by the *USyd-NLP-Spider*, a multi-thread spider written in Python. We seeded the spider with links from the *Open Directory*[2]. The broad topic coverage of this open source classification tree allows us to create a topic-diverse collection of web text. However, certain topics in the directory have more links than others (not reflective of its coverage on the web) and topics of similar generality are placed at different depths. The Open Directory is flattened using a rule-based algorithm to reduce the topic skew. A list of 358 general topics and associated URLs is created.

From these seed URLs, the spider performs a breadth-first search. For each link, the spi-

[2]The Open Directory Project, `http://www.dmoz.org`

der samples pages from the same section of the website until a minimum word quota has been reached. External links are extracted and added to the link collection of the parent topic.

4 Text Cleaning

The HTML collected by the USyd-NLP-Spider must be transformed into a format usable by NLP algorithms – whitespace delimited tokens, organised into sentences, one per line. We call this process *text cleaning*. Text cleaning consists of many low-level processes, beginning with interpreting character encoding on HTML pages and transforming them into ISO Latin-1, followed by sentence boundary identification, tokenisation, and text filtering.

Our sentence boundary identification component is based on Ratnaparkhi (1998). We adapted his model for regular English text by adding additional features for HTML tags. Our tokeniser is based on the one used for the Penn Treebank (MacIntyre, 1995), modified to correctly tokenise URLs, email addresses, and other web-specific text.

The filtering component is especially important for cleaning web text. Not all parts of web pages consists of grammatical sentences; they may contain an ingredient list for a cooking recipe or fragment of C++ code. Our rule-based filter removes non-content words and foreign language text. It removes sentences and documents with a low percentage of dictionary words.

5 Token Types

We are interested in the type of tokens in each corpus. For example, are there more numbers on the web than in newspaper text? From each corpus, we randomly select a 1 billion word sample and classified the tokens into seven disjoint:

Numeric – At least one digit and zero or more punctuation characters, e.g. 2, 3.14, $5.50
Uppercase – Only uppercase, e.g. REUTERS
Title Case – An uppercase letter followed by one or more lowercase letters, e.g. Dilbert
Lowercase – Only lowercase, e.g. violin
Alphanumeric – At least one alphabetic and one digit (allowing for other characters), e.g. B2B, mp3, RedHat-9
Hyphenated Word – Alphabetic characters and hyphens, e.g. serb-dominated, vis-a-vis
Other – Any other tokens

Finally, we also measure the number of dictionary words using the Unix words file.

	Gigaword	Web Corpus
Numeric	1.8%	1.2%
Uppercase	1.4%	2.2%
Title Case	14.2%	14.4%
Lowercase	68.4%	68.7%
Alphanumeric	0.3%	0.2%
Hyphenated	0.9%	0.7%
Other	13.0%	12.6%
Dictionary Words	69.6%	66.9%

Table 1: Tokens for each corpus

	Gigaword		Web Corpus	
Tokens	1 billion		1 billion	
Token Types	2.2 million		4.8 million	
Numeric	343k	15.6%	374k	7.7%
Uppercase	95k	4.3%	241k	5.0%
Title Case	645k	29.3%	946k	19.6%
Lowercase	263k	12.0%	734k	15.2%
Alphanumeric	165k	7.6%	417k	8.6%
Hyphenated	533k	24.3%	970k	20.1%
Other	150k	6.8%	1,146k	23.7%
Dict. Words	43k	2.0%	45k	0.9%
% of Dict.	94.3%		98.0%	
45,427 words	42,835 words		44,539 words	

Table 2: Token types for each corpus

5.1 Token Classification

At the macroscopic level, the two corpora appear similar. Table 1 shows the percentage *by token* in each corpora across the seven categories. The results are very close, with the only significant difference being the 2.7% drop for dictionary words in the Web Corpus relative to the Gigaword. However, an analysis by *token type* shows big differences between the two corpora (see Table 2). The same size samples of the Gigaword and the Web Corpus have very different number of token types. While only 2.2 million token types are found in the 1 billion word sample of the Gigaword, about twice as many token types (4.8 million) are found in an equivalent sample of the Web Corpus.

An analysis of the token types show similar percentages in four of the seven categories: uppercase, lowercase, alphanumeric, and hyphenated tokens. Although the Web Corpus has about twice the number of *token types*, it has similar number of numeric token types as the Gigaword. The percentage of numeric token types in the Gigaword is more than twice that of the Web Corpus. The Web Corpus has a lower percentage of title case tokens, at 19.6%, than the Gigaword at 29.3%.

	Unique to Gigaword		Unique to Web Corpus	
All	1,413,427		4,048,531	
Numeric	282k	19.9%	313k	7.7%
Uppercase	36k	2.5%	182k	4.5%
Title Case	351k	24.8%	654k	16.2%
Lower Case	100k	7.1%	571k	14.1%
Alphanumeric	138k	9.8%	389k	9.6%
Hyphenated	395k	28.0%	832k	20.5%
Other	111k	7.9%	1,107k	27.3%
Dict. Words	0k	0.0%	2k	0.0%

Table 3: Token types unique to each corpus

A large percentage difference is also observed in the number of dictionary words. These percentages don't give the whole picture, as the Unix dictionary has only 45,427 words. Both corpora contain a high percentage of the words in the Unix dictionary, at 98.0% for the Web Corpus and 94.3% for the Gigaword.

The percentages of token types within a corpus is also very informative. While only 0.9% of the Web Corpus vocabulary is dictionary words, it accounts for 66.9% of the actual tokens. In the Gigaword, the dictionary words account for 2.0% of the token types but 69.6% of the token instances. About 734,000 (15.2%) of Web Corpus token types are lowercase, most of which are not found in the dictionary. Another 946,000 (19.6%) of Web Corpus token types are title case, which includes named entities. In the Web Corpus, and similarly in the Gigaword, the non-dictionary words are a large percentage of the token types but a relatively small percentage of the actual tokens.

5.2 Unique Token Types

To better account for the difference between the 2.2 million token types in the Gigaword compared with the 4.8 million token types in Web Corpus, we extracted the terms found in one corpus but not the other. Table 3 shows the percentage of token types unique to each corpus (i.e. found in the Gigaword but not in the Web Corpus, or vice-versa). Virtually no dictionary words are unique to each corpus, as both corpora already contain most of the words in the Unix dictionary.

Four significant categories are numeric, title case, hyphenated, and other tokens. They explain some of the difference between the vocabulary of the two corpora. Numeric tokens tend to be unique to texts; for example, the number 1,349,343 is unlikely to appear again in a different context. Title case tokens contains many named entities, which tend to be context specific. Hyphenated tokens behave more like bigrams as they are the combination of two unigrams. Other than the conventional hyphenated words (e.g. ice-cream), these bigram-like words tend to be more sparse. The above results suggest that the token types unique to Gigaword tend to be numbers and named-entities, whereas token types unique to the Web Corpus are non-standard words (e.g. email addresses and URLs).

5.3 Misspellings

A possible explanation for the significant difference between the number of token types is the misspelling of words. The web contains documents written by people with a widely varying command of English. Their work is not checked by professional editors unlike the newspaper text. Thus we expect that there are many more ungrammatical sentences and misspellings in the Web Corpus than the Gigaword. The misspellings in the Web Corpus are new "words" that contribute to the relatively higher token type count than the Gigaword.

To determine the degree that misspellings contribute to the number token types in the Web Corpus, we examined letter combinations that are one character away from the correct spelling. For a target word, we generate the letter combinations that are one operation from the correct spelling. Four operations are considered:

Insertion – A new letter is inserted into the correct word (not before the first letter)

Deletion – One letter in the correct word (except the first) is deleted

Substitution – One letter in the correct word (except the first) is substituted by another letter in the alphabet

Letter Reordering – One letter in the correct word (except the first) is swapped with the next letter

The only letter preserved in all of the above transformations is the first, as very few misspelling replaces the first letter of the word. Any combination found in a dictionary is also discounted, so that the correct word is not transformed into another valid word (e.g. difference to differences). Figure 4 shows the misspellings

Web Corpus		Gigaword
differeince	disfference	differencre
differrence	differience	differencce
differece	differenced	differnce
differenece	diffeerence	diffference
dfference	differenc	diference
differnce	differnence	diffrence
diffference	differennce	diffderence
diference	diffeence	differencel
diffrence		
3.7 matches per word		1.7 matches per word

Table 4: Misspelling of difference in Web Corpus and Gigaword

	Gigaword Rank	Web Corpus Rank	Diff. Rank
Kafelnikov	7,078	733,477	14
Vicario	9,658	613,056	19
Ivanisevic	7,147	569,627	23
Seles	5,285	179,175	77
McCurry	5,631	147,544	111
Walesa	7,287	146,494	112
Ciller	7,537	1,125,901	9
Serb-held	4,343	569,627	21
Muslim-Croat	8,791	381,462	32
SARAJEVO	9,556	300,220	38

Table 5: Selected words with Gigaword rank much higher than Web Corpus

of the word difference found in the Web Corpus and the Gigaword. While there are 17 misspellings of difference that are one transformation from the correct spelling in the Web Corpus, there are only 8 such misspellings in Gigaword. For all words found in the Unix dictionary, we calculated the average number of misspellings found in each of the two corpora. The Web Corpus has more than twice the number of misspellings than the Gigaword, 3.7 per word compared to 1.7 for the latter. Misspellings are another cause of the higher token type count for the Web Corpus.

6 Topical Words

Some topical differences between two corpora can be identified by finding words frequent in one corpora but not the other, and vice-versa. From each corpus we extract the 10,000 most frequent words and find the words with the biggest difference in rank between the corpora. This process highlights the differences between the two corpora, showing the words and topics with high coverage in one but little or no coverage in the other.

6.1 Frequent Gigaword Words

Table 5 shows examples of the top 10,000 ranked words in the Gigaword with the biggest difference with the Web Corpus rank. The words shown in the figure were selected to illustrate certain points and they are not indicative of all the words with a large difference in rank. The words can be divided into three groups:

The words in the first group, Kafelnikov, Vicario, Ivanisevic, and Seles, reflect the years covered by documents in the Gigaword. As the Gigaword contains newspaper articles from the years 1994-2001, these terms correspond to names of active professional tennis players of the time. This included Yevgeny Kafelnikov (active 1995-2004), Arantxa Sánchez Vicario (active 1989-2002), and Goran Ivanisevic (active 1988-2001). The Web Corpus on the other hand contains mostly texts from late 1990's onward, with a significant proportion written in the past few years. As these tennis players were no longer active (or no longer making the headlines) at the time that many Web Corpus documents were written, their names were not frequent terms in the Web Corpus.

The next two groups also reflect the news covered by the Gigaword articles. McCurry, Walesa, and Ciller are names of political figures during early and mid-1990's. Mike McCurry was the press secretary of U.S. President Bill Clinton from 1994-98, Lech Walesa was the Polish President from 1990-95, and Tansu Ciller was the Turkish Prime Minister from 1993-96.

The terms Serb-held, Muslim-Croat, and SARAJEVO in the third group are terms from newspaper articles about the Yugoslav War (a series of conflicts from 1991-2001). Possible phrases include Serb-held territories and Muslim-Croat army and SARAJEVO as the locational identifier at the start of an article.

6.2 Frequent Web Corpus Words

The terms dvd, MySQL, and mp3 were not found in the Gigaword. The all lowercase formatting of dvd and mp3 is likely the reason they were not found. While both were invented in the mid-1990's, they would probably always appear capitalised in newspapers text as DVD and MP3. MySQL, released in 1995, does not appear in the 1 billion word Gigaword sample.

Some web-oriented words with much higher ranks in the Web Corpus include unsubscribe and emailed. As the Internet only began to

	Web Corpus Rank	Gigaword Rank	Diff. Rank
dvd	6,546	Not found	16
MySQL	6,948	Not found	23
mp3	9,092	Not found	30
unsubscribe	8,932	753,428	47
emailed	8,102	641,461	52
pissing	8,337	351,980	63
pee	8,946	119,101	157

Table 6: Selected words with Web Corpus rank much higher than Gigaword

gain prominence only during the second half of the Gigaword timeline, such terms rarely appeared in that corpus. Many instances of the term unsubscribe may also have not been properly filtered out from the Web Corpus with non-content terms such as Click here to unsubscribe. This increased word rank of unsubscribe is an artifact of the text cleaning process of the Web Corpus.

Slang and expletives also have much lower usage in newspaper text. The terms pissing and pee, slang words for urinate, appear relatively more frequently in web text than in newspaper text. As newspaper text is carefully edited, use of expletives is restricted, and the use of slang and other colloquialisms is discouraged.

7 Thesaurus Extraction

Thesauri are useful in many NLP and Information Retrieval (IR) applications. They expand the recall and coverage of the system by providing synonyms of a target word. In NLP, for example, this expansion technique is helpful when n-gram counts for a target word are unreliable. In IR, synonyms help expand keyword queries into many related queries, boosting the recall rate of the system. While thesauri are traditionally manually collected, automatic thesaurus extraction is superior to manual construction in several aspects (Curran, 2004). Manual thesaurus construction is labour-intensive and time consuming, and the result suffers from bias, low coverage, and inconsistency. Bias and inconsistency of lexical resources can be seen in WordNet, in which similar categories of words have different degrees of distinction. As such lexical resources are constructed by human experts, their personal biases are also reflected in the final product. We extract thesauri from the Gigaword and the Web Corpus for the same set of headwords to see the differences in word usage and word similarity in each corpus.

7.1 Method

We used the thesaurus extraction system developed by Curran (2004). It is based on the *distributional hypothesis* that *similar words appear in similar contexts*. The system extracts one-word noun synonyms (i.e. not multi-word expressions). The extraction process is divided into two main parts. First, all target noun contexts are represented as relations and compiled into one context vector for each noun. Second, a comparison between all context vectors is made to identify the closest (i.e. most similar) terms.

Contexts are extracted from raw sentences using a maximum entropy POS tagger, chunker, and a relation extractor (Curran and Clark, 2003). Six different types of relationship are identified:

- Between a noun and a modifying adjective.
- Between a noun and a noun modifier.
- Between a verb and a subject.
- Between a verb and a direct object.
- Between a verb and an indirect object.
- Between a noun and the head of a modifying prepositional phrase.

The nouns in each case (including the subjects and objects) are the target headword. All context relations for a particular headword are aggregated into the headword's context vector. Words are identified as synonyms on the basis of the number of context vectors they have in common.

7.2 Evaluation

Curran evaluates against a combination of four gold standard thesauri: Macquarie (Bernard, 1990), Roget's (Roget, 1911), Moby (Ward, 1996), and Oxford (Hanks, 2000). The gold standard synonyms of a headword are aggregated into one unranked list. The *inverse rank* (InvR) evaluation metric takes the rankings within the extracted list into account. For example, if the extracted terms at ranks 3, 5, and 28 are found in the gold standard list, then $\text{InvR} = \frac{1}{3} + \frac{1}{5} + \frac{1}{28} \cong 0.569$.

200 synonyms are extracted for 300 headwords from 2 billion words of the Web Corpus and from 2 billion words of the Gigaword. The headwords are test nouns created to cover interesting properties – including across frequency bands of several corpora (Curran, 2004).

Corpus	InvR	InvR max
Gigaword	1.86	5.92
Web Corpus	1.81	5.92

Table 7: Average InvR for 300 headwords

	Word	InvR Scores			Diff.
1	picture	3.322	to	0.568	2.754
2	star	2.380	to	0.119	2.261
3	program	3.218	to	1.184	2.034
4	aristocrat	2.056	to	0.031	2.025
5	box	3.194	to	1.265	1.929
6	cent	2.389	to	0.503	1.886
7	home	2.306	to	0.523	1.783
8	newspaper	3.036	to	1.381	1.655
9	statement	3.199	to	1.629	1.570
10	firm	2.347	to	0.829	1.518

Table 8: Headwords with biggest InvR difference, Gigaword > Web Corpus

7.3 Results

Table 7 shows the average InvR scores for the Gigaword and the Web Corpus for the 300 headwords. While the overall performance of the two corpora are very similar, on a per word basis one corpus can significantly outperform the other.

7.4 Gigaword Higher InvR Score

Table 8 shows the top 10 terms which the Gigaword InvR results were better than Web Corpus. For the headword home, much better synonyms were extracted from the Gigaword. Table 9 shows the top 50 extracted terms from both corpora. A similar number of matches were made with the gold standard list, with 24 matches for Gigaword to 18 for the Web Corpus. However, the matches were among the top terms in Gigaword but not in the Web Corpus. The top two terms house and apartment were extracted from the Gigaword, but the terms such as page and loan were extracted from the Web Corpus. Collocations, such as home page, were incorrectly extracted instead of synonyms.

7.5 Web Corpus Higher InvR Score

Table 10 shows the top 10 terms which the Web Corpus InvR results were better than Gigaword. The Web Corpus outperformed Gigaword in extracting synonyms for terms such as chain. Table 11 shows the top 50 extracted terms from both corpora. 53 gold standard synoyms were extracted out of the Web Corpus compared to only 9 for the Gigaword. This difference in performance can be attributed to the topic skew

Gigaword (24 matches out of 200)
house apartment building run office resident **residence headquarters** victory **native place mansion** room trip mile **family** night hometown town win neighborhood life suburb school restaurant hotel store city street season area road homer day car shop **hospital** friend game farm facility center north child land weekend community loss **return** hour ...
Web Corpus (18 matches out of 200)
page loan contact **house** us owner search finance mortgage office map links building faq equity news center estate privacy community info business car **site** web improvement extention heating rate directory room **apartment family** service rental credit shop life city school **property place location** job online vacation store facility library free ...

Table 9: Synonyms for home

	Word	InvR Scores			Diff.
1	chain	3.139	to	0.224	2.915
2	walk	3.184	to	0.774	2.410
3	point	3.540	to	1.477	2.063
4	bloke	2.445	to	0.425	2.020
5	game	2.799	to	1.097	1.702
6	graph	2.400	to	0.714	1.686
7	reinforcement	1.808	to	0.244	1.564
8	announcement	1.993	to	0.495	1.498
9	sport	3.116	to	1.642	1.474
10	solicitor	1.634	to	0.161	1.473

Table 10: Headwords with biggest InvR difference, Web Corpus > Gigaword

of the Gigaword and the gold standards. The terms extracted by Gigaword belong to only one sense of the word chain, as in chain stores. The gold standard terms included a more physical sense of chain, such as necklace chain.

A bias is apparent in the topic coverage of both Gigaword and the gold standard. Gigaword is skewed toward the business sense of chains, reflecting financial text that is a significant portion of newspaper articles. The gold standard is skewed toward other senses. The wide topic coverage of Web Corpus becomes apparent in this example. While the top extracted Web Corpus terms also corresponds to the physical sense of chains (e.g. necklace, bracelet, and pendant), terms were also extracted belonging to the business sense of the word (e.g. retailer). Synonyms of chain extracted from the Web Corpus have a much better coverage of the different senses of the word than Gigaword or the gold standard thesauri alone.

Gigaword (9 matches out of 200)
store retailer supermarket restaurant outlet operator shop shelf owner grocery **company** hotel manufacturer retail franchise clerk maker discount business sale superstore brand clothing food giant shopping **firm** retailing industry drugstore distributor supplier bar insurer inc. **conglomerate** network unit apparel boutique mall electronics carrier division brokerage toy producer pharmacy airline inc ...
Web Corpus (53 matches out of 200)
necklace supply **bracelet pendant rope belt ring earring** gold bead silver **pin wire cord** reaction clasp jewelry **charm** frame **bangle strap** sterling loop timing plate metal **collar** turn **hook** arm length **string** retailer repair **strand** plug diamond wheel industry tube surface neck **brooch** store **molecule ribbon** pump **choker** shaft body ...

Table 11: Synonyms for chain

Gigaword (13 out of 200)
acushnet zoolander working-class marshak interchangeability scouse ghyll dubliner **fella** film guy yorkshireman aussie bostonite irishman **lad** bumbler **chap** scrum-half texan ex-marine profane kansan medavoy **gentleman guy** ballplayer Irishman anybody lunk **somebody** up-and-down vaudevillian yorker theatricality englishman **person** hobby newspaperman klutz goof everyman chicagoan scotsman artilleryman brazilian **fellow** midwesterner ref ballclub ...
Web Corpus (16 matches out of 200)
lad fella somebody bondsman endomorphism **gentleman** aussie **dude** boucher **guy** englishman **chap** stranger balfour iraqi youngster nobody policeman cop passer-by everybody waitress boyfriend anybody no-one **punter** mum irishman lowepro teenager businessman bartender girlfriend fiance buffy neighbour 40ml hippie **bastard** beggar sandstorm kiwi foreigner grandma frenchman dad yank pooch brit spectator ...

Table 12: Synonyms for bloke

Web Corpus also significantly outperformed Gigaword for the term bloke (see Table 12). Bloke, British and Australian slang for a man, has a much higher InvR score on the Web Corpus list than the Gigaword list. This reflects the international nature of the web, where terms specific to British and Australian English were found often enough to be reliably characterised by their context. Documents included in Gigaword have a skew towards American English, with the New York Times contributing the majority of text in that corpora. Without many training examples in non-American English, it is difficult to correctly extract synonyms for words such as bloke.

7.6 Discussion

While the Gigaword and Web Corpus have similar overall averages in the InvR scores, there are significant differences in performance for different terms. The Gigaword consists of newspaper text and better synonyms are extracted for topics covered in the news. The Web Corpus is more international and more topic-diverse, successfully extracting synonyms in different varieties of English and for different senses of words. However the dominance of certain topics on the web, with web-specific vocabulary, means that sometimes a highly biased thesauri is extracted.

To create a better Web Corpus, not only is there a need to cover a wide ranging number of topics, but one must actively prevent specific topics from dominating the corpus. A more balanced corpus can be created with better spidering strategies. For example, the spider could be designed to automatically identify the topics of the websites visited.

8 Conclusion

The web is a promising source for creating large corpora for Natural Language Processing. In this paper, we compared our Web Corpus to the traditional Gigaword Corpus and demonstrated that the Web Corpus is useful for the task of automatic thesaurus creation.

Words and word usage differ in corpora, especially when they are compiled from different sources and medium. We examined the words and word usage in the Gigaword Corpus as compared with the Web Corpus. We have shown some of the differences in topics covered by the two corpora, as well as vocabulary variants and errors. Some of these contrasts can be attributed to the genre of text, but some are artifacts of the corpus creation process.

Our results in thesaurus extraction showed that the web text obtained similar overall results to a corpus of newspaper text. The alternative topical and lingustic information suggests that web-collected corpora is a viable addition or even alternative to traditional corpora of newspaper and other printed text.

As the Web Corpus is significantly larger than most corpora of printed text, better results can be obtained by training algorithms on the Web Corpus. This is especially true of tasks that suffer from the data sparseness problem. With much more text available for download on the web, the limits of the Web Corpus in size have yet to be reached.

Acknowledgements

We like to thank our anonymous reviewers and the Language Technology Research Group at the University of Sydney for their comments and feedback. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

John R.L. Bernard, editor. 1990. *The Macquarie Encyclopedic Thesaurus*. The Macquarie Library, Sydney, Australia.

Lou Burnard, editor. 2000. *Reference Guide British National Corpus (World Edition)*. British National Corpus Consortium.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 91–98, Budapest, Hungary, 12–17 April.

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

W. Nelson Francis and Henry Kucera. 1979. Manual of Information to accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers. Technical report, Brown University, Providence, RI USA.

David Graff. 2003. English Gigaword. Technical Report LDC2003T05, Linguistic Data Consortium, Philadelphia, PA USA.

Peter Halacsy, Andras Kornai, Laszlo Nemeth, Andras Rung, Istvan Szakadat, and Vikto Tron. 2004. Creating open language resources for Hungarian. In *Proceedings of the Language Resources and Evaluation (LREC)*, pages 203–210, Lisbon, Portugal.

Patrick Hanks, editor. 2000. *The New Oxford Thesaurus of English*. Oxford University Press, Oxford, UK.

M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. 2000. On Near-Uniform URL Sampling. In *Proceedings of the 9th International World Wide Web Conference*.

Steve Lawrence and C. Lee Giles. 1999. Accessibility of information on the web. *Nature*, 400:107–109, 8 July.

Robert MacIntyre. 1995. Sed script to produce Penn Treebank tokenization on arbitrary raw text. http://www.cis.upenn.edu/~treebank/tokenizer.sed.

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.

Edward T. O'Neill, Patrick D. McClain, and Brain F. Lavoie. 1997. A Methodology for Sampling the World Wide Web. *Annual Review of Online Computer Library Center (OCLC) Research*.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA USA.

Peter Mark Roget. 1911. *Thesaurus of English words and phrases*. Longmans, Green and Company, London, UK. Available from http://promo.net/pg/.

Grady Ward. 1996. *Moby Thesaurus*. Moby Lexicon Project. Available from http://etext.icewire.com/moby/.

Automatic induction of a POS tagset for Italian

R. Bernardi
KRDB,
Free University of Bolzano Bozen,
P.zza Domenicani, 3
39100 Bolzano Bozen, Italy,
bernardi@inf.unibz.it

A. Bolognesi, C. Seidenari and **F. Tamburini**
CILTA,
University of Bologna,
P.zza San Giovanni in Monte, 4,
I-40124, Bologna, Italy,
{bolognesi,seidenari,tamburini}@cilta.unibo.it

Abstract

In this paper we present work in progress on the PoS annotation of an Italian Corpus (CORIS) developed at CILTA (University of Bologna). We aim to automatically induce the PoS tagset by analysing the distributional behaviour of Italian words by relying only on theory-neutral linguistic knowledge. To this end, we propose an algorithm that derives a possible tagset to be further interpreted and defined by the linguist. The algorithm extracts information from loosely labelled dependency structures that encode only basic and broadly accepted syntactic relations, namely Head/Dependent, and the distinction of dependents into Argument vs. Adjunct.

1 Introduction

The work presented in this paper is part of a project aiming to annotate CORIS/CODIS (Rossini Favretti et al., 2002), a 100-million-word synchronic corpus of contemporary written Italian, with part-of-speech (PoS) tags.

Italian is one of the languages for which a set of annotation guidelines has been developed in the context of the EAGLES project (Monachini, 1995). Several research groups have worked on PoS annotation in practice (for example, Torino University, Xerox and Venice University), but comparing the tag sets used by these groups with Monachini's guidelines reveals that though there is a general agreement on the main parts of speech to be used[1], considerable divergence exists when it comes to the actual classification of Italian words with respect to these main PoS classes. The classes for which differences of opinion are most evident are adjectives, determiners and adverbs. For instance, words like *molti* (many) have been classified as "indefinite determiners" by Monachini, "plural quantifiers" by Xerox, "indefinite adjectives" by the Venice and Turin groups. It is not simply a matter of different terminological options resolvable by a mere one-to-one relabelling, nor a matter of simply mapping different classes into a greater one. Crossings between tagsets are complex mostly because of the different theoretical points of view used in categorizing words. For instance, the single tag DET "determiner" in the Xerox tagset matches with DIM "demonstrative adjective" or ART "article" in the Venice group (and with DET "determiner" or ART "article" in Monachini) whereas, viceversa, the single tag DEIT "deictic pronoun" by the Venice group matches alternatively with DEM "demonstrative" or PRON "personal pronoun" in Xerox.

These simple examples show that the choice of PoS tag is already influenced by the underlying linguistic theory adopted. This theoretical bias will then influence the kind of conclusions one can draw from the annotated corpus.

Our aim is to automatically derive an empirically founded PoS classification making no *a priori* assumptions about the PoS classes to be distinguished.

Early approaches to this problem were based on the hypothesis that if two words are syntactically and semantically different, they will appear in different contexts. There are a number of studies based on this hypothesis in the fields of both computational linguistics and cognitive science aiming at building automatic or semi-automatic procedures for clustering words (Brill and Marcus, 1992; Pereira et al., 1993; Schütze, 1993; Clark, 2000; Redington et al., 1998).These papers examine the distributional behaviour of some target words by comparing the lexical distribution of their respective collocates and by using quantitative measures of distributional similarity.

The main drawback of these techniques is the

[1]The standard classification consists of nouns, verbs, adjectives, determiners, articles, adverbs, prepositions, conjunctions, numerals, interjections, punctuation and a class of residual items which differs from project to project.

Proceedings of the Australasian Language Technology Workshop 2005, pages 176–183,
Sydney, Australia, December 2005.

limited context of analysis. Information is collected from a restricted context, of for instance 3 words, which can conceal syntactic dependencies longer than the context interval.

Our approach to solve this problem is to use basic syntactic relations together with distributional and morphological information. The system we have developed consists of three phases: (1) a first basic distinction of word classes is induced by means of Brill's algorithm (Brill and Marcus, 1992); (2) in the second phase, this distinction is further specified by means of minimal syntactic information; and (3) in the third phase, the ultimate PoS tagset is obtained by using distributional and morphological knowledge. Little, if any, language-specific knowledge is used, hence the method is in principle applicable to any language.

A large number of localized syntactic descriptions per word are exploited to identify differences in the syntactic behaviour of words. Associating rich descriptions to lexical items, our approach is, to some extent, related to supertags (Bangalore and Joshi, 1999).

The outcome is a *hierarchy* of PoS tags that is expected to help annotators and enhance the search interface of the annotated corpus.

Section 2 gives an outline of our work; Section 3 describes in details the algorithm; Section 4 analyses the results of the work, listing the PoS tags obtained with this method; section 5 briefly outlines further work.

2 Proposal

The present paper focuses on the second phase of the system describing how syntactic information can be exploited to induce the PoS tagset. It builds on the results obtained in (Tamburini et al., 2002) where it is shown that Brill's algorithm identifies three main word classes, namely noun (N), verbs (V) and all the others (X).

In this article we will focus on the X class, describing how this can be further broken down by automatically grouping words that share similar syntactic behaviours. The algorithm uses the tags obtained in the first phase and dependency structures carrying only basic syntactic information about Head/Dependent relations and Argument/Adjunct distinctions among the Dependents.

Starting from these loosely labelled dependency structures, the type resolution algorithm obtains type assignments for each word. The syntactic type assignments obtained encode the different syntactic behaviour exhibited by each word. Examples of the labelled dependency structures and the obtained assignments are given in Figure 2. An information lossless simplification algorithm is used to automatically derive a first tagset approximation (see Section 3).

At the end of the second phase, the X class is divided into 9 PoS tags that are sets of syntactic behaviours. In the third phase, we plan to further divide the classes obtained by means of distributional and morphological information.

3 The Algorithm

The algorithm consists essentially of three components: (i) in the first, each word is assigned the complete set of syntactic types extracted from loosely labelled dependency structures; (ii) in the second, we obtain a first approximation of relevant classes by grouping words that display similar behaviours, and we build their inclusion chart. This is obtained by creating the sets of those words that in (i) showed the same type at least once, and by pairing these sets of words with their shared set of types. In the following sections we will refer to such pairs as Potential PoS (PPoS); (iii) finally, we prune the obtained inclusion chart by highlighting those paths that relate pairs which are significantly similar, where the similarity is measured in terms of frequency of types and words. The pruning results in a forest of trees whose leaves form sets identifying the induced PoS tags.

Figure 1 shows a flow chart which summarizes the three phases of our algorithm.

3.1 Dependency Structures

Our dependency structures are derived from a sub-treebank of TUT, The Turin University Treebank (Bosco et al., 2000; Bosco, 2003). The treebank currently includes 1500 sentences organized in different sub-corpora from which we converted 441 dependency trees, maintaining only the basic syntactic information required for this study. More specifically, we mantained information on Head-Dependent relations by distinguishing each dependent either as an Argument or as an Adjunct.

Moreover, words are marked as N (nouns), V (verbs) or X (all others) according to the results obtained in (Tamburini et al., 2002). We use $< >$ to mark Head-Argument relation and $\ll$ and $\gg$ to mark Head-Adjunct relation where the arrows point to the Head. From

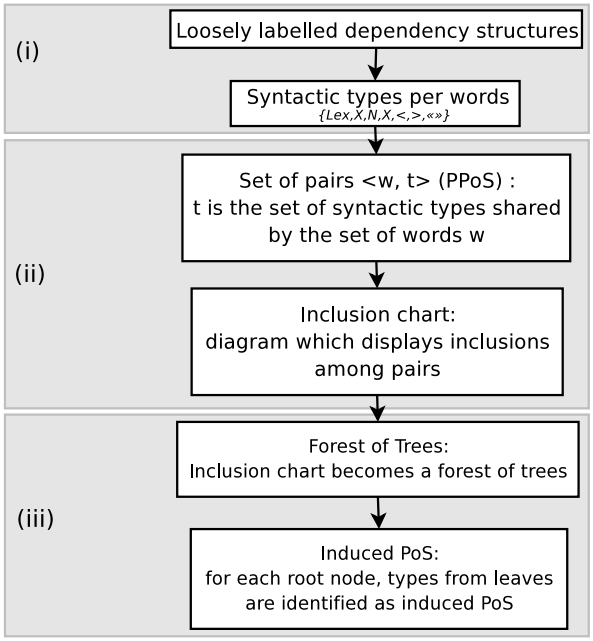


Figure 1: Algorithm Architecture

these dependency structures we extract syntactic type assignments by projecting dependency links onto formulas. Formulas are built out of $\{<, >, \ll, \gg, N, X, V, lex\}$ where the symbol lex stands for the word the formula has been assigned to. The formal details of the type resolution algorithm are provided below.

Type Resolution Let $W = \langle w_1, ..., w_n \rangle$ stand for an ordered sequence of words in a given sentence and let $w_j = \langle orth_j, bl_j, t_j \rangle$ stand for a word in the sentence, where $orth_j$, $bl_j \in \{N, V, X\}$ and t_j represent the orthographic transcription, the basic label and the type of the j-th word respectively. Let $E = \{\langle R, w_i, w_k \rangle\}$ be the set of edges where $R \in \{<, >, \ll, \gg\}$ is ordered by $|k - i|$ in ascending order. Given a dependency structure represented by means of W and E,

$-\forall w_j \in W, \quad t_j = lex$

$-$ foreach $\langle R, w_i, w_j \rangle \in E$

if $R = '>'$ $\quad \langle w_j, bl_j, t_j \rangle \rightsquigarrow \langle w_j, bl_j, bl_i > t_j \rangle$

if $R = '<'$ $\quad \langle w_i, bl_i, t_i \rangle \rightsquigarrow \langle w_i, bl_i, t_i < bl_j \rangle$

if $R = '\ll'$ $\quad \langle w_j, bl_j, t_j \rangle \rightsquigarrow \langle w_j, bl_j, bl_i \ll t_j \rangle$

if $R = '\gg'$ $\quad \langle w_i, bl_i, t_i \rangle \rightsquigarrow \langle w_i, bl_i, t_i \gg bl_j \rangle$

where the operator $\rightsquigarrow$ replaces the first item with the second in W.

For the sake of simplicity in Figure 2 for each word w_j only $orth_j$ and bl_j are displayed.

After applying the type resolution algorithm to all the given dependency structures, a lexicon is built with sets of types assigned to all words except nouns and verbs, which are discarded as

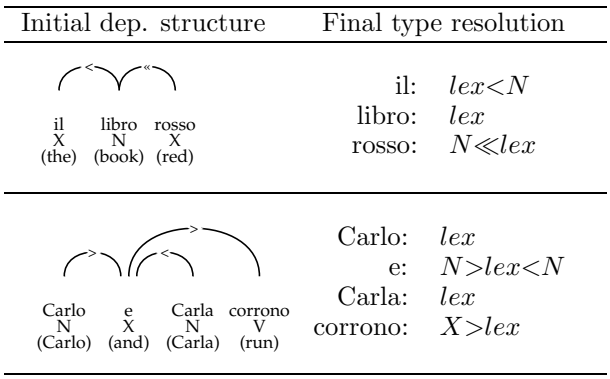


Figure 2: Type resolution example

they are not the subject of the present investigation.

For instance, the lexicon entry for the word "e" (*and*) is as below.

$$\text{e} : \begin{cases} X{>}lex{<}X \\ V{>}lex{<}V \\ N{>}lex{<}N \\ N{\ll}X{>}lex{<}X \\ V{\ll}X{>}lex{<}X \\ N{\ll}V{>}lex{<}V \\ N{>}lex{<}X \\ X{>}lex{<}N \\ X{>}lex{<}X \gg N \end{cases}$$

3.2 Inclusion chart

Lexicon entries are gathered together by connecting words which have received the same types. This results in a set of pairs $\langle W, T \rangle$ comprising a set of words W and their shared set of types T.

A consequence of this is that sets of words are composed of at least two occurrence words. In doing this we are assuming that a set of syntactic types represented by a single word does not have a linguistic significance.

Consider for example the following sample words with the corresponding types:

$$w_1 : \begin{cases} t_1 \\ t_2 \\ t_4 \end{cases} \quad w_2 : \begin{cases} t_1 \\ t_4 \end{cases} \quad w_3 : \begin{cases} t_3 \\ t_5 \end{cases} \quad w_4 : \begin{cases} t_1 \\ t_2 \\ t_3 \end{cases}$$

where $w_1, w_2, ..., w_n$, $n \in \mathbb{N}$ is the lexicon of our example, and t_i, $i \in \mathbb{N}$ stands for types. w_1 is connected both to w_4 and w_2 since they have $\{t_1, t_2\}$ and $\{t_1, t_4\}$ types in common respectively; furthermore, w_4 is connected both to w_2 and w_3 since they have $\{t_1\}$ and $\{t_3\}$ in common, as shown in Figure 3.

From the connection structure built as described above, we obtain the pairs $\langle W, T \rangle$ where W is the set of connected words and T is the

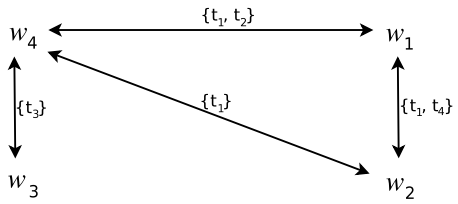


Figure 3: Example of connection structures

set of types carried by the corresponding connection arrow.

For instance, from the example in Figure 3 we obtain the following pairs:

$$\langle \{w_1, w_4\},\ \{t_1, t_2\} \rangle,$$
$$\langle \{w_1, w_2\},\ \{t_1, t_4\} \rangle,$$
$$\langle \{w_1, w_2, w_4\},\ \{t_1\} \rangle,$$
$$\langle \{w_3, w_4\},\ \{t_3\} \rangle$$

We will refer to each pair $\langle W, T \rangle$ as *Potential* PoS (PPoS).

From the given dependency structures we have obtained 215 pairs. They provide us with a first word class approximation with their associated syntactic behaviours.

In order to interpret the classification obtained and to further refine it, we first organize the pairs into an *Inclusion chart* based on subset relations among the PPoS and then we prune it as described below.

Our basic assumption is that type-set inclusions are due to syntactic similarities between words.

Definition 1 (Inclusion Chart) *The nodes of the Inclusion chart are pairs $\langle W, T \rangle$ where W and T are sets of words and sets of types respectively. Given two nodes $n_i = \langle W_i, T_i \rangle$ and $n_j = \langle W_j, T_j \rangle$ of the Inclusion chart, there is an inclusion relation between n_i and n_j, and we write $n_i \sqsubset n_j$, iff $W_i \supset W_j$ and $T_i \subset T_j$. Two nodes n_i, n_j of the Inclusions chart are **connected**, and we write $n_i \rightarrow n_j$, iff $n_i \sqsubset n_j$ and $\neg\exists\ n_k$ such that $n_i \sqsubset n_k$ and $n_k \sqsubset n_j$.*

To illustrate this, let us consider the lexicon entries "e" (*and*), "o" (*or*) and "p_com" (*comma separator*). The set of types assigned to "e" is shown above, those for "o" and "p_com" are as below.

$$\text{o} : \begin{cases} X{>}lex{<}X \\ X{>}lex{<}X{\gg}V \\ N{>}lex{<}N \\ V{>}lex{<}V \\ N{\ll}X{>}lex{<}X \\ N{\ll}N{>}lex{<}N \end{cases} \qquad \text{p\_com} : \begin{cases} X{>}lex{<}X \\ V{>}lex{<}V \\ N{>}lex{<}N \\ N{\ll}X{>}lex{<}X \\ N{>}lex{<}X \\ N{\ll}V{>}lex{<}V \\ N{>}lex{<}X \\ V{>}lex{<}X \end{cases}$$

The set of words

$$W_1 = \{\ \text{p\_com},\ \text{e},\ \text{o}\ \}$$

with the shared set of types

$$T_1 = \{V{>}lex{<}V, X{>}lex{<}X, N{>}lex{<}N, N{\ll}X{>}lex{<}X\}$$

constitute the pair $\langle W_1, T_1 \rangle$.

Once we have obtained the set of all pairs out of the lexicon entries, we build the *Inclusion chart*. Figure 4 shows a portion of this, which contains the pair $\langle W_1, T_1 \rangle$ discussed above.

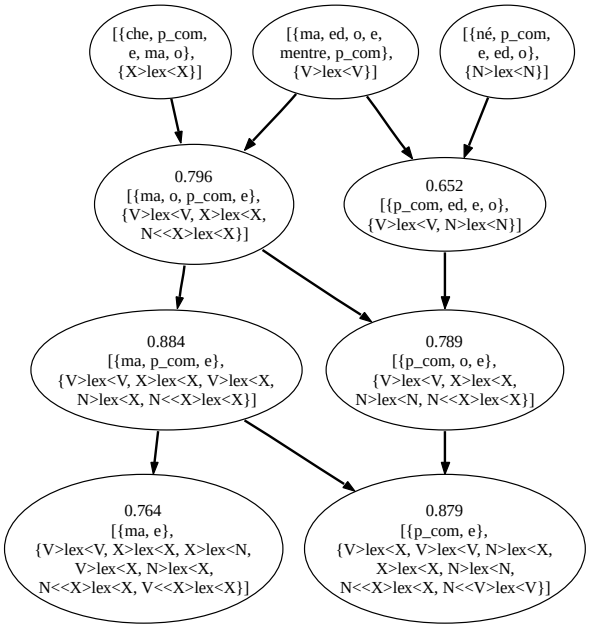


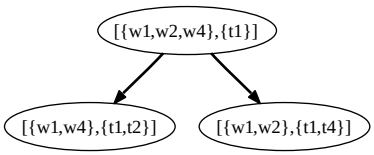
Figure 4: Example of *Inclusion chart*.

Since the *Inclusion chart* obtained displays all possible subset relations between all the pairs, it is rather complex and it conceals the linguistically relevant information we are actually looking for, namely the syntactic similarities between words which lead to their PoS classification.

It is our belief that by identifying the closest connections we can establish the correct PPoS links, i.e. induce a PoS hypothesis.

Consider the example at the beginning of this section, where $P_1 = \langle \{w_1, w_2, w_4\},\ \{t_1\} \rangle$

is included in $P_2 = \langle\{w_1, w_4\},\ \{t_1, t_2\}\rangle$ and $P_3 = \langle\{w_1, w_2\},\ \{t_1, t_4\}\rangle$. This means that both PPoS P_2 and P_3 increase PPoS P_1 by one syntactic type. The following *Inclusion chart* represents the connections between these pairs:



At this point, it is necessary to establish which is the better way to extend P_1, i.e. which of the two syntactic behaviours represented by t_2 and t_4 has to be selected to make the PPoS P_1 closer to a correct PoS.

In order to extract a suitable PoS classification from the *Inclusion chart*, this must be pruned by discarding less relevant nodes; hence, we need to introduce a relevance criterion.

3.3 Forest of Trees

The pruning phase is handled by means of a distance measure between PPoS which helps to highlight the closest pairs.

Before formally defining the distance measure and explaining its role in depth, we present the pruning algorithm.

Pruning Algorithm Let P be the set of all pairs of the *Inclusion chart* and let $e = \langle p_i, p_j, weight_j\rangle$ be an edge, where p_i is connected to p_j and $weight_j$ is a cohesion measure of p_j. For all $p_i \in P$ we indicate with E_{p_i} the set of all edges leaving p_i.

Given P:
$\forall p_i \in P$
  $\forall \langle p_i, p_j, weight_j\rangle \in E_{p_i}$
    if $weight_j$ differs from $\max_j\{weight_j\}$
      then remove $\langle p_i, p_j, weight_j\rangle$ from E_{p_i}

For each pair p_i only the edge connecting it to a pair p_j exhibiting the maximal cohesion measure is maintained.

Figure 5 shows the pruned portion of the *Inclusion chart* given in Figure 4. Notice that each node is weighted apart from the leaf node, because weighting leaves is not necessary for the algorithm proposed. The graph is then transformed into a *Forest of trees*.

We can now move on to explain how linguistically relevant similarities are automatically identified by means of the distance measure. First of all, we need to measure the relevance of a PPoS in terms of how representative its members are with respect to each other.

Definition 2 (Word Frequency)
Let Ω be the set of all words, Ψ the set of all types, and $o : \Omega \times \Psi \to \mathbb{N}$ the function which returns the number of occurrences of word per type. Let $\eta : \Omega \to \mathbb{N}$ be a function which returns the total number of occurrences of a given word.

*We call **word frequency** of $\langle W, T\rangle$ the function $F_{words} : \mathcal{P}(\Omega) \times \mathcal{P}(\Psi) \to \mathbb{N}$ defined as follows:*

$$F_{words}(\langle W, T\rangle) = \frac{1}{|W|} \cdot \sum_{i=1}^{k} \sum_{j=1}^{m} \frac{o(\langle w_i, t_j\rangle)}{\eta(w_i)}$$

where $W = \{w_1, w_2, ..., w_k\}$ is a set of words and $T = \{t_1, t_2, ...t_m\}$ is a set of types.

Definition 3 (Type Frequency)
Let $\xi : \Psi \to \mathbb{N}$ be a function which returns the total number of occurrences of a given type.

*We call **type frequency** of $\langle W, T\rangle$ the function $F_{type} : \mathcal{P}(\Omega) \times \mathcal{P}(\Psi) \to \mathbb{N}$ defined as follows:*

$$F_{types}(\langle W, T\rangle) = \frac{1}{|T|} \cdot \sum_{i=1}^{k} \sum_{j=1}^{m} \frac{o(\langle w_i, t_j\rangle)}{\xi(t_j)}$$

where W and T are as in Definition 2.

Given a pair $\langle W, T\rangle$, we evaluate the internal cohesion of its members as follows. The *word frequency* focuses on the similarity between words in W by rating how far words agree in their syntactic behaviour. Roughly, if the word frequency returns a high value for a pair then we can conclude that words within that pair have a close syntactic resemblance. On the other hand, the *type frequency* rates the similarity between types in T according to the number of times the words to which they have been assigned in the lexicon have shown that syntactic behavior in the dependency structures.

The evaluation of the pair $p_i = \langle W_i, T_i\rangle$ is given by the average of the two cohesion evaluations. We indicate this value by means of the symbol $\mathcal{C}_i$:

$$\mathcal{C}_i = \frac{F_{words}(\langle W_i, T_i\rangle) + F_{types}(\langle W_i, T_i\rangle)}{2}.$$

For each node of the example seen so far Figure 5 displays a weight which measures the cohesion of each node pair.

At first sight, C_1 may appear simplistic, with words and types being equally weighted. However other measures had been tried before C_1

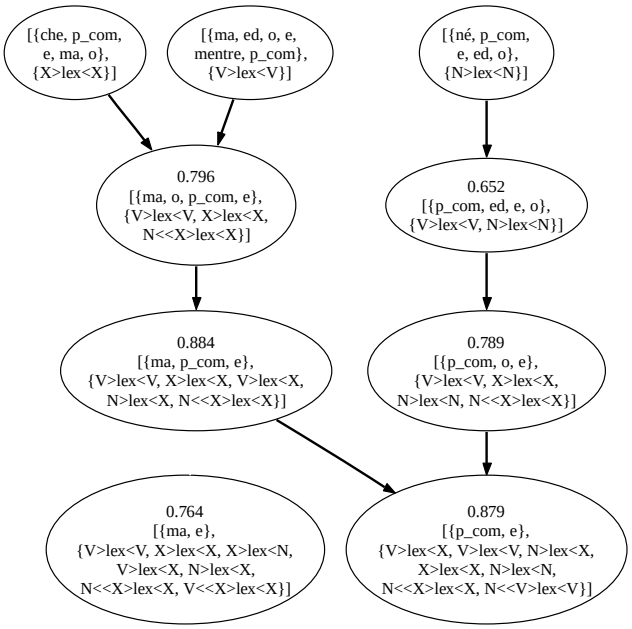


Figure 5: Example of *Forest of trees*.

was decided on, as giving the same importance to a set of words and a set of syntactic behaviours showed itself to be effective.

New kind of measures are currently being carried out. For instance, we are testing how the system works by varying the weight for each edge on the basis of the words added and the frequency with which they demonstrated the syntactic types of the augmented initial PPoS.

3.4 Induced PoS

Each tree in the Forest marks off complex groups of syntactic types. However, the same types occur in more than one tree, therefore we need to identify all and only those belonging to a given tree.

To this end, let us call **leaf nodes**[2] those PPoS with singleton type set not including any other; **root nodes**[3] PPoS not included by any other.

Leaves of each tree are grouped together; such groups constitute the whole type set partition. Clearly each group corresponds to a unique root node.

Syntactic types from leaf nodes encode few specialized syntactic patterns. We assume those patterns to be the syntactic core of a given tree, i.e. the relevant syntactic component of the corresponding PPoS root node.

Once a syntactic core is defined, the corresponding lexical core is automatically derived by identifying word sets showing exclusively sets of types belonging to that syntactic core.

Syntactic core extraction algorithm The following algorithm extracts syntactic cores from root nodes: for all type sets belonging to root nodes we identify the syntactic core as the subset of types obtained by the union of all type sets from the leaves of the corresponding tree. Given R, sets of root nodes:

$\forall \langle W_i, T_i \rangle = p_i \in R$
  $\forall t_k \in T_i$
    $N = \bigcup_j T_j$,where p_j leaf node of p_i tree
    if $t_k \in N$ then
      let $t_k \in T_i$ into the syntactic core

Consider the example proposed in Figure 5, which displays a portion of the Inclusion chart. Here we have the following two PPoS root nodes:

$$\langle \{\text{ma, e}\}, \{V{>}Lex{<}V, X{>}Lex{<}X, X{>}Lex{<}N, V{>}Lex{<}X, N{>}Lex{<}X, V{\ll}X{>}Lex{<}X, N{\ll}X{>}Lex{<}X\} \rangle,$$
$$\langle \{\text{p\_com, e}\}, \{V{>}Lex{<}X, V{>}Lex{<}V, N{>}Lex{<}X, X{>}Lex{<}X, N{>}Lex{<}N, N{\ll}X{>}Lex{<}X, N{\ll}V{>}Lex{<}V\} \rangle$$

The first root node has no leaf, being a root without branches, so it contains no syntactic core. On the other hand, the second has the following three leaves:

$$\langle \{\text{che, p\_com, e, ma, o}\}, \{X{>}Lex{<}X\} \rangle$$
$$\langle \{\text{ma, ed, o, e, mentre, p\_com}\}, \{V{>}Lex{<}V\} \rangle$$
$$\langle \{\text{nep\_apo, p\_com, e, ed, o}\}, \{N{>}Lex{<}N\} \rangle$$

Thus its type set contains the syntactic core

$$\{X{>}Lex{<}X, V{>}Lex{<}V, N{>}Lex{<}N\}$$

In order to associate it with its lexical core a visit to the tree rooted by this node is needed to collect those words $w \in W$ which show only types belonging to the syntactic core, for a given pair $\langle W, T \rangle$.

For example, the word "o" has shown $X{>}Lex{<}X$, $V{>}Lex{<}V$, $N{>}Lex{<}N$, but also $N{\ll}X{>}Lex{<}X$ which belongs to both root nodes so the word "o" cannot be part of the lexical entries the syntactic core is represented by.

The second root node is then associated with the lexical core consisting of { ed, mentre, né, che }. Hence the algorithm

[2] shown at the top of the tree in Figure 5

[3] shown at the bottom of the tree in Figure 5

concludes the existence of the following PoS prototype:

$$\langle \quad \{\text{ed, mentre, né, che}\}, \quad \{X{>}Lex{<}X, V{>}Lex{<}V, N{>}Lex{<}N\} \quad \rangle$$

Notice that this PoS corresponds to the *Coordinators* PoS depicted in Table 1, but here it is simpler because of the simplification of the *Inclusion chart* taken as an example.

The syntactic and lexical core is the output of our algorithm. We assume the core to be the syntactic (and lexical) prototype to be used for PoS classification.

4 Results and Evaluation

The proposed automatic method leads to the subdivision of the first level within the X class (see Section 2) as shown in Table 1.

The sets of automatically extracted syntactic types represent the prototypical syntactic behaviours of the corresponding words summarised by the explanatory PoS labels.

This classification is not fine-grained enough to be used by a tagger to reach an informative and useful annotation and should be intended as a first step through the empirical construction of a hierarchical tagset, e.g. following the parameters for taxonomic classification shown in (Kawata, 2005). Further analysis for each class must be carried out to increase the granularity of the tagset, for instance by exploiting morphological information.

The present study was carried out on a limited quantity of data; the sparseness of primary information we used to derive the proposed tagset might affect the conclusions we have drawn. The results will need to be checked with more data and with different treebanks to avoid biases introduced by the treebank used (TUT) from which the initial dependency structures were extracted.

Despite this, and the fact that further results of the third phase are currently being induced and remain to be investigated, it is promising that the 9 parts of speech induced in this second phase are not in marked contrast with traditional ones nor with widely accepted guidelines, such as (Monachini, 1995).

However, employing dependency structures as described in section 3.1, which means minimal syntactic information, leads to some ambiguities between word classes which may disagree with the linguist's intuitions.

From this point of view, the overlapping of determiners and prepositions within the same PoS is noteworthy. The lack of accuracy this classification results in is due, on the one hand, to the wide range of highly specific syntactic constructions involving determiners and prepositions that share the same loosely labelled dependency structures. Moreover, Italian monosyllabic (or 'proper') prepositions may be morphologically joined with the definite article (for example *di* ('of') + *il* ('the') = *del* ('of the')), performing sintactically both as a preposition and a determiner. Clearly this class will be further specialized by exploiting morphological information.

Polysyllabic (or 'not proper') prepositions, as opposed to monosyllabic ones, tend to occur in a lower number of syntactic patterns and, more crucially, cannot be fused with the article. In this case our system performs more accurately as it is able to correctly detect the syntactic similarities between such prepositions. As they typically tend to carry the function of the head (together with prepositional locutions) in verb-modifying structures they have been classified as 'Verb-Modifing Prepositionals' as shown in Table 1.

The 4 word classes grouping words commonly classified as adjectives and conjunctions may be considered an interesting result of the syntactically motivated induction algorithm presented here. As for adjectives[4] they have been divided into 2 separate classes depending on predicative or attributive distribution with respect to the noun they modify ('Left/Right Adjectivals' in Table 1). As far as conjunctions (and conjunctional locutions) are concerned, again, their syntactic patterning enforced a very clear split between 'Coordinators' and 'Subordinators'.

By contrast a relatively strong syntactic resemblance has been automatically recognised between words (and locutions) traditionally described as adverbs (and adverbial locutions): hence, the single 'Adverbials' word class is derived. Again, further anlysis exploiting distributional and morphological data may be useful in obtaining a finer-grained classification if necessary.

A final point to make is about copulative structures: our system proved not to prop-

[4]We refer to qualifying adjectives; other items traditionally classified as adjectives, for example 'determinative adjectives' as proposed by (Serianni, 1989), in our system are grouped together with determiners

PoS Label		Associated types	Prototypical words
Nouns		N	nuvola, finestra, tv
Verbs		V	stupire, raggiunto, concludendo, abbiamo
X	Prepositionals & Determiners	Lex<N, Lex<X, N≪Lex<N, N≪Lex<X, N≪Lex<V, X≪Lex<N, X≪Lex<V, X≪Lex<X	alcuna, della, dieci, diversi, le, molti, negli, numerose, quegli, questi, sei, sull'
	Verb-Modif. Prepositionals	V≪Lex<N, Lex<N≫V, V≪Lex<X, Lex<X≫V	a_causa_del, attraverso, contro, davanti_al, secondo, senza
	Left Adjectivals	Lex≫N	forti, giovane, grande, nuove, piccolo, suo,
	Right Adjectivals	N≪Lex, X≪Lex	economici, elettorale, idrica, importanti, positiva, ufficiale
	Adverbials	V≪Lex, Lex≫V, Lex≫X	allora, appena, decisamente, ieri, mai, molto, persino, rapidamente, presto, troppo
	Coordinators	V>Lex<V, N>Lex<N, X>Lex<X, N>Lex<X, X>Lex<N, V>Lex<X, V≪X>Lex<X, N≪V>Lex<V, N≪X>Lex<X	e, ed, ma, mentre, o, sia
	Subordinators	Lex<V, Lex<V≫V, V≪Lex<V	in_modo_da, oltre_a, quando, perché, se
	Relatives	N>Lex	che, cui, dove, quale
	Entities	Lex	ci, di_più, in_salvo, io, inferocito, noi, ti, sprovveduto, una

Table 1: Resulting PoS classification

erly process them in general, as shown by the fact that their predicative components ended up classified under either 'Entities' or 'Prepositionals & Determiners'.

5 Conclusions and Further Research

The final output of the three phase system will be a *hierarchy* of PoS tags. Such structured organization is expected to help the linguist during the annotation phase as well as when searching the annotated corpus.

On the one hand, the linguist can browse the graph for a given word to get a sense of its syntactic distribution or to improve the proposed classification (e.g. by splitting an induced category that is too coarse.)

On the other hand, since the resulted PoS classification is organized as a hierarchy with inclusion relations, a more intelligent search interface can be constructed to help the user extract the relevant information from the annotated corpus.

References

S. Bangalore and A. Joshi. 1999. Supertagging: An approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265.

C. Bosco, V. Lombardo, Vassallo D., and Lesmo L. 2000. Building a treebank for Italian: a data-driven annotation schema. In *Proc. 2nd International Conference on Language Resources and Evaluation - LREC 2000*, pages 99–105, Athens.

C. Bosco. 2003. *A grammatical relation system for treebank annotation*. Ph.D. thesis, Computer Science Department, Turin University.

E. Brill and M. Marcus. 1992. Tagging an unfamiliar text with minimal human supervision. In *Proceedings of the Fall Symposium on Probabilistic Approaches to Natural Language*, pages 10–16, Cambridge.

A. Clark. 2000. Inducing Syntactic Categories by Context Distribution Clustering. In *Proceedings of CoNLL-2000 and LLL-2000 Conference*, pages 94–91, Lisbon, Portugal.

Y. Kawata. 2005. *Tagsets for Morphosyntactic Corpus Annotation: the idea of a 'reference tagset' for Japanese*. Ph.D. thesis, University of Essex, Colchester, UK.

M. Monachini. 1995. ELM-IT: An Italian Incarnation of the EAGLES-TS. Definition of Lexicon Specification and Classification Guidelines. Technical report, Pisa.

F. Pereira, T. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st ACL*, pages 183–190, Columbus, Ohio.

M. Redington, N. Chater, and S. Finch. 1998. Distributional Information: a Powerful Cue for Acquiring Syntactic Categories. *Cognitive Science*, 22(4):425–469.

R. Rossini Favretti, F. Tamburini, and C. De Santis. 2002. CORIS/CODIS: A corpus of written Italian based on a defined and a dynamic model. In A. Wilson, P. Rayson, and T. McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*. Munich: Lincom-Europa.

H. Schütze. 1993. Part-of-speech induction from scratch. In *Proceedings of the 31st ACL*, pages 251–258, Columbus, Ohio.

L. Serianni. 1989. *Grammatica italiana. Italiano comune e lingua letteraria*. UTET, Torino.

F. Tamburini, C. De Santis, and Zamuner E. 2002. Identifying phrasal connectives in Italian using quantitative methods. In S. Nuccorini, editor, *Phrases and Phraseology -Data and Description*. Berlin: Peter Lang.

A Dual-Iterative Method for Concept-Word Acquisition from Large-Scale Chinese Corpora

Guogang Tian
Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
Graduate University of the Chinese Academy of Sciences
Beijing, China 100080
naitgg@hotmail.com

Cungen Cao
Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
Graduate University of the Chinese Academy of Sciences
Beijing, China 100080
cgcao@ict.ac.cn

Abstract - This paper proposes a dual-iterative method, a hierarchical inner and outer iteration method (HIO), to acquire concept words from a large-scale, un-segmented Chinese corpus. It has two levels of iteration: the EM-CLS algorithm and the Viterbi-C/S algorithm constitute the inner iteration for generating concept words, and the concept word validation constitutes the outer iteration together with the concept word generation. Through multiple iterations, it integrates the concept word generation and validation into a uniform acquisition process. In the process of acquisition, the HIO method can cope with the problem of over-segmentation, over-combination and data sparseness. The experimental result shows that the HIO method is valid for concept word acquisition that can simultaneously increase the precision and recall rate of concept word acquisition.

1. Introduction

Concept word acquisition is an important research in knowledge acquisition from text (KAT) (Cao and Sui, 2003), and it is also the foundation of ontology learning (Maedche, 2002). Its main purpose is to acquire plentiful concept words from text corpora. It is very similar to unknown word recognition (Chen and Bai, 1998), (Feng, Chen, et al., 2004) and term extraction (Bourigault and Jacquemin, 1999). However, there are subtle distinctions among these three researches. Generally, concept word can be classified into three types: proper name, compound word and derived word. Except for these three word types, unknown word recognition also identifies numeric-type compounds, and it does not concern known words listed in a dictionary. Term extraction (Bourigault and Jacquemin, 1999) mainly processes domain texts, and often extracts commonly used professional terms from a specific domain text corpus.

Fu and Luke (2003) proposed a two-stage Chinese segmentation system. At the first stage, it segmented the input text according to known words on the basis of 2-gram statistical model, and then identified unknown words at the second stage using a hybrid method which consisted of word context, word composition and word juncture model.

Yang and Li (2003) proposed a heuristic method that it generated five rules using mutual information and significance estimation to extract unknown word.

Peng and Schuurmans (2001) proposed an unsupervised training method to build probability models that accurately segmented Chinese character sequences into words. It used successive EM phases to learn a good probability model over character strings, and then prunes the model with a mutual information selection criterion to obtain a more accurate word lexicon.

Lai and Wu (2000, 2002) proposed a likelihood ratio method to extract possible unknown words or phrases defined by PLUs (phrase-like-unit). The final PLU was decided by two principles of overlap competition and inclusion competition.

Nagao and Mori (1994) proposed a rapid n-gram extraction method to extract adjacent substrings with same prefix in an ordered prefix table. It was noted that it was an affix method intrinsically.

From these above works, we can summarize that there are two kinds of method to identify or acquire unknown words, that is, the non-iterative statistical method and the affix method. The non-iterative unknown word recognition (Fu and Luke, 2003), (Yang and Li, 2003) , (Peng and Schuurmans, 2001) , (Lai and Wu, 2000, 2002), (Zhang, Lv, et al., 2003) usually adopts n-gram statistical model that is combined with segmentation and combination operation to identify unknown words. It can deal with over-segmentation, but can not tackle over-combination. In addition, the length of unknown word must be restricted in order to ensure system performance. The acquired unknown words are often 2-grams, 3-grams and 4-grams. The affix method (Nagao and Mori, 1994) has even more

Proceedings of the Australasian Language Technology Workshop 2005, pages 184–190,
Sydney, Australia, December 2005.

limits. For example, it can not deal with unknown words that have not obvious affix features, and it can not use contextual information of unknown words, either.

This paper, motivated by the work of Chang and Su (1997) and Liu, Zhang, et al. (2004) presents a hierarchical inner and outer iteration method to acquire concept words from a large-scale, un-segmented Chinese text corpus. It has two levels of iteration which involves concept word generation and validation. It makes some extension on EM algorithm and Viterbi algorithm which make up the concept word generation. The concept word validation combines mutual information and context entropy into a validation criterion. These two levels of iteration can simultaneously increase the precision and recall rate of concept word acquisition.

The main contribution of this paper is that it proposes a HIO method for concept word acquisition. The HIO method unifies concept word generation and validation into a consecutively iterative process so that it can increase precision and recall simultaneously. The rest of this paper is organized as follows: Section 2 presents the HIO method. Concept word generation is discussed in section 2.1, and concept word validation is discussed in section 2.2. The whole HIO algorithm is presented in Section 2.3. The experiment result and error analysis are provided in section 3. Section 4 concludes this paper and outlines the future work.

2. The HIO Method

The HIO method (a Hierarchical Inner and Outer iteration method) has two levels of iteration, that is, the inner iteration and the outer iteration. The alternation of EM-CLS and Viterbi-C/S algorithm constitutes the inner iteration of the HIO – concept word generation, and concept word validation constitutes the outer iteration of the HIO. The basic structure of the HIO method is illustrated in Fig. 1.

The HIO method can cope with the two primary problems in concept word acquisition: over-segmentation and over-combination. Data sparseness is one of common problems in statistical language processing. Concept word acquisition is not the exception. In the acquisition process, it may produce the sparse data. Katz smoothing is applied in the HIO method to smooth sparse data and reduce their effect on concept word acquisition.

2.1 Concept Word Generation

2.1.1 EM-CLS algorithm

The EM-CLS algorithm, which is based on EM (expectation maximization) algorithm, estimates generated terms' probability distribution and identifies their types in a large corpus.

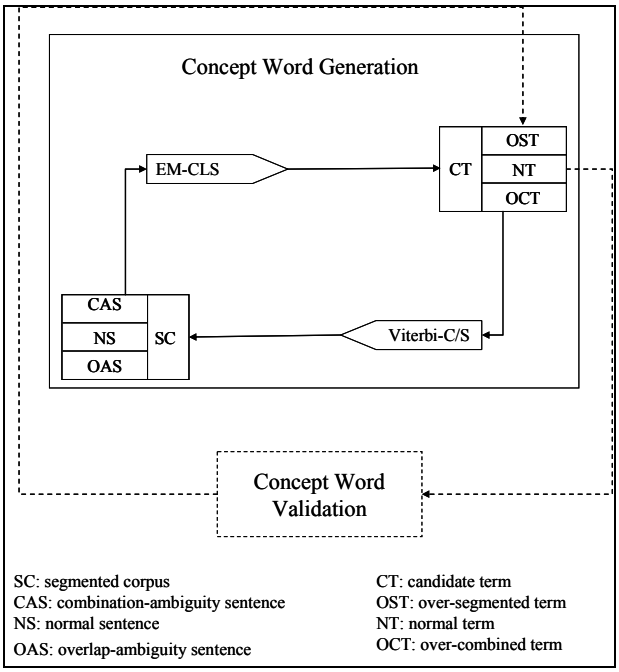


Fig.1. The Structure of HIO Method

EM algorithm (Figueiredo, 2004), (Prescher, 2003) is a common method for estimating maximum-likelihood when missing data are present. It has two steps: *E-step* (expectation step) and *M-step* (maximization step). Given the observed data x and the current parameter estimation $\hat{\theta}^{(t)}$, E-step computes the conditional expectation (with respect to the missing data y) of the logarithm of a complete *posteriori* probability function, *logp(y,θ|x)*. Usually E-step is called as Q function, as illustrated in (1). Equation (2) shows the M-step of EM algorithm. M-step chooses the parameters which can maximize Q function as the estimated parameters. Through consecutive iterations of E-Step and M-Step, EM algorithm can get stabilized parameters.

E-Step:

$$\begin{aligned} Q(\theta \mid \hat{\theta}^{(t)}) &\equiv E[\log p(y,\theta \mid x) \mid x, \hat{\theta}^{(t)}] \\ &\propto \log p(\theta) + E[\log p(y,x \mid \theta) \mid x, \hat{\theta}^{(t)}] \\ &= \log p(\theta) + \int p(y \mid x, \hat{\theta}^{(t)}) \log p(y,x \mid \theta) dy \end{aligned} \quad (1)$$

M-Step:

$$\hat{\theta}^{(t+1)} = \arg\max_{\theta} Q(\theta \mid \hat{\theta}^{(t)}) \quad (2)$$

An un-segmented corpus is denoted as $C=\{C_1, C_2, \ldots, C_n\}$ where $C_i (1 \leqslant i \leqslant n)$ represents an

un-segmented sentence. After segmentation, C is converted into the segmented corpus denoted as $S=\{S_1, S_2, \ldots, S_n\}$ where S_i $(1 \leqslant i \leqslant n)$ is a segmentation of C_i. The generated candidate terms[1] are grouped into a set denoted as $T=\{t_1, t_2, \ldots, t_m\}$, where t_j $(1 \leqslant j \leqslant m)$ is the generated candidate term.

If C_i is taken as the observed data, S_i as the missing data, we can estimate the maximum-likelihood of term t_j with the EM algorithm which is deemed as its probability distribution in the corpus C. Equation (3) shows the probability estimation of term t_j. In (3), S_i^* denotes the optimal segmentation of sentence C_i, which can be achieved by the Viterbi-C/S algorithm (to be discussed in the next section). $f(t_j, S_i)$ denotes the frequency of term t_j in sentence S_i. $\hat{t}_j = p(t_j), \hat{T} = \{p(t_j) \mid t_j \in T\}$.

After estimating the probability of term t_j, we still have to judge to which type it belongs. The candidate term has three types that are *normal term*, *over-segmented term* and *over-combined term*.

$$\hat{t}_j^{(t+1)} = \frac{\sum_{i=1}^{n} f(t_j, S_i) \times p(S_i^* \mid C_i, \hat{T}^{(t)})}{\sum_{j=1}^{m}\sum_{i=1}^{n} f(t_j, S_i) \times p(S_i^* \mid C_i, \hat{T}^{(t)})} = \frac{\sum_{i=1}^{n} f(t_j, S_i) \times p(S_i^*, C_i \mid \hat{T}^{(t)})}{\sum_{j=1}^{m}\sum_{i=1}^{n} f(t_j, S_i) \times p(S_i^*, C_i \mid \hat{T}^{(t)})} \quad (3)$$

If a concept word (or meaningful word) is segmented into several components, it is called *over-segmented term.* For example, *高 血 糖 (hyperglycemia)* is possibly spitted into *高 (high)* and *血糖 (blood sugar)*.

If a word is combined with another word, but their combination is not a concept word (or meaningful word), it is called *over-combined term*, such as *但也 (but also)*.

Equation (4) can assign a type label to t_j, which is denoted by $CLS(t_j)$.

$$CLS(t_j) = \arg\max_{H^{(i)}} \frac{|\{S_k \mid S_k \in Sen(t_j) \wedge S_k \in H^{(i)}\}|}{|Sen(t_j)|} \quad (4)$$

In (4), $Sen(t_j)=\{S_i \mid t_j \in S_i\}$, $H^{(i)}$ denotes the sentence type label.

2.1.2 Viterbi-C/S algorithm

The Viterbi-C/S algorithm dynamically segments a corpus using the estimated probability of candidate terms and executes combination and segmentation operations on ambiguous terms in order to achieve the optimal segmentation. After completing corpus segmentation, it judges if a sentence contains overlap ambiguity or combination ambiguity.

Given a segmented sentence S_i, $S_i=t_1t_2\ldots t_k$ $(1 \leqslant j \leqslant k, t_j \in T)$, it is assumed that terms are independent each other, *the likelihood of sentence S_i* is defined as:

$$p(S_i) = \prod_{j=1}^{k} p(t_j) \quad (5)$$

Definition 1: It is *the optimal segmentation* that its likelihood is maximal among all segmentations of a sentence. The optimal segmentation is denoted as S_i^*

$$S_i^* = \arg\max_{S_i} p(S_i \mid C_i, \hat{T}) = \arg\max_{S_i} p(S_i, C_i \mid \hat{T}) \quad (6)$$

Like candidate terms, segmented sentences are also classified into three types: normal sentence (*N-Sen*), overlap-ambiguity sentence (*OA-Sen*), and combination-ambiguity sentence (*CA-Sen*).

If a segmented sentence contains over-combined terms, it is considered as an *OA-Sen*.

If a segmented sentence contains over-segmented terms, it is considered as a *CA-Sen*.

It is observed that there is a direct correspondence between the type of candidate term and segmented sentence: *normal term – N-Sen*, *over-segmented term – CA-Sen* and *over-combined term – OA-Sen*.

Definition 2: Segmented Density is defined as the number of segmented term in each length unit.

For a sentence S_i,

$$SD(S_i) = \frac{p(S_i) \times NT(S_i)}{length(S_i)} \quad (7).$$

For a corpus S,

$$SD(S) = \frac{\sum_{S_i \in S} p(S_i) \times NT(S_i)}{length(S)} = \frac{\sum_{S_i \in S} p(S_i) \times NT(S_i)}{\sum_{S_i \in S} length(S_i)} \quad (8).$$

In (7)-(8), *NT(X)* denotes the number of terms in sentence *X*, and *length(Y)* denotes the length of sentence *Y*.

The type of segmented sentence is measured by (9). Setting a threshold range $[r_1, r_2]$ $(r_1<r_2)$, if $CLS(S_i)<r_1$, S_i is a *OA-Sen*, if $CLS(S_i)>r_2$, S_i is a *CA-Sen*, if $r_1 \leqslant CLS(S_i) \leqslant r_2$, S_i is a *N-Sen*.

$$CLS(S_i) = \frac{SD(S_i)}{SD(S)} \quad (9)$$

[1] The meaning of "term" here is different from the meaning of "term" in term extraction. Here term refers to ordinary word.

We make an extension to the classical Viterbi algorithm (Rabiner, 1989), thus get the Viterbi-C/S algorithm as illustrated in Fig. 2.

When segmenting a corpus, Viterbi-C/S binds combination and segmentation operations (C/S operation) on the selected terms according to their types. If over-segmented term is selected, combination operation is performed, if over-combined term is selected, segmentation operation is performed. These combination or segmentation operations on candidate term possibly causes data sparseness problem. So we use Katz Smoothing method (Goodman, 2001) as the smoothing strategy to eliminate sparse data.

$$p(t_{i-n+1}...t_{i-1}t_i) = \alpha(t_{i-n+1}...t_{i-1})p(t_{i-n+2}...t_{i-1}t_i) \quad (10)$$

In (10), t'=$t_{i-n+1}\ldots t_{i-1}t_i$ doesn't exist in candidate term set T, α is a normalization constant.

2.2 Concept Word Validation

The generated candidate terms need to be further validated to filter out ambiguous terms. The validation takes into considerations candidate term's composition and local context. The former is considered as a cohesion validation which adopts mutual information method (Sproat and Shih, 1990). The latter is considered as an independence validation which adopts context entropy method (Tung and Lee, 1994). So the concept word validation is the combination of mutual information and context entropy method.

Viterbi-C/S Algorithm

Input: un-segmented corpus C, candidate terms' probability estimation

Output: the optimal segmentation and its type

1. selecting a sentence C_i from corpus C;
2. selecting all possible candidate terms at the current position of sentence C_i, which constitute a set denoted as T^p={t_1^p, t_2^p, t_3^p, …};
3. selecting a candidate term which has maximum-likelihood from the set T^p as a possible segmented term of sentence C_i, denoted as *st*;
4. performing the corresponding operation according to the type of term *st*
   a. if an over-segmented term, performing segmenting operation on it and re-estimating the likelihood of new term, goto (3);
   b. if an over-combined term, performing combining operation on it and re-estimating the likelihood of new term, goto (3);
   c. if a normal term, segmenting the sentence, computing the likelihood of current segmentation and moving the position pointer forwardly
5. repeating step 1-4, until all sentences in corpus are segmented
6. computing segmented density for corpus and sentences, and determining the type label of sentence

Fig. 2. The Viterbi-C/S Algorithm

The basic assumptions of concept word validation are that:

If a term is an over-combined term, it contains at least a division point where its cohesion degree must be low.

If a term is an over-segmented term, its local context features in corpus must be weak.

2.2.1 Mutual Information

It is assumed that there is at most two division points in a validating term $t_v = c_1c_2\ldots c_n$.

If $t_v^l = c_1c_2\ldots c_l \in T$ ($1 \leqslant l < n$) and t'= $c_1c_2\ldots c_lc_{l+1} \notin T$, t_v^l is called the *maximal left substring* of t_v, and *l* is the *left division point* of t_v.

If $t_v^r = c_rc_{r+1}\ldots c_n \in T$ ($1 < r \leqslant n$) and t'= $c_{r-1}c_r\ldots c_n \notin T$, t_v^r is called the *maximal right substring* of t_v, and *r* is the *right division point* of t_v.

(1) If l<r-1, t_v has two division points, which is denoted as $t_v = t_v^2 = t_v^l t_v^m t_v^r$;

(2) If l=r-1, t_v has a division point, which is denoted as $t_v = t_v^1 = t_v^l t_v^r$;

(3) If l⩾r, t_v has two possible divisions which are denoted as $t_v = t_v^{1\text{-}L} = t_v^l t_v^{-l}$ and $t_v^{1\text{-}R} = t_v = t_v^{-r} t_v^r$ respectively.

To case (1)

$$MI(t_v) = MI(t_v^2) = \log \frac{p(t_v^l t_v^m t_v^r)}{p(t_v^l)p(t_v^m)p(t_v^r) + p(t_v^l)p(t_v^m t_v^r) + p(t_v^l t_v^m)p(t_v^r)} \quad (11)$$

To case (2),

$$MI(t_v) = MI(t_v^1) = \log \frac{p(t_v^l t_v^r)}{p(t_v^l)p(t_v^r)} \quad (12)$$

To case (3),

$$MI(t_v^{1-L}) = \log \frac{p(t_v^l t_v^{-l})}{p(t_v^l)p(t_v^{-l})};$$

$$MI(t_v^{1-R}) = \log \frac{p(t_v^{-r} t_v^r)}{p(t_v^{-r})p(t_v^r)}; \quad (13)$$

$$MI(t_v) = \min\{MI(t_v^{1-L}), MI(t_v^{1-R})\}$$

Before computing the mutual information of the validating term, we above all identify to which type it belongs among the above case (1) to (3) and then adopt the corresponding equation (11-13). Similarly, we still apply equation (10) to deal with data sparseness problem.

2.2.2 Context Entropy

It is assumed that t_v is a validating term. Its left

context is denoted as $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_l\}$, and its right context is denoted as $\beta=\{\beta_1, \beta_2, \ldots, \beta_r\}$. The left context entropy, right context entropy and context entropy of the validating term t_v is defined in (14).

$$Entr_L(t_v) = -\sum_{\alpha_i \in \alpha} p(\alpha_i t_v) \log\ p(\alpha_i t_v);$$
$$Entr_R(t_v) = -\sum_{\beta_i \in \beta} p(t_v \beta_i) \log\ p(t_v \beta_i); \quad (14)$$
$$Entr\ (t_v) = \min\{\ Entr_L(t_v), Entr_R(t_v)\}$$

Table 1 lists the joint validation rules combining mutual information and context entropy criterions. th_{mi} and th_{entr} are thresholds we designate to mutual information and context entropy, respectively.

The wrong candidates are removed from the dictionary. The other three types of terms are saved into the candidate dictionary again. After validating terms, concept word generation is restarted again and the concept word acquisition goes into the next iteration.

2.3 The HIO Algorithm

The HIO Algorithm is illustrated in Fig.3.

HIO Algorithm

Input: un-segmented corpus *C*, system dictionary, the predefined iteration number (including outer iteration and inner iteration)

Output: concept words set

1. Initialization
   a. identifying *numeric-type string* and *English string*;
   b. extracting concept words matching with specific pattern;
   c. segmenting corpus *C* with dictionary and *Forward Maximum Matching algorithm*, and estimating initial probability of terms
2. outer iteration
   a. concept word generation
      I. estimating term probability and identifying term's type using *EM-CLS* algorithm;
      II. achieving the optimal segmentation of sentence and identifying their type using *Viterbi-C/S* algorithm;
      III. if the number of iteration < the predefined number, goto (Ⅰ);
   b. concept word validation
      I. validating normal candidate terms using *mutual information and context entropy*, identifying their type and precluding wrong results
      II. if the number of iteration < the predefine number, goto (2.a)
3. outputting all normal candidate terms as concept words set

Fig.3. The HIO Algorithm

3. Experimental Result and Error Analysis

Table 1. The Joint Validation of MI and Entropy

$MI(t_v)$	$Entr(t_v)$	Term type(t_v)
$\geq th_{mi}$	$\geq th_{entr}$	correct candidate
$\geq th_{mi}$	$< th_{entr}$	over-segmented
$< th_{mi}$	$\geq th_{entr}$	over-combined
$< th_{mi}$	$< th_{entr}$	wrong candidate

We adopt a *400M* Chinese corpus extracted from web pages as the experimental corpus. Before running the HIO method, a series of preprocessing operations are performed, which involve recognizing special unknown words such as numeric-type words, English words, etc., acquiring concept words matching with specific context patterns, using forward maximum matching method to initially segment the corpus and estimating the initial probability of terms.

We set the inner iteration to 10 times and the outer iteration to 5 times. When completing the HIO operations, we randomly select 2000 sentences from this corpus. After filtering out many common words such as auxiliary words, adjectives and adverbs, we get many concept words which have higher precision and recall rates as listed in Table 2.

Table 2. The Experimental Result

length	Count	P (%)	R (%)
2	7782	92.2	83.4
3	2234	86.1	69.0
4	1627	89.3	70.4
5	1893	94.7	60.3
≥6	856	91.6	51.1
Sum.	14392	91.2	73.1

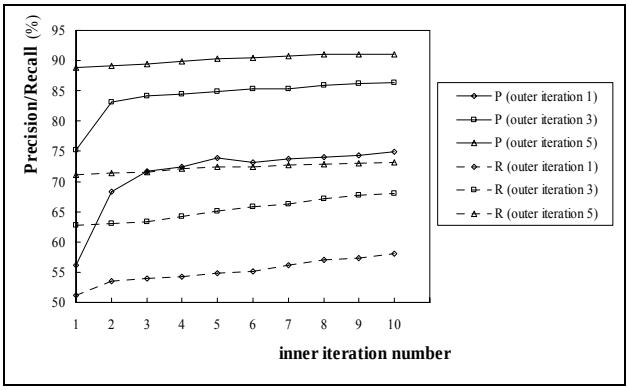


Fig. 4. The Precision and Recall *w.r.t.* the Inner and Outer Iteration

We get 5387 unknown words in total 14392 terms, among which there are 833 bi-gram words, 1593 tri-gram words, 1049 four-gram words, 1196 five-gram words, 716 six- and over-six-gram words.

Fig. 4 shows the effect of the inner and outer iteration on the precision and recall rate of concept word acquisition. It is observed that the precision and recall rate are both increased with the increase of iteration times.

There are two types of errors produced in HIO: commission error and omission error (Yang and Li, 2004). A commission error is that an acquired term is actually not a concept word, but the HIO considers it as a concept word. The reason is that every component of this term is common words and often occurs simultaneously. An omission error is that the HIO misses a concept word in the corpus. The reason is that one component of this word is more commonly used than the rest and the statistical feature of their combination is not prominent. However, the error distribution we get is contrary to the result of Yang and Li (2004). The number of omission errors exceeds that of commission errors, especially in tri-gram concept word.

4. Conclusions and Future Work

This paper proposes a hierarchical inner and outer iteration method (HIO) for concept word acquisition. It can deal with the problem of over-segmentation, over-combination and data sparseness produced in the process of acquisition. Its prominent features involve:

(1) The HIO method is the combination of the inner and outer iteration, which can increase the precision and recall rate of concept words acquisition simultaneously.
(2) Concept word generation and validation are uniform and consistent in the HIO method.
(3) The EM-CLS algorithm can classify candidate terms as well as estimate their probability distribution.
(4) The Viterbi-C/S can perform segmenting and combining operations on terms while segmenting corpus.
(5) HIO uses Katz smoothing to lessen data sparseness effect on concept word acquisition.

Now we are going on a series of research on knowledge acquisition from text. The acquired knowledge types include concepts and their relations. Concept word acquisition is fundamental, which can provide essential support for other work in KAT research. We are also developing methods for acquiring relations, including isa, part-of and co-title.

Acknowledgement

This work is supported by the Natural Science Foundation (grant no. 60273019 and 60496326), and the National 973 Programme (grant no. 2003CB317008).

References

Christopher C. Yang and K.W. Li. 2003. Segmenting Chinese Unknown Words by Heuristic Method, *ICADL 2003*, LNCS 2911, pp 510–520

Christopher C. Yang and K.W. Li. 2004. Error Analysis of Chinese Text Segmentation using Statistical Approach. *Proceedings of 2004 Joint ACM/IEEE Conference on Digital Library (JCDL' 04)*, Tucson, Arizona, USA, pp256-257

Cungen Cao and Yuefei Sui. 2003. Constructing Ontology and Knowledge Bases from Text. *20th International Conference on Computer Processing of Oriental Languages*, Shenyang, China, pp 34-42

Detlef Prescher. 2003. A Tutorial on the Expectation-Maximization Algorithm Including Maximum-Likelihood Estimation and EM Training of Probabilistic Context-Free Grammars, Presented at the 15th European Summer School in Logic, Language and Information (ESSLLI 2003), Vienna, Austria, August 18-29, 2003

Didier Bourigault and Christian Jacquemin. 1999. TERM EXTRACTION + TERM CLUSTERING: An Integrated Platform for Computer-Aided Terminology. *Proceedings of EACL '99*, pp. 15-22

Fuchun Peng and Dale Schuurmans. 2001 Self-supervised Chinese Word Segmentation. *In Advances in Intelligent Data Analysis (Proceedings of IDA-01)*, pp 238-247

Goodman, J. T. 2001. A Bit of Progress in Language Modeling. *Computer Speech and Language*, 2001 (10), pp 403-434

Guhong Fu and K.K Luke. 2003. A two-stage statistical word segmentation system for Chinese, *Second SIGHAN Workshop on Chinese Language Processing*, Sapporo, Japan, pp. 156-159

Haodi Feng, Kang Chen, Xiaotie Deng, Weimin Zheng. 2004. Accessor Variety Criteria for Chinese Word Extraction, Computational Linguistics, 30 (1), pp. 75-93

Jing-Shin Chang and Keh-Yih Su. 1997. An Unsupervised Iterative Method for Chinese New Lexicon Extraction, *Computational Linguistics and Chinese Language Processing,* 2 (2), pp 97-148

Keh-Jiann Chen and Ming-Hong Bai. 1998. Unknown word detection for Chinese by a corpus-based learning method, *International Journal of Computational Linguistics and Chinese Language Processing*, 3(1):27–44

Lawrence R. Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in

Speech Recognition. *Proceedings of the IEEE*, 77 (2), pp 257-286

Liu Qun, Zhang Huaping, Yu HongKui, and Cheng Xueqi. 2004. Chinese Lexical Analysis Using Cascaded Hidden Markov Model, *Journal of Computer Research and Development*, 41(8), pp. 1421-1429

Maedche. 2002. Ontology Learning for the Semantic Web. Kluwer Academic Publishers

Mário A. T, Figueiredo. 2004. Lecture Notes on the EM Algorithm, http://www.lx.it.pt/~mtf/learning/aboutEM.pdf

Nagao, M and Mori,S. 1994. A New Method of N-gram Statistics for Large Number of N and Automatic Extraction of Words and Phrases from Large Text Data of Japanese. *COLING-94*

Sproat, R. and Shih, C. 1990. A Statistical Method for Finding Word Boundaries in Chinese Text, *Computer Processing of Chinese and Oriental Languages*, 1990 (4), pp.336–351

Tung, Cheng-Huang and Hsi-Jian Lee. 1994. Identification of Unknown Words from a Corpus, *Computer Processing of Chinese and Oriental Languages*, Vol. 8, pp. 131-145

Yusheng Lai and ChungHsien Wu. 2000. Unknown Word and Phrase Extraction Using a Phrase-Like-Unit-Based Likelihood Ratio. *International Journal of Computer Processing of Oriental Languages*, 13 (1), pp 83–95

Yu-sheng Lai and Chung-hsien Wu. 2002. Meaningful Term Extraction and Discriminative Term Selection in Text Categorization via Unknown-Word Methodology. *ACM Transactions on Asian Language Information Processing*, Vol. 1, No. 1, March 2002, pp 34-64

Zhang Le, Lv Xue-qiang, Shen Yan-na and Yao Tian-shun. A Statistical Approach to Extract Chinese Chunk Candidates from Large Corpora. *ICCPOL 2003*, pp109-117, 2003

Programming With Unrestricted Natural Language

David Vadas and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dvadas1,james}@it.usyd.edu.au

Abstract

We argue it is better to program in a natural language such as English, instead of a programming language like Java. A natural language interface for programming should result in greater readability, as well as making possible a more intuitive way of writing code. In contrast to previous controlled language systems, we allow unrestricted syntax, using wide-coverage syntactic and semantic methods to extract information from the user's instructions.

We also look at how people actually give programming instructions in English, collecting and annotating a corpus of such statements. We identify differences between sentences in this corpus and in typical newspaper text, and the effect they have on how we process the natural language input. Finally, we demonstrate a prototype system, that is capable of translating some English instructions into executable code.

1 Introduction

Programming is hard. It requires a number of specialised skills and knowledge of the syntax of the particular programming language being used. Programmers need to know a number of different languages, that can vary in control structures, syntax, and standard libraries. In order to reduce these difficulties, we would like to express the steps of the algorithm we are writing in a more natural manner, without being forced into a particular syntax. Ideally, we want a *plain English description*.

We have built an initial prototype of such a system, taking unrestricted English as input, and outputting code in the Python programming language. There are many advantages of such a system. Firstly, any person that can write English, but not a programming language, would still be able to program. Also, it is often easier to write an English sentence describing what is to be done, than to figure out the equivalent code. Many programmers write in a pseudocode style that is almost English before elaborating on the details of an algorithm. There are also many tasks that can easily be described using English sentences, but are much harder to express as code, such as negation and quantification.

Another advantage is that code written in English will be much easier to read and understand than in a traditional programming language. Quite often, it is a difficult task to read another programmer's code. Even understanding one's own code can be hard after a period of time. This is because without sufficient commenting — this is an explanation in plain English — one cannot tell what individual steps are meant to do together. In our system, the comments become the code.

Novice programmers could make great use out of such a system. They make simple syntax errors because they do not know the language well enough. Similarly, a novice programmer may know what function they want to use, but not its specific name and required arguments.

Finally, standard programming languages exhibit numerous technical details that are not evident in natural languages. Examples of this include typing, integer division and variable declarations. When we say in English $\frac{3}{5}$, we expect the result to be 0.6, not 0, as will result in many programming languages. These complications are a result of the computer's implementation, rather than the algorithm we are trying to describe. We would like to abstract away these issues, using information present in the English sentences to figure out the correct action to take.

2 An Example

We can see in Figure 1 two example programs that could be entered by a user. The code for the first program matches what is outputted by the current system, but the second is more complicated and does yet work correctly.

Looking at the these examples, we can see a number of difficulties that make the problem hard, as well as form some intuitions that can help to solve the task. For example, the first line of both programs involves three function calls because of variable typ-

Proceedings of the Australasian Language Technology Workshop 2005, pages 191–199,
Sydney, Australia, December 2005.

ENGLISH	PYTHON
read in a number	`number = int(sys.stdin.readline().strip())`
add 2 to the number	`number += 2`
print out the number	`print number`
read in 2 numbers	`number1 = int(sys.stdin.readline().strip())` `number2 = int(sys.stdin.readline().strip())`
add them together	`result = number1 + number2`
print out the result	`print result`

Figure 1: Some example English sentences and their Python translations.

ing. In Python, we must first read in a string, then strip away the newline character, and finally convert it to an integer. We can tell that integer conversion is required, firstly because of the name of the variable itself, and secondly, because a mathematical operation is applied to it later on. Of course, it is still ambiguous. The user may have expected the number to be a string, and to have the string 2 concatenated to what was read in. However, the code in Figure 1 is more likely to be correct, and if the user wants to use a string representation, then they could specify as much by saying: read in a number as a string.

Another problem to deal with is the referencing of variables. In the first program, it is fairly easy to know that `number` is the same variable in all three sentences, but this is not as easy in the second. For the first sentence of the second program, the system needs to interpret 2 numbers correctly, and map it to multiple lines of code. Another complication is them, which references the previously mentioned variables. Finally, result, which does not appear in the second line, must still be part of the equivalent code, so that it can be used later.

One possibility that we could use to simplify the task that we are undertaking is to use a restricted natural language. However, we do not want to restrict the vocabulary available to a user, or force them to construct sentences in a specific way, as is the case for existing restricted natural languages (Fuchs and Schwitter, 1996). Of course, this means that we must then deal with the inherent ambiguity and the great breadth of unrestricted natural English. For this reason, we employ wide-coverage syntactic and semantic processing, that is able to process this extensive range of inputs. In order to resolve ambiguities, we can apply the intuitions we have described above. We may not be sure that the number should be treated as an integer, but this is more likely than treating it as a string. This is the conclusion that our system should come to as well.

3 Background

Clearly, the task we are undertaking is not trivial. Though there are a number of related systems to the one we propose, which have had success implementing a natural language interface for some task.

3.1 Natural Language Interfaces to Databases

The most popular task is a Natural Language Interface for a Database (NLIDB) (Androutsopoulos et al., 1995). This is because databases present a large amount of information, which both novice and expert users need to query. A specific query language such as SQL must be used, which requires one to understand the syntax for entering a query, and also the way to join the underlying tables to extract data that is needed. A NLIDB simplifies the task, by not requiring any knowledge of a specific query language, or of the underlying table structure of the database. We can see how this is similar to the English programming system that we are constructing. Both take a natural language as input, and map to some output that a computer can process.

There are a number of problems that exist with NLIDBs. Firstly, it is not easy to understand all the ambiguity of natural language, and as such, a NLIDB can simply respond with the wrong answers. As a result of this, many NLIDBs only accept a restricted subset of natural language. For example, in the NLIDB PRE (Epstein, 1985), relative clauses must come directly after the noun phrases they are attached to.

One feature of many NLIDBs, is the ability to engage the user in a dialogue, so that past events and previously mentioned objects can be referenced more easily. Two examples of this, anaphora and elliptical sentences, are shown in Figure 2.

Understanding that *it* refers to the ship, and that the female manager's degrees are again the subject of the question, reduces the amount of effort required by the user, and makes the discourse more natural. We also intend to maintain a discourse between the user and the computer for our own sys-

- ANAPHORA

> *Is there a ship whose destination is unknown?*
Yes.
> *What is it?*
What is [the ship whose
destination is unknown]?
Saratoga

- ELLIPTICAL SENTENCE

> *Does the highest paid female manager have any degrees from Harvard?*
Yes, 1.
> *How about MIT?*
No, none.

Figure 2: An example of anaphora and an elliptical sentence

tem. This would also allow us to resolve much of the ambiguity involved in natural language by asking the user which possibility they actually meant.

3.2 Early Systems

One of the first natural language interfaces is SHRDLU (Winograd, 1972), which allows users to interact with a number of objects in what was called *Blocksworld*. This system is capable of discriminating between objects, fulfilling goals, and answering questions entered by the user. It also uses discourse in order to better interpret sentences from the user.

There were also a handful of systems that attempted to build a system similar to what we describe in this paper (Heidorn, 1976; Biermann et al., 1983). Most of these used a restricted syntax, or defined a specific domain over which they could be used. Our system should have much greater coverage, and be able to interpret most instructions from the user in some way.

More generally, we can look at a system that interprets natural language utterances about planetary bodies (Frost and Launchbury, 1989). This system processes queries about its knowledge base, but is restricted to sentences that are covered by its vocabulary and grammar. It deals with ambiguous questions by providing answers to each possible reading, even when those readings would be easily dismissed by humans. With our system, we will determine the most likely reading, and process the sentence accordingly.

3.3 Understanding Natural Language

One thing that we have not yet considered is how people would describe a task to be carried out, if they could use English to do so. The constructs and formalisms required by traditional programming languages do not apply when using a natural language. In fact, there are many differences between the way non-programmers describe a task, to the method that would be employed if one were using a typical programming language (Pane et al., 2001). Firstly, loops are hardly ever used explicitly, and instead, aggregate operations are applied to an entire list. These two methods for describing the same action are shown in Figure 3.

- AGGREGATE

```
sum up all the values in the list
```

- ITERATION

```
start the sum at 0
for each in value in the list
    add this value to the sum
```

Figure 3: Finding the sum of the values in a list

Another point of difference comes in the way people use logical connectives such as AND and OR, which are not neccesarily meant in the strictly logical way that is the case when using a programming language. There are also differences in the way that people describe conditions, remember the state of objects, and the way they reference those objects.

HANDS (Pane et al., 2002) is a programming language that has been designed with this information, and with the idea of providing a programming interface that is more natural to a human user. This system takes a controlled language as input, but still demonstrates a number of methods, such as the aggregate operations described above, which make it possible for people to describe the actions they want performed as if they were writing in English.

There are actually many ways in which natural language constructions map onto programming concepts. These *programmatic semantics* (Liu and Lieberman, 2004) can be seen in syntactic types, where nouns map to objects or classes, verbs map to methods, and adjectives to attributes of the classes. Using these concepts could allow us to more easily understand an English sentence, and map it to a corresponding code output.

Metafor (Liu and Lieberman, 2005) is a system

that uses these ideas, taking a natural language description as input. As output, the system provides *scaffolding code*, that is, the outline for classes and methods, and only a small amount of actual content. The code is not immediately executable, but can help the programmer in getting started.

NaturalJava (Price et al., 2000) is another natural language programming system that allows users to create and edit Java programs using English commands. Each sentence in the natural language input given to the system is mapped to one of 400 manually created case frames, which then extracts the triggering word and the arguments required for that frame. The frame can generate a change in the Abstract Syntax Tree (AST), an intermediate representation of the code, which is turned in Java code later.

This system has a number of problems that we intend to improve on. Firstly, it can only handle one action per sentence. Our prototype can detect multiple verbs in a sentence, and generate code for each of them. Also, the AST representation NaturalJava uses makes it hard to navigate around a large amount of code, since only simple movement operations are available.

Another problem with NaturalJava is that it maps to specific operations that are included in Java, rather than more general programming language concepts. This means that it is not adaptable to different programming languages. We intend to be more language-neutral. A user of our system should not need to look at the underlying code at all, just as a programmer writing in C does not need to look at the machine code.

4 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) is a type-driven, lexicalised theory of grammar (Steedman, 2000). Each word receives a syntactic category that defines its predicate-argument relationship with surrounding words. We can see a simple example of this in Figure 4.

Each word is assigned a category that defines how it is involved with other words in the sentence. These relationships are carried out through a number of rules, such as forward and backward application, which can be seen in the example. Additional rules such as composition and conjuction also allow the formalism to easily capture long-range dependencies. This is particularly important for our system, as the constructions used to describe programming instructions often contain non-standard constituents such as extraction, relativization, and coordination.

These possiblities result in a large number of interpretations, as a single word can be assigned a different category depending on how it is used, and the words that surround it. However, the application of statistical parsing techniques for CCG have shown that it is capable of performing wide-coverage parsing at state-of-the-art levels (Clark and Curran, 2004).

5 English Code Corpus

In order to investigate the way that people would use English to describe a programming task, we elicited responses from programmers, asking them to describe how they would solve sample tasks. These tasks included finding the smallest number in a list, splitting a string on a character and finding all primes less than 100. The respondents were all experienced programmers, since computer science staff were all that were easily available. As a result of this, they tended to impose typical programming constructs on what they wanted to do, rather than using a simpler English sentence. For example, one respondent wrote For each number in the list compare to min., when Compare each number in the list to the min. is more straightforward. This demonstrates quite well the way that programming languages force us to use a specific unnatural syntax, rather than the freer style that a natural language allows. It also shows that experienced programmers can supply utterances that are less grammatically correct and therefore *harder* to process than what novices would be expected to write.

The corpus is comprised of 370 sentences, from 12 different respondents. They range in style quite significantly, with some using typically procedural constructs such as loops and ifs (complete with the non-sensical English statement: end loop in some cases), while others used a more declarative style.

We have semi-automatically tagged the entire corpus with CCG categories (called *supertags*). This process consisted of running the parser on the corpus, and then manually correcting each parse. Corrections were required in most sentences, as the way people express programming statements varies significantly from sentences found in newspaper text. An example of this is in Figure 5.

This sentence uses an imperative construction, beginning with a verb, which is quite different from declarative sentences found in newspaper text, and the earlier example in Figure 4. We can also notice that the final category for the sentence is $S[b]\backslash NP$, rather than simply S. Another difference is in the vocabulary used for programming tasks, compared to Wall Street Journal (WSJ) text. We find if, loop, and variables in the former, and

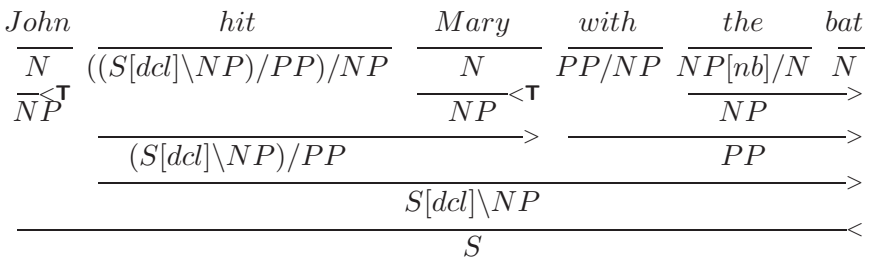


Figure 4: An example CCG derivation

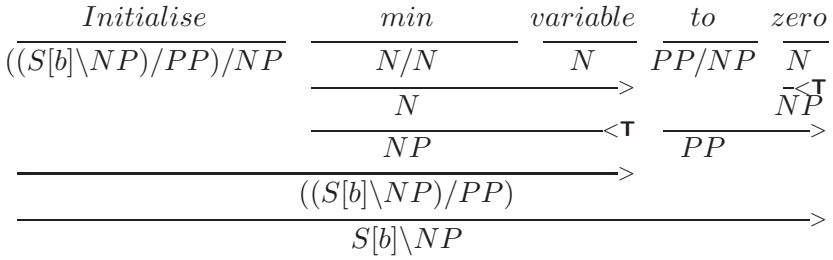


Figure 5: A CCG derivation for an English programming instruction

million, dollars, and executives in the latter. Particular words can also have different grammatical functions. For example: print is usually a noun in the WSJ, but mostly a verb while programming.

6 System Architecture

The system architecture and its components are shown in Figure 6.

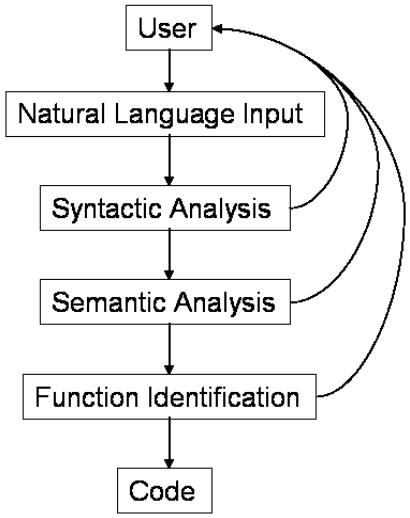


Figure 6: The system architechture

Firstly, the user will enter text that will be parsed by the CCG parser. We then translate the predicate-argument structure generated by the parser into a first-order logic representation of DRS predicates. This gives us a more generic representation of the sentence, rather than the specific wording chosen by the user. The final step is to generate the code itself.

Throughout these three phases, we also intend to use a dialogue system that will interact with the user in order to resolve ambiguity in their input. For example, if the probability with which the parser gives its output is too low, we may ask the user to confirm the main verb or noun. This is especially important, as we do not intend for the system to be foolproof, but we do intend that the user should be able to solve the problems that they encounter, either through greater specification or rephrasing.

There are also a number of smaller tasks to be dealt with, such as anaphora resolution, and GUI construction. At this current stage though, we have only dealt with basic functionality.

We will now describe each of the components of the system in detail. Also, as we progress through each stage, we will follow the example previously shown in Figure 5. We will see how the processing we do manages to begin with this English input, and eventually output working Python code.

7 Parser

We use the C&C CCG parser (Clark and Curran, 2004) for this first stage of processing. This has the advantage of being a broad coverage, robust parser, that is able to extract long range dependancies reliably. We also have access to the code, and are thus able to make changes if needed, and are able to build new training models. In fact, we found that we

did indeed need to train a new model for the parser as a result of the differences between programming statements and typical newspaper text, as described above. If we look at our example sentence, we can see some of the problems quite well. Figure 7 shows the parse provided by the original model.

We can see that Initialise not been identified as a verb, but is instead tagged as a proper noun. min is also misclassified as a verb, when it is the noun. This highlights the fact that the parser does not expect the first word in a sentence to be a verb. We could not use this parse and expect to perform adequately in the following stages of the system.

For this reason we created and annotated the English code corpus, in order to provide training data and allow us to build a new, and better performing model. A similar process had been followed in Question Answering (QA) (Clark et al., 2004), because questions also show quite different syntactic properties to newspaper text. This technique produced a significant improvement for QA, and so we have reused this idea.

Following Clark et al., we used multiples of the English corpus, as it is quite small in comparison to the entire WSJ. These results are shown in Figure 8, for training with just the WSJ (original), with the WSJ and the English code corpus (1x code), and with the WSJ and multiples of the English corpus (5x, 10x, 20x). We show results for POS tagging and supertagging, on a word-by-word basis, and also the proportion of whole lines that are tagged correctly. We can see that as we add more copies of the English code corpus, all accuracies continue to improve.

These results come from both training and testing on the English corpus, and thus are not completely rigourous. However, it does demonstrate the data is fairly consistently annotated. As a fairer comparison, we conducted 10-fold cross validation on the 20x corpus, where each fold contained the 20 copies of one-tenth of the English code corpus, together with sentences from sections 2–21 of the WSJ. Each fold contained lines from throughout the English corpus and the WSJ. The results show that the accuracies from training with the English code corpus were still significantly greater than the original model. Finally, with this new model, our example sentence is parsed correctly, as shown in Figure 5.

8 Semantics

From the syntactic representation of the sentence, we wish to build a more semantically abstracted version of what the user wants to translate into code. The advantage of this is that we can more readily extract the particular verbs and nouns that will become functions and their arguments respectively. Having a logical form also means we can apply inference tools, and thereby detect anomolies in the user's descriptions, as well as including other sources of knowledge into the system.

```
%%% Initialise 'min' variable to zero .
 _________________
| x4 x3 x5 x1 x2  |
|-----------------|
| thing(x4)       |
| 'min'(x5)       |
| nn(x5,x3)       |
| variable(x3)    |
| initialise(x1)  |
| agent(x1,x4)    |
| patient(x1,x3)  |
| |x2|>0         |
| to(x1,x2)       |
| event(x1)       |
|_________________|
```

Figure 9: DRS for example sentence

The ccg2sem system (Bos et al., 2004; Blackburn and Bos, 2005) performs this task, taking CCG parse trees as input, and outputting DRS logical predicates. A single unambiguous reading is always outputted for each sentence. The DRS for our example sentence is shown in Figure 9. We can see that the verb (x1) is identified by an event predicate, while the agent (x4) and patient (x3) are also found. One particular discriminating feature of the imperative sentences that we see, is that the agent has no representation in the sentence. We can also find the preposition (x2) attached to the verb, and this becomes an additional argument for the function.

This logical form also extracts conditions that would be found in if statements and loops very well. Figure 10 shows the DRSs for the sentence: If num is -1, quit. We can see the proposition DRS (the middle box) and the proposition itself (x2), which entails another verb (x3) to be interpreted. That is, we should carry out the verb quit (x1), if the proposition is true. Almost all if statements in the corpus are identified in this way.

9 Generation

Having extracted the functional verb and its arguments, we then need to find a mapping onto an equivalent line of code. The simplest technique, which the current system uses, consists of a list of primitives, each of which describes the specific verb in question as well as a number of arguments. If the semantic information matches perfectly with a

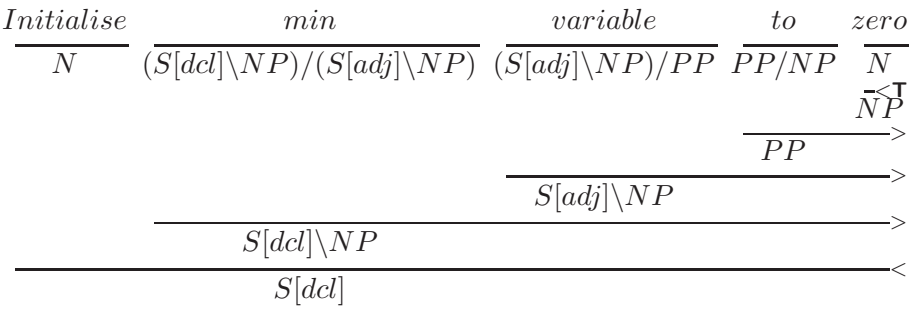


Figure 7: The original, incorrect CCG derivation

TRAINING DATA	COVERAGE	POS WORD	SUPER WORD	POS LINE	SUPER LINE
Original model	95.6757%	0.873	0.736	0.359	0.197
1x code	93.2432%	0.975	0.848	0.829	0.510
5x code	85.9459%	0.994	0.931	0.962	0.708
10x code	82.4324%	0.996	0.962	0.975	0.821
20x code	82.7027%	0.998	0.978	0.986	0.889
20x code, 10-fold cross validation	85.40542%	0.974	0.896	0.840	0.624

Figure 8: Parser results

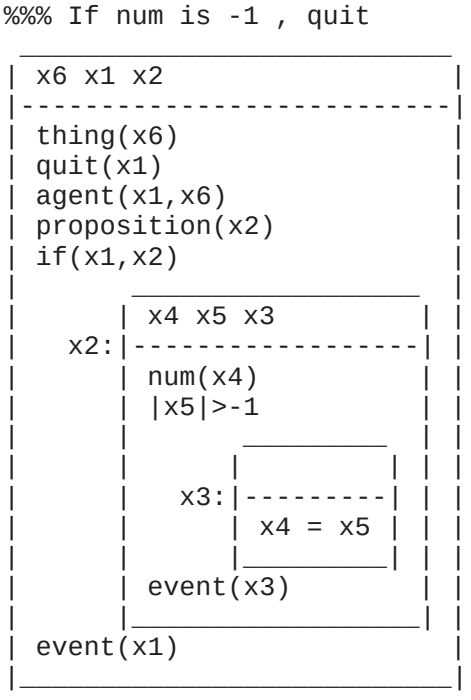


Figure 10: DRS for if statement

primitive, then the equivalent code is generated. At present, there exist only a few primitives, shown in Figure 11. These primitives are made up of the functional verb, the list of arguments they take, and a code template that the arguments are used in.

This system obviously has a number of weaknesses. Firstly, that if the user chooses a verb that is not listed in a primitive, then no code can be generated. Also, some primitives would be described with the same verb and arguments, but require different code, such as adding two numbers together, compared to adding one number to a list. This is similar to operator overloading, a feature present in a number of programming languages such as C++.

We can make a number of observations that can help us improve this step in the system. Firstly, we can reduce the number of possibilities by looking at the program as a whole, rather than as individual lines. For example, for the second problem mentioned above, if the user had previously declared that the second argument was a list, then we would know which of the two primitive operations was correct. We can also constrain the number of possibilities by using intuitive notions, such as not being able to output a previously unseen variable.

Also, we can take advantage of the limited domain of programming. Rather than trying to list every sense of every verb in the English language together with its equivalent programming concept, we could create a much smaller set of programming primitives, and simply map everything onto one of those. Considering the small number of choices and the constraints mentioned above, this may be possible using a machine learning approach.

Of course, we must consider what to do when a function that is not one of the primitives is referred to. In such a case, and assuming it can be detected, we believe the most sensible thing to do is to ask the user to describe how to carry out such a function, using the more basic primitive functions that already exist. Thus we would allow the creation

FUNCTIONAL VERB	ARGUMENTS	CODE TEMPLATE
read	input	<input> = int(sys.stdin.readline())
print	output	print <output>
add	addAmount, addTo	<addTo> += <addAmount>
initialise	variable, setting	<variable> = <setting>
set	variable, setting	<variable> = <setting>
assign	variable, setting	<variable> = <setting>
iterate	item, list	for <item> in <list>:

Figure 11: Primitives used for generation

of user-defined functions, just as a normal programming language would.

Looking back to our example sentence once more, we proceed to extract the predicate (initialise) and argument information (min variable, 0) from the DRS. This maps to the initialise primitive in Figure 11. The matching code, stored in the primitive, then comes out as:

```
min_variable = 0
```

This is clearly a suitable outcome, and we can say that for this case, the system has worked perfectly.

10 Future Work

There is a great deal of work still to be done, before we will have constructed a usable system. We intend to progress initially by expanding the generation component to be able to process most of the commands contained in the English code corpus. We also expect to do more work with the parser and semantic engine. For example, if we find that the coverage or accuracy of the parser is insufficient, then we can create more data for our corpus, or design specialised features to help disambiguate certain word types. Similarly, we may find that some information is not relevant or simply missing from the DRSs that are currently produced, in which case we would be required to extend the current system so that it can extract what is needed.

Once the three basic components described above function at a satisfactory level, then we will begin work on other components of the system. The largest of these is a dialogue component, which should solve a wide range of problems. These could include simple questions about the parse for a sentence:

> *Blerg the number*
```
Is Blerg a verb?
```

It could also help in resolving some ambiguity, or inquire about some missing information.

> *Read in 2 numbers*
> *Add 2 to the number*
```
Which number do you mean?
```

> *Open a file for reading*
```
What is the name of the file?
```

Anaphora resolution is another problem frequently encountered in the English code corpus we have collected. As discovered previously in the case of NLIDBs, having a system capable of dealing with this phenonemon makes it a great deal easier to use. For this reason, we intend to implement such a component for our final system.

Lastly, we intend to develop a GUI that allows a user to interact more easily with the system. Integrating the syntactic, semantic and generation components, together with a text editor, would allow the system to highlight certain functions and arguments. This would make it clearer to the user what the system is doing. The dialogue component in particular would gain a great deal from this, as it could be made clear what sentence or word was being clarified, as well as the context it was in.

11 Conclusion

Programming is a very complicated task, and any way in which it can be simplified will be of great benefit. The system we have outlined and prototyped aims to allow a user describe their instructions in a natural language. For this, a user may be asked to clarify or rephrase a number of points, but will not have to corrrect syntax errors as when using a normal programming language.

Using modern parsing techniques, and a better understanding of just how programmers would write English code, we have built a prototype that is capable of translating natural language input to working code. More complicated sentences that describe typical programming structures, such as if statements and loops, are also understood. Indeed, much of the work to be done involves increasing the coverage of the system in a general manner, so that it is able to understand a wider variety of user input. Once we have built a complete system that can make

some understanding of almost any input, we expect it to be usable by novice and experienced programmers alike.

Acknowledgements

We would like to thank members of the Language Technology Research Group and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81.

A. Biermann, B. Ballard, and A. Sigmon. 1983. An experimental study of natural language programming. *International Journal of Man-Machine Studies*, 18:71–87.

P. Blackburn and J. Bos. 2005. *Representation and Inference for Natural Language. A First Course in Computational Semantics.* CSLI Publications.

J. Bos, S. Clark, M. Steedman, J.R. Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, Geneva, Switzerland.

S. Clark and J.R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, Barcelona, Spain.

S. Clark, M. Steedman, and J.R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the SIGDAT Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 111–118, Barcelona, Spain.

S.S. Epstein. 1985. Transportable natural language processing through simplicity – the PRE system. *ACM Transactions on Office Information Systems*, 3:107–120.

R. Frost and J. Launchbury. 1989. Constructing natural language interpreters in a lazy functional language. *The Computer Journal. Special issue on lazy functional programming*, 32(2):108–121, April.

N. E. Fuchs and R. Schwitter. 1996. Attempto controlled English (ACE). In *Proceedings of the First International Workshop on Controlled Language Applications*, pages 124–136.

G. E. Heidorn. 1976. Automatic programming though natural language dialogue: A survey. *IBM Journal of Research and Development*, 20(4):302–313, July.

H. Liu and H. Lieberman. 2004. Toward a programmatic semantics of natural language. In *Proceedings of VL/HCC'04: the 20th IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 281–282, Rome, September.

H. Liu and H. Lieberman. 2005. Metafor: Visualizing stories as code. In *Proceedings of the ACM International Conference on Intelligent User Interfaces*, pages 305–307, San Diego, CA, USA, January.

J.F. Pane, C.A. Ratanamahatana, and B.A. Myers. 2001. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54(2):237–264, February.

J.F. Pane, B.A. Myers, and L.B. Miller. 2002. Using hci techniques to design a more usable programming system. In *Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC 2002)*, pages 198–206, Arlington, VA, September.

D. Price, E. Riloff, J. Zachary, and B. Harvey. 2000. NaturalJava: A natural language interface for programming in Java. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, pages 207–211.

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

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

Identifying FrameNet Frames for Verbs from a Real-Text Corpus

Matthew HONNIBAL and **Tobias HAWKER**
Language Technology Research Group
School of Information Technologies
Madsen Building (F09)
University of Sydney
NSW 2006, Australia
{mhonn,toby}@it.usyd.edu.au

Abstract

Previous systems that automatically tag text with FrameNet labels have been trained from the FrameNet example data, as there is no FrameNet annotated corpus. The FrameNet data is systematically biased by the criteria for the examples' selection, as annotators attempt to select simple sentences that include the target word.

Instead of using the FrameNet examples, we train a maximum entropy model classifier to identify verb frames on text from the Penn Treebank. We use examples of verbs with only one entry in FrameNet as training data, and evaluate the system on human annotated text from the Wall Street Journal. We accurately identify the frame used by 76% of finite verbs.

We also investigate how well the system performs on verbs it has not encountered before. This task examines the feasibility of using the system to automatically extend the coverage of FrameNet by classifying verbs with no FrameNet entries. The classifier accurately assigns a frame to 55% of instances of verbs it has not been trained on.

1 Introduction

FrameNet (Ruppenhofer et al., 2005) is a lexical semantic database that categorises words into frames, and gives extensive examples of their use. A frame records the semantic type of a predicate and the semantic roles of its arguments. Usually, the predicate will be a verb, and its roles will be realised by constituents of its clause.

Recently, Senseval-3 (Litowski, 2004) used FrameNet as the basis of a semantic role labelling shared task. Participants were asked to replicate the role labelling of frame elements in the FrameNet example data given the frame. However, a system which is trained and evaluated on the FrameNet examples may not perform comparably on other text. FrameNet examples are selected by the annotators to illustrate the semantic and syntactic combinatory possibilities of each word, with minimal confusion from irrelevant performance variables and complicated syntactic constructions. They are therefore a systematically skewed sample, with much of the complexity of natural text under-represented or missing entirely.

The problem is that there is currently no FrameNet annotated corpus — only four manually annotated Wall Street Journal texts, recently released on the FrameNet website. There is, however, still a way to train a system for a subset of the FrameNet annotation task on real text without one. Frame selection is often unambiguous given a particular verb: currently, 73% of verbs entered in FrameNet are associated with only one frame. These verbs head 40% of the finite clauses in the Penn Treebank (Marcus et al., 1993).

The verb senses associated with a particular FrameNet frame share similar semantics and argument structures. For instance, the Activity_finish frame includes the unambiguous verbs *finish* and *complete*, and the ambiguous verb *conclude*, which is also associated with the Coming_to_believe frame.

Clearly, these two types of *conclude* are different word senses, and the Activity_finish sense should be closer to the other Activity_finish verbs than the Coming_to_believe sense in a sense taxonomy like WordNet (Fellbaum, 1998). The two frames also have different argument structures: the Activity_finish verbs are all transitive, and would usually require a conscious agent and an event noun object. The Coming_to_believe frame verbs, such as *ascertain* and *deduce*, expect a sentiential complement instead of a noun phrase object. Such patterns can be learnt even if the learner has access to examples using only a few verbs of each frame.

We train a maximum entropy model on the unambiguous clauses in the Penn Treebank, and evaluate it on two tasks. First, the FrameNet website has recently added four human annotated texts from the Wall Street Journal, which we use as test data. Second, we hold out the instances of one quarter of the verbs in FrameNet, in order to evaluate how

Proceedings of the Australasian Language Technology Workshop 2005, pages 200–206,
Sydney, Australia, December 2005.

well the classifier can deal with unknown words. FrameNet's lexical coverage is currently quite low, so a system which can accurately assign frames to currently unclassified words could be used to automatically extend FrameNet's vocabulary.

2 Labelling FrameNet predicates

FrameNet is a lexical semantic database that records the semantic type of a predicate and the semantic roles of its arguments, as well as how they are realised syntactically. The predicates are usually verbs, but can also be nouns, adjectives, adverbs or prepositions. The database is organised into *frames*, which group predicates that share similar semantics and argument structures.

Full FrameNet annotation involves labelling each predicate with a frame, and then allocating role labels to the constituents that realise its frame elements. Verb frame elements are usually realised by direct arguments of the predicate, such as its subject and object.

The seminal work on automatic FrameNet annotation, Gildea and Jurafsky (2002), and the Senseval-3 task (Litowski, 2004) after it, both concentrate on labelling frame elements given a frame label. This semantic role labelling task is a sequence tagging problem, a little like chunk tagging, in terms of its sequantiality and boundary detection.

We are not aware of any other attempts to identify frame labels, rather than frame elements. This problem is rather different, as there will be exactly one label per frame. We therefore assign labels to each clause independently, based on the semantics of the verb and the semantics of its syntactic arguments. Section 5 discusses how we capture this information in our features.

3 FrameNet and WordNet

The current problem with using FrameNet as the basis of a semantic parser is its coverage. The latest release of FrameNet, 1.2, has approximately 6,765 lexical entries, covering only 64% of tokens (and 26% of token types) in the Penn Treebank. Another potential issue is that annotation is proceeding frame by frame, rather than word by word. This means that there is no guarantee that existing lexical entries are complete. For instance, the verb *occur* belongs only to the Event frame; its cognition sense, as in *'the idea never occurred to me'*, is undocumented.

Both of the problems noted above could be corrected — or at least, alleviated — by mapping FrameNet entries to WordNet senses. WordNet (Fellbaum, 1998) is a lexical database that focuses on semantic relations between synonym sets, such as hyponymy and meronymy, and exhaustively cataloguing all senses of its entries. Unlike FrameNet, it does not include much detail on argument structure or type. The two resources are therefore complementary.

Shi and Mihalcea (2005) argue that mapping the lexical entries in FrameNet to WordNet senses via VerbNet (Kipper et al., 2000) is a promising approach to connecting these complementary resources. They map the 2,393 verb intersection of these three resources, and additionally map another 839 verbs by generalising with WordNet synonymy and hyponymy. This is a relatively small subset of the 11,488 verbs entered in WordNet, and only increases the size of FrameNet a little.

However, the mapping does reduce frame ambiguity a little in WordNet sense disambiguated text — of which there is a reasonably sized corpus. We find that using the mapping Shi and Mihalcea publish allows us to increase the diversity of verbs in our training data, which consists of examples of unambiguous verbs, thereby increasing performance.

4 Training Data

We use examples of a limited set of verbs — the unambiguous ones — to represent each frame. This allows us to train the system without annotated data. This approximation rests on several assumptions, none of which are quite correct. We take a few simple steps to alleviate the problems they introduce.

First, we assume that the similarities between the verbs in each frame are strong enough that an example of one verb is a reasonable surrogate for an example of a different verb. In the best case scenario, the two verb senses will have identical semantic and syntactic profiles: those senses will be interchangeable in all contexts, but one verb will be polysemous, and the other monosemous.

Second, we assume that using only the unambiguous instances will not radically change the distribution of the classes in the training data. In the worst case scenario, some frames will have no monosemous verbs from which to learn, so some of the classes will be missing entirely in the training data. The distribution may also be skewed when a frame has a particularly common unambiguous verb, over-representing it significantly. For instance, the verb *be* is associated with only one frame, Performers_and_roles. The problem is exacerbated by the fact that polysemy is correlated with the frequency of the word, so most other frequent verbs will be ambiguous.

Of course, *be* is not actually monosemous: in

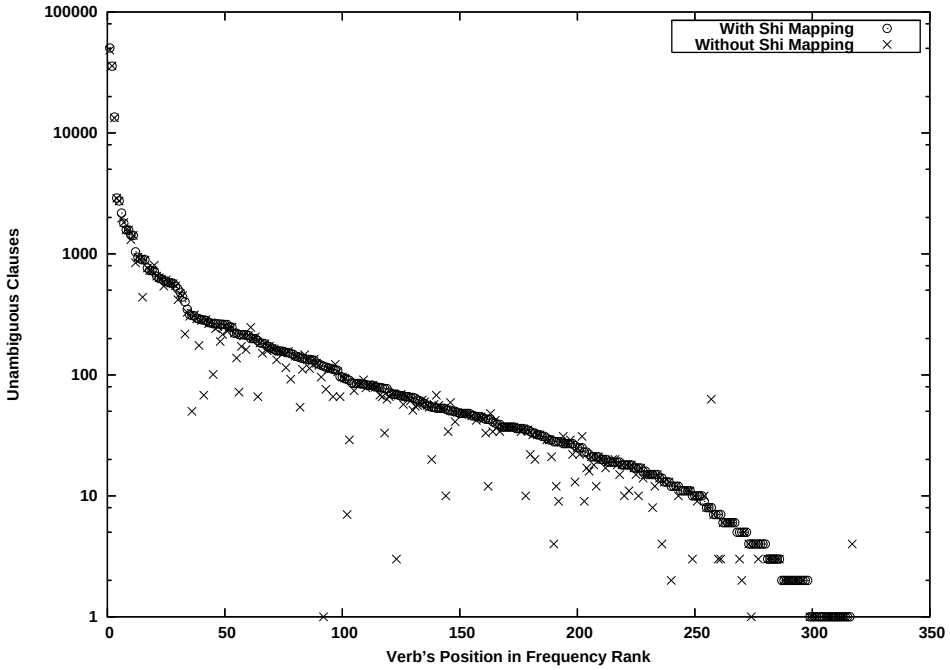


Figure 1: Training data available with and without Shi & Mihalcea's mapping

WordNet, it has thirteen senses. We assume that the lexical entries in FrameNet are complete; that when a verb has only one frame associated with it, there is only one frame it can use. In reality, FrameNet is a work in progress, and it is proceeding frame-by-frame, rather than word-by-word. The annotators add a frame, populate it with some lexemes, find examples for them, and describe its semantics and relations to other frames. They do not ensure that a given word is associated with every frame it can use.

Of the three assumptions, this last is the most troubling — as it makes it difficult to know what the answer is, not just how it can be predicted. We alleviate the other two problems with two measures that are both designed to increase the variety of verbs we can train from.

First, we use the FrameNet-WordNet mapping released by Shi and Mihalcea (2005) and use the SemCor corpus (Miller et al., 1993), which is a WordNet sense disambiguated portion of the Brown corpus (Francis, 1964) as part of our training data. So our training data includes the WordNet sense of the verb, and Shi and Mihalcea's map translates that into a single frame label. We can then use instances of that verb as training data, where before it may have been ambiguous. Figure 1 shows how many clauses in the Penn Treebank are available for training for each class with and without Shi and Mihalcea's mapping.

The second way we try to correct for the under-representation of classes in the training data is to use an iterative bootstrapping system. We first train a classifier on the unambiguous instances, and then use it to predict a class for those whose frame we are unsure of. When the predicted class is within the possibilities listed in FrameNet (if there are any), and the confidence of the prediction is above a threshold C, we add the instance to the training data for the next iteration. Figure 2 shows how many instances are drawn into the training data at each iteration when C is set to 0.4. Clearly, the number levels off, so we limit it to six iterations.

5 Feature extraction and selection

A frame is defined by the semantic type of its predicate and the semantic roles of its arguments. FrameNet does not allow contextual elements to realise semantic roles. Instead, semantic roles must be realised by a syntactically local constituent, or be considered missing ("null instantiation").

The classifier therefore must have access to some representation of the semantic type of each constituent in the predicate's clause, as some of these will realise the semantic roles which partly define the frame. Representing the semantics of a whole constituent is difficult, so we restrict ourselves to using head words. For instance, if the subject of a clause is *'those two old electric trains'*, we will attempt to represent the semantics of *'trains'*. Sometimes, the grammatical head is not the semantic head, particularly in *'of'* expressions like *'a glass*

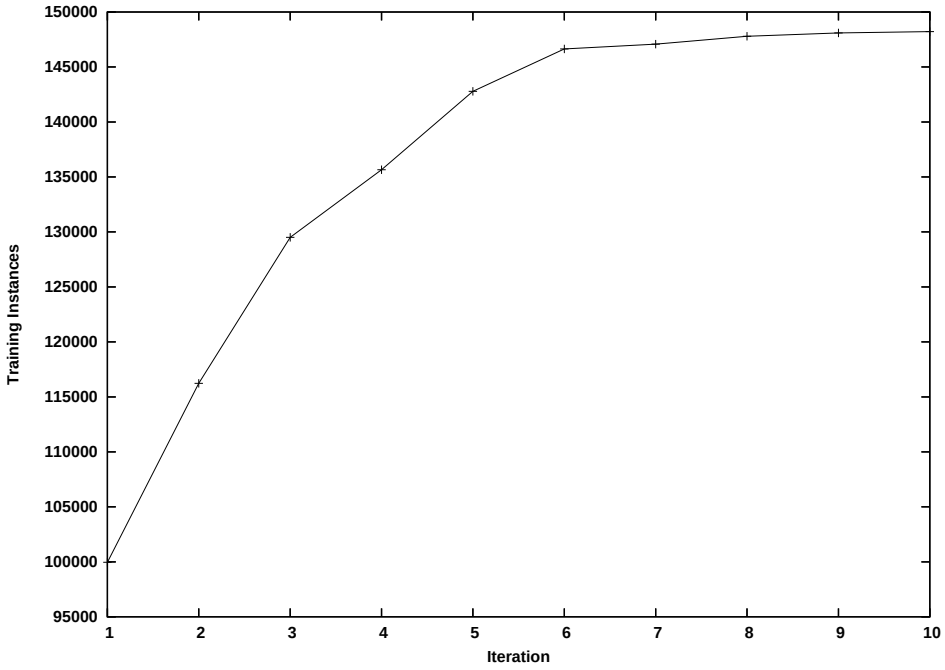


Figure 2: Training instances available per iteration (bootstrapping process)

of wine'. This type of construction is difficult to reliably distinguish from uses of *'of'* like *'a glass of the finest crystal'*, so we simply accept the imprecission. We do, however, consider the head of prepositional phrases to be the head of their component noun phrase, instead of the preposition. We attempt to model the semantics of the following clause constituents:

- Main verb
- Logical subject
- Logical object
- Indirect object
- Prepositional phrases
- Adverbial phrases

We represent the semantics of a word using its WordNet synset and hypernyms. The feature space consists of a list of WordNet synsets, whose value is initially the probability that a given constituent's head is either a member or hyponym of that synset.

Probabilities allow us to use non-word sense disambiguated data, by distributing the probability (or in Bayesian terms, belief) mass amongst the possible senses of the word. It would be possible to use a sophisticated word sense disambiguation system to arrive at these belief scores, but we use a naive method which simply weights the distribution by the senses' frequency ranks, such that the most frequent sense has twice the belief strength of the least frequent sense. Hypernym belief scores are the sum of the belief scores of its hyponymys, as shown in figure 3. So even if a word is sense-ambiguous, it can have an unambiguous synset amongst its hypernyms, if its senses have one in common. The classifier will therefore favour features corresponding to more general synsets, because they will usually occur more frequently and have higher belief scores.

Each constituent type listed above controls its own section of the feature space, as shown in table 1. There are 152,059 unique synsets in WordNet, but 335,928 possible features in our feature space. This is because the same noun synset might be four distinct features, depending on whether it occurs in the logical subject, logical object, indirect object or a prepositional phrase. This replication preserves the distinction between the different grammatical functions, so that a clause with an animate subject and an inanimate object is represented differently from a clause with an inanimate subject and an animate patient. Compare *'he considered his options'* with *'the current swept him away'*.

Of course, in the examples above, the word *'he'* will have no entry in WordNet at all. Animacy is one of the most important lexical semantic distinctions, and animate constituents are realised most commonly by pronouns and proper nouns. To correct this problem, we supply the "person" synset for variants of *'he'* and *'she'*, as well as *'who'*. We con-

Syntactic Role	Synset Count
Verb	13,508
Log. Subj.	79,689
Dir. Obj.	79,689
Ind. Obj.	79,689
PP Head	79,689
Adverb	3,664
Total	335,928

Table 1: Potential Feature Space Size

sider proper nouns, as distinguished by their Treebank part of speech, to be polysemous: it is not immediately obvious whether they stand for a person, organisation, country, etc. We therefore supply a small group of general synsets ("person", "organisation", "place") to represent them.

We reduce the size and complexity of our feature space by discretising each feature into a boolean value, and selecting features with the top *N* information gain. The maximum entropy implementation we are using, Zhang Le's Maxent Toolkit (Le, 2005), does allow real valued features, but they are nevertheless unideal with our data. Real valued features with non-uniform distributions may interfere with the parameter estimation algorithm, so we discretise them into boolean features at the threshold which maximises their information gain. We then use the information gain scores to rank the features and select the *N* best, thereby reducing the feature space.

6 Evaluating on unseen verbs

We paid particular attention to how the system would perform on verbs it had not been trained with, to explore how well it can deal with unknown words. Paying individual attention to unknown words is a familiar strategy in tagging problems, but to our knowledge has not been applied to FrameNet labelling tasks.

There are two reasons why unknown words are even more important for a FrameNet tagger than, for example, a part-of-speech tagger. First, the vocabulary of FrameNet is quite small, so a tagger running on natural text will encounter plenty of them. In the Penn Treebank (Marcus et al., 1993), only 64% of tokens and 26% of token types have an entry in FrameNet. Second, assigning labels to previously unclassified words could be used to extend FrameNet's vocabulary semi-automatically, by manually correcting the tagger's suggestions.

To evaluate performance on unknown verbs, we set aside instances of 25% of the verbs in the training set. This is a tough evaluation measure. First, we already struggle to represent each frame adequately, as section 4 discusses. This problem is exacerbated by holding out a set of the verbs, since the test set may contain a disproportionate amount of the verbs associated with a particular frame, leaving few or no instances of that frame in the training data. Stratifying the held out set so that it contains roughly 25% of the instances, 25% of the verbs, and maintains a similar distribution of frames to the training data has proven difficult.

	Unseen verbs	WSJ Data
SemCor Shi	47%	62%
PTB Shi	55%	76%
SemCor no Shi	44%	52%
PTB no Shi	48%	67%
Baseline	27%	40%

Table 2: Results

To alleviate the problem, we allow the early iterations of the classifier access to the test data, only removing it for the final classifier. This means that the final classifier has not had access to the test data, but the system has the opportunity to draw in examples similar to it from the unclassified data during the bootstrapping process.

For instance, if all of the unambiguous examples of the Performers_and_roles frame use the verb *be*, and *be* is a test data verb, the training data will contain no examples from that class, and consequently get every example of *be* wrong. So we at first allow the system to train on examples of *be*, so that it can classify ambiguous examples of Performers_and_roles, such as instances of the verb *play*. Finally, instances of *be* are removed before the final iteration, so that the final classifier has some instances of Performers_and_roles, from verbs like *play*, but has not been trained on instances of *be*. The classifiers for the earlier iterations just need to be as accurate as possible. We do not need to evaluate them, so we can provide them the test data to build a better model. However, we do need to evaluate the final classifier, so we cannot train it with the test data.

7 Results

Table 2 presents results from different datasets on our two classification tasks, with and without using the mapping described by Shi and Mihalcea (2005). The "WSJ Data" task evaluates our classifier on four human annotated Wall Street Journal texts made available on the FrameNet website. This quantity of text is obviously insufficient for training, but does make a good test set, after the texts have been set aside from the rest of the WSJ training data.

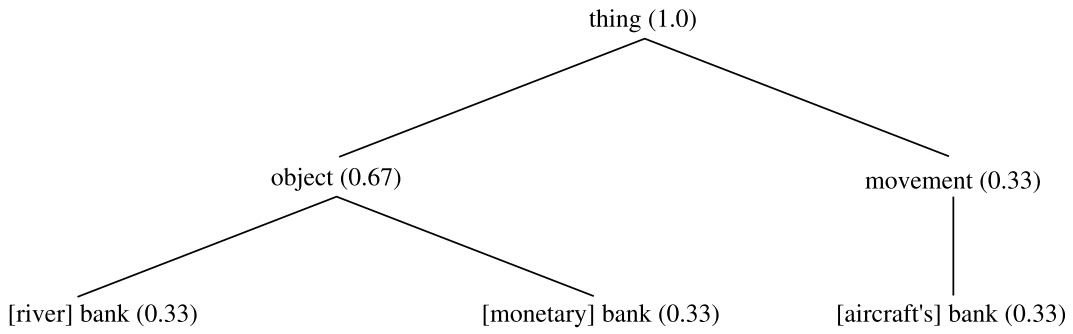


Figure 3: Belief propagation to hypernyms

The "Unseen verbs" task is the classification of instances of verbs the classifier has not been trained with, as discussed in section 6 above. Assigning the most common frame to all examples gives an accuracy of 27%, which we use as the baseline in table 2.

As discussed in section 4, using the WordNet mapping improves the balance of the training data by making more verbs available for training, so it is not surprising that there is such a clear improvement in performance.

The other obvious result is that performance is better when the classifier is trained on the whole Penn Treebank, instead of simply the SemCor subset. This indicates that on these tasks, it is worth having more data even at the cost of more ambiguous feature values. The disparity between the two datasets is much greater on the Wall Street Journal test data than on the classification of unseen verbs. We believe this is because the Penn Treebank training data includes the rest of the Wall Street Journal text, while our SemCor data is limited to the intersection of the SemCor text and the parsed Brown section from the Penn Treebank. This intersection is 31,456 clauses, and contains no newswire text. The Penn Treebank does not include skeletal parses of the newswire sections of the Brown corpus because the Wall Street Journal data amply represents that genre.

The performance on unseen verbs is relatively poor, but nevertheless encouraging. Even noisy prediction on this task may reduce the time required for manual extension of FrameNet's vocabulary. This is especially true if only verbs classified with especially high confidence are considered as candidates for addition.

The main result we report is the 76% accuracy identifying verb frames in real text. While there are no directly comparable results reported in the literature, this result considerably outperforms the trivial method of simply assigning frames to verbs when they are unambiguous, which gives 40% accuracy. We use this figure as the baseline for comparison.

8 Related Work

This research falls under the broad umbrella of lexical semantics acquisition, such as the work of Pantel and Ravichandran (2004) or Fouvry (2003). The most pertinent research of this type to our system comes from the SALSA project(Erk et al., 2003), which is creating a German frame lexicon and annotated corpus. In particular, Erk (2005) uses complement and adjunct heads (in addition to bigram and trigrams) as features in a naive Bayes classifier to assign frame labels to verbs. She reports a result of 74% on this task for the SALSA lexicon. Because the classification scheme — and indeed, its language — is different from ours, this result is not directly comparable with our own; however, Erk does compare baseline scores for both the SALSA projecttask and the equivalent FrameNet task, finding that the SALSA task's baseline is considerably lower: assigning each verb its most frequent frame yields 93% accuracy for the labelling task on the FrameNet example corpus, and only 69% on the SALSA corpus. This reflects the fact that the SALSA lexicon is more polysemous than the current release of FrameNet: 4.12 for SALSA, 2.27 for FrameNet. This baseline would be quite meaningless for our task, as we are testing on unambiguous verbs — so our baseline according to Erk's method would be 100%.

9 Conclusion

We have reported two results on a novel kind of task: the assignment of frame labels to text which was not manually selected for FrameNet annotation. Despite the lack of obvious training data, our system achieves 76% accuracy, 55% when the verb is new.

These results support the argument made by Shi

and Mihalcea (2005): a comprehensive mapping between WordNet and FrameNet is both possible and desirable. A comprehensive mapping will immediately make available a corpus of frame labelled material, in the form of the SemCor corpus. This text will also be annotated with syntactic bracketing and WordNet senses. A useful task for future work is to investigate how the results from our classifier might be used to produce such a mapping semi-automatically. High confidence misclassifications may also be able to reveal situations where a word is missing from one or more frames.

For now, we demonstrate that the FrameNet classification scheme is at least robust enough to perform on data that was not used to design it.

10 Acknowledgements

We would like to thank Zhang Le for making the Maximum Entropy toolkit publicly available. Our thanks also go to Shi and Mihalcea for releasing the list of mappings their system produces between WordNet and FrameNet. Finally we would like to thank Jon Patrick and James Curran from the University of Sydney for their invaluable insights.

References

K. Erk. 2005. Frame assignment as word sense disambiguation. In *Proceedings of IWCS*.

K. Erk, A. Kowalski, S. Padó, and M. Pinkal. 2003. Towards a resource for lexical semantics: A large german corpus with extensive semantic annotation. In *Proceedings of ACL-03*.

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

Frederik Fouvry. 2003. Lexicon acquisition with a large-coverage unification-based grammar. In *Proceedings of the 10th Conference of the EACL (EACL 2003)*.

W. Francis. 1964. A standard sample of present-day english for use with digital computers. Report to the U.S Office of Education on Cooperative Research Project No. E–007. Brown University, Providence.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-2000)*. Austin, Texas.

Zhang Le. 2005. Maximum entropy modeling toolkit for python and c++. http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Kenneth C. Litowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 9–12.

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

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

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of HLT/NAACL-04*, pages 321–328.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, and Christopher R. Johnson. 2005. *FrameNet: Theory and Practice*. Online, available at http://framenet.icsi.berkeley.edu.

Lei Shi and Rada Mihalcea. 2005. Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. In A. Gelbukh, editor, *CICLing 2005*, pages 100–111.

A Distributed Architecture for Interactive Parse Annotation

Baden Hughes
Department of Computer Science
and Software Engineering
The University of Melbourne
Victoria 3010, Australia
badenh@csse.unimelb.edu.au

James Haggerty, Saritha Manickam
Joel Nothman, **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{j.h,jnothman,saritha}@student.usyd.edu.au
james@it.usyd.edu.au

Abstract

In this paper we describe a modular system architecture for distributed parse annotation using *interactive correction*. This involves interactively adding constraints to an existing parse until the returned parse is correct. Using a *mixed initiative approach*, human annotators interact live with distributed CCG parser servers through an annotation GUI. The examples presented to each annotator are selected by an *active learning* framework to maximise the value of the annotated corpus for machine learners. We report on an initial implementation based on a distributed workflow architecture.

1 Introduction

Annotating sentences with parse trees is perhaps the most complex and intensive linguistic annotation. The time and expense of developing parsed corpora is almost prohibitive. As a result there are only a small number of such corpora, including the Penn Treebank (Marcus et al., 1994), the German TiGer Corpus (Skut et al., 1997) and more recently the LinGO Redwoods Treebank (Oepen et al., 2002). These corpora are also limited in size, typically around one million words.

Unfortunately, the statistical approaches to parsing which have been most successful rely heavily on both the quality and quantity of annotated resources. Also, these approaches are very sensitive to the statistical properties of the corpus, and so a parser trained on one genre may perform badly on another (Gildea, 2001).

Another major problem with parsed corpora is that they must, at least to some extent, follow a particular syntactic theory or formalism. This is a major difficulty for two reasons: firstly, it means we need separate annotated corpora for each formalism; and secondly, it means that comparing parser evaluations across formalisms is difficult.

Fully automated conversion of parse trees between formalisms is difficult because each analyses certain constructs in idiosyncratic ways. An example is CCGbank (Hockenmaier and Steedman, 2002), a treebank of Combinatory Categorial Grammar (Steedman, 2000) derivations which were converted semi-automatically from the Penn Treebank trees. The result still required laborious editing to produce idiomatic CCG derivations (Hockenmaier, 2003).

We intend to create a new corpus of CCG derivations on a wide range of text. We face three key problems: 1) selecting sentence to annotate which creates the most *useful* corpus for statistical parsers. 2) maximising the *annotator efficiency* and *minimising error*; 3) allowing distributed annotators to *share expertise*.

The selection problem is addressed using *active learning* (AL). Active learning involves computing which training instances provide the most new information to one (or more) machine learners (Cohn et al., 1995; Dagan and Engelson, 1995). The annotators become oracles answering specific queries posed by the learners.

The annotation problem is addressed by *interactive correction* of the output of our statistical CCG parser. This is similar to the *discriminant* strategy employed for Redwoods annotation (Oepen et al., 2002) but generalises to grammars where parse enumeration is infeasible. Annotators interactively add constraints to the parser which returns the most probable parse satisfying the constraints.

The distributed expertise problem is addressed using a *workflow manager*. Annotators will be able to add comments and queries to derivations and have them sent to (potentially remote) experienced annotators for verification. The workflow manager will also handle scheduling for the active learning infrastructure.

This paper describes the architecture and initial implementation of a system which addresses these problems for distributed parse annotation.

Proceedings of the Australasian Language Technology Workshop 2005, pages 207–214,
Sydney, Australia, December 2005.

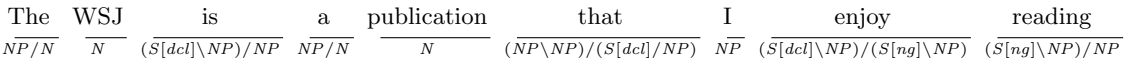


Figure 1: Example sentence with CCG lexical categories

2 Related Work

Since annotating parse trees is a significant bottleneck in NLP there have been several attempts to make the process more efficient. In this work we exploit two approaches: using choice points to select the correct parse and using active learning to select sentences to parse.

A *discriminant* is a property that distinguishes between a set of interpretations. They can be designed for linguistic non-experts (Carter, 1997). In the Redwoods project, the annotator is presented with discriminants on the trees themselves which eventually lead to the correct HPSG parse (Oepen et al., 2002). These discriminants are calculated from the enumerated set of all parses. Unfortunately, our automatically extracted CCG grammar produces far too many derivations (billions) for enumeration to be feasible.

Baldridge and Osborne (2004) demonstrate how active learning (AL) can be used to significantly reduce the annotation cost for annotating text with HPSG parses. They compare random selection with approaches based on uncertainty sampling (Cohn et al., 1995) and committee based sampling (Dagan and Engelson, 1995) and demonstrate a reduction in annotation effort of 72%. A key point that Baldridge and Osborne identify is that each sentence cannot be treated as equally difficult to annotate. Tang et al. (2002) also evaluate AL on statistical parsing and find the total cost of annotation can be reduced to one third. Finally, Becker et al. (2005) compares bootstrapping techniques including AL for developing new named entity corpora.

Formally introduced in Day et al. (1997), *mixed initiative annotation* (where the division of labour between computational facilities and human effort is coordinated for increased efficiency) has become an increasingly common methodology for the preparation of large corpora. Typically however, mixed initiative approaches have largely decoupled human and machine effort, even for larger scale tasks.

Extending the mixed initiative model specifically to a distributed environment, Hughes and Bird (2003) offer a model for the type of solution we implement here. Additionally, the architecture advocated by Curran (2003) allows us flexibility in designing individual components of this system independently, and then marshalling them into a single application instance.

Experiments with distributed NLP tasks of building n-gram language models (Hughes et al., 2004a) and generalised textual indexing and linguistically motivated retrieval (Hughes et al., 2004b) are broadly indicative of other work in this area. To date, however we are not aware of any work in this vein specifically involving mixed initiative annotation.

3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2000) is a type-driven lexicalized theory of grammar based on categorial grammar. Almost all of the grammatical information in CCG is represented in the categories assigned to each word, which are either simple atomic categories (e.g. NP) or complex functor categories (e.g. $(S[dcl]\backslash NP)/NP$ a transitive declarative verb). An example sentence with lexical categories is shown in Figure 1. These categories are combined together according to a small number of *combinatory rules*.

The set of these lexical categories is obtained from CCGbank (Hockenmaier and Steedman, 2002; Hockenmaier, 2003), a corpus of CCG normal-form derivations derived semi-automatically from the Penn Treebank. The category set consists of those category types which occur at least 10 times in sections 2-21 of CCGbank, which results in a set of 409 categories. Clark and Curran (2004a) demonstrates that this relatively small set has high coverage on unseen data and can be used to create a robust and accurate parser. In order to obtain semantic representations for a particular formalism, only 409 categories have to be annotated.

3.1 CCG Parsing

In our system we are using the C&C CCG parser (Clark and Curran, 2004b), which uses a log-linear model over normal-form derivations to select an analysis. The parser takes a POS tagged sentence as input with a set of one ore more categories assigned to each word. A CCG supertagger (Clark and Curran, 2004a) assigns the lexi-

cal categories, using a log-linear model to identify the most probable categories. Clark and Curran (2004a) show how dynamic use of the supertagger — starting off with a small number of categories assigned to each word and gradually increasing the number until an analysis is found — can lead to a highly efficient and robust parser.

The parser uses the CKY chart-parsing algorithm from Steedman (2000). The combinatory rules used by the parser are functional application (forward and backward), generalised forward composition, backward composition, generalised backward-crossed composition, and type raising. There is also a coordination rule which conjoins categories of the same type.

3.2 CCG Supertagging

Lexicalised grammar formalisms, such as LTAG and CCG, assign one or more syntactic structures to each word which are then manipulated by the parser. Supertagging was introduced for LTAG to increase parsing efficiency by reducing the number of structures assigned to each word (Bangalore and Joshi, 1999).

The parser model parameters are estimated using a discriminative method, that is, one which requires statistics across all incorrect parses for a sentence as well as the correct parse. Since an automatically extracted CCG grammar can produce an extremely large number of parses, the use of a supertagger is crucial in limiting the total number of parses for the training data to a computationally manageable number.

The supertagger is also crucial for increasing the speed of the parser. We have shown that spectacular increases in speed can be obtained, without affecting accuracy or coverage, by tightly integrating the supertagger with the CCG grammar and parser (Clark and Curran, 2004a). To achieve maximum speed, the supertagger initially assigns only a small number of CCG categories to each word, and the parser only requests more categories from the supertagger if it cannot provide an analysis. Clark et al. (2004) has demonstrated that annotating new data at just the lexical category level can be enough to significantly improve the performance of a parser on a new domain.

3.3 Interactive Correction

For a given sentence, the automatically extracted grammar can produce a very large number of derivations. Clark and Curran (2004b) describes how a packed chart can be used to efficiently represent the derivation space, and also efficient algorithms for finding the most probable derivation. Unfortunately, this massive derivation space means it is not possible to enumerate all parses, so the discriminant strategy for interactive annotation outlined previous is infeasible.

We therefore introduce the idea of *interactive correction* where the parser is given a number of constraints by the annotator. Rather than enumerate the parse, the process only involves finding the most probable parse that satisfies the expressed constraints. This can be performed efficiently as part of the dynamic programming algorithm which finds the highest probability derivation.

Given that the CCG categories contain so much information we expect that it will only require annotators to constrain the lexical categories on a few words to reach a correct parse.

4 Example Use Case

The annotation process begins by the annotator requesting a sentence to annotate. The active learning component determines which sentence from a large corpus of raw sentences may provide the most *new* information if it were to be annotated. AL can be very computationally intensive process and so will only occur after a given number of sentences have been annotated. The AL component will return a queue of sentences that then scheduled to be annotated.

The annotator will receive the top sentence on the queue along with the most probable derivation for that sentence. They can add the following constraints to a given sentence:

force a specific lexical category
ban a current lexical category
ban a current non-leaf category
force a specific chart span
ban a current chart span

Adding one (or more) new constraints will cause the derivation to be returned which satisfies the existing and new constraints (if such a derivation exists). This process continues until the correct derivation is reached and the parse is checked in as correct. Once enough new annotated sentences have been completed the AL component regenerates a new queue of sentences based on the retrained statistical parser model.

An alternative case is that the annotator is not sure about the correct derivation for the sentence. They can then annotate the derivation

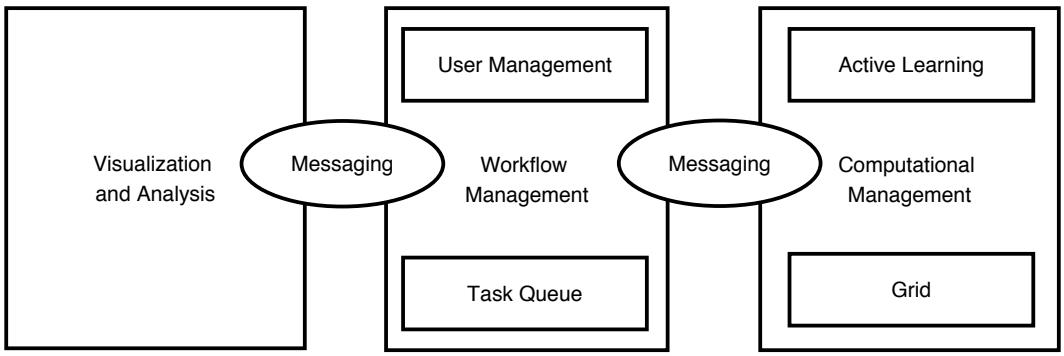


Figure 2: System Architecture

with a comment/question and it will be scheduled on the queue for other more experienced annotators. The experienced annotator will see the constraints and the comments added by the original annotator. They can make a decision or propagate it to some other annotator. Once a decision has been reached the information is returned to all annotators. This process is handled by the workflow manager.

5 System Architecture

The system architecture for distributed annotation and parsing with active learning can be seen in Figure 1. The Visualization and Analysis module provides the end user interface by which a human annotator can review and revise the parser output. The actual content rendered by this module is provided by the Workflow Management module.

The Workflow Management module has three main roles: first to interact with the Visualization and Analysis, providing parses to be visualised and refined; second to manage the user and tasks in the process of analysis; and third to interact with the Computational Management module by instantiating the active learning framework for incremental parsing of the corpus data, and subsequent grid execution.

The Computational Management module has two sub-modules. The Active Learning submodule allows for incremental application of refined parses as training data for subsequent iterations of the parser. The Grid sub-module handles low level execution including the queuing, dispatch and execution of analysis tasks, and fetching the results from the distributed computation environment.

Having described the high level architecture of the system, we now turn to an in depth discussion of each of the components in turn.

6 Visualization and Analysis Module

The visualisation GUI is implemented in wxPython (Dunn, 2005), an extension of the cross-platform GUI toolkit wxWidgets (Smart et al., 2005) for Python. wxWidgets is particularly notable for its use of native graphical components for a given operating system platform, allowing the interface a native look and feel when run on Windows, Mac or Linux environments.

Both the GUI and the export to file functionality are built on a flexible cross-platform code base. As can be seen in Figure 3 an initial implementation already succeeds in displaying the CCG parser output in a user-friendly form similar to that used by (Steedman, 2000) and widely adopted as a standard format.

To facilitate the re-use of the rendered parse tree, our GUI uses the ReportLab Toolkit (Rep, 2005) to facilitate export to common vector graphics formats such as PDF, PostScript, PNG and SVG. These are useful for inclusion in publications and presentations.

By using basically the same rendering code, consistent output is provided to the GUI and the export formats, including plain-text. Each provides a Canvas object whose role is simply to provide metric information for displaying text on that output device, and then to place the text at given locations. All calculations for positioning text nodes are done external to the canvas, so new output formats can easily be supported. Similarly, if a user selects a change in font-styling, this is reflected in all graphical export formats and on-screen.

The graphical interface provides some interactivity in order to assist a user in viewing, manipulating and annotating a parsed sentence Most importantly, an annotator can change or constrain the available categories for sentence

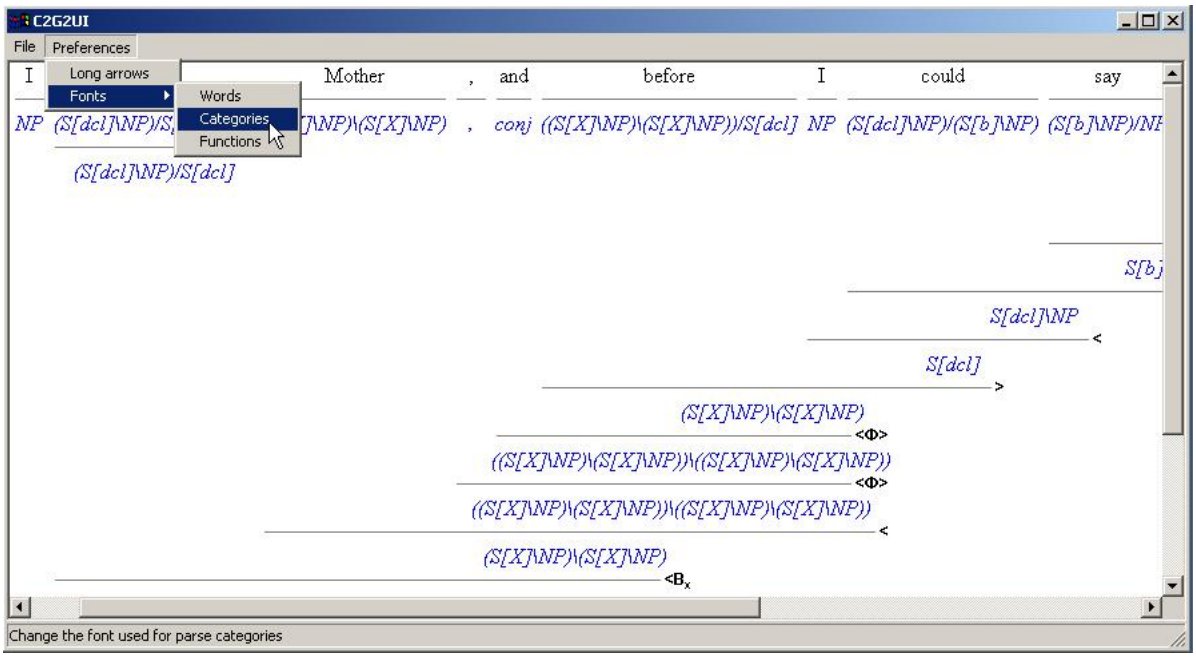


Figure 3: Screenshot from GUI

constituents. As soon as these constraints are added, they are passed back in real-time to the Workflow Management module, and the GUI is updated to reflect the results of those changes. Thus, in almost all cases, correct parses can be generated without the need for the user to laboriously construct an entire derivation.

To simplify the viewing of large parse trees, a user has the facility to 'collapse' chosen sections of the tree with a simple point-and-click operation. Hiding most collapsed words from view and only showing the derived category can significantly reduce the horizontal and vertical space occupied by the parse image. This feature is particularly useful when an annotator needs to focus on a particular section of a given parse: once a certain partial derivation is checked, it may be collapsed and will remain fixed there through other parse modifications.

Collectively these features should make the job of annotators far less painful. The graphical rendering of the CCG parser's output makes the incorrect grouping of words obvious, and by using the parser in collaboration with the user as described above, annotating a sentence correctly will usually be a matter of seconds rather than minutes.

7 Workflow Management Module

The basic workflow of the system is as follows. Sentences are parsed from a corpus by the CCG parser on the grid, results being added to review stack. A user logs in and starts a session, requesting the next review parse. The user reviews parse, and either confirms parse, or modifies and submits a revised parse or promotes the candidate parse to new reviewer. Accepted parses are sent back to the active learner for subsequent retraining of the CCG parser.

As mentioned earlier, the Workflow Management module has two interfaces: one to the Visualization and Analysis module, and the other to the Execution module; as well as an internal user and task management function. We will first discuss the latter, before returning to the interfaces themselves. It is however, important to note that the workflow here is analytical, as distinct from computational.

7.1 User Management Sub-module

The user registry allows for the tracking of user (i.e. annotator) names, together with corresponding passwords and user level attributes. In addition, a log is kept of the activity of each user, in particular annotation times, which are useful for monitoring the effectiveness of the interactive correction approach.

7.2 Task Queue Sub-module

The task queue sub-module is basically a parse review queue contains the list of sentences pending review together with an user allocation, and

a parse/sentence status (pending, reviewed).

7.3 Interface to Visualization and Analysis

To facilitate communication the Simple Object Access Protocol (SOAP) (Gudgin et al., 2003) has been used to implement a lightweight interface. The SOAP implementation supports 5 basic functions:

- `authenticate_user` implements a lightweight user authentication protocol based on a username and password;
- `submit_accepted_parse` is used when the current parse and constraints are acceptable, and is parameterised by the sentence ID, a parse ID, and the set of constraints with any associated commentary.
- `submit_uncertain_parse` is used when the current parse is not fully understood by the user, and a second opinion is required. It is is parameterised by the sentence ID, a parse ID, and the set of constraints with any associated commentary.
- `get_next_parse` is the stack retrieval method, used to retrieve the next parsed sentence in the individual user queue. It returns a the next sentence ID, sentence, parse ID, and set of constraints and associated commentary.
- `get_modified_parse` allows the user to get a subsequent sentence and parse matching a revised set of constraints. It takes a parse ID and a set of constraints.

7.4 Interface to Computational Management

Again to facilitate communication the Simple Object Access Protocol (SOAP) (Gudgin et al., 2003) has been used to implement a lightweight interface between the Workflow Management module and the Computational Management module. The SOAP implementation supports 2 basic functions: `submit_sentence_for_parsing` and `get_next_sentence_for_review`.

- `submit_sentence_for_parsing` is used for transferring the sentences from the reviewed sentences queue to the active learning framework; and
- `get_next_sentence_for_review` is used for transferring the parsed sentences from the active learning framework to the Workflow Management module

8 Computational Management Module

As mentioned earlier, the Computational Management Module consists of two further submodules, one for active learning and the other for computational grid interaction management.

8.1 Active Learning Sub-Module

Our implementation of active learning involves a variety of differently parameterised instances of the CCG parser, with the view that an evaluation of each model will identify the best parse and constraints for a given sentence.

This module is instantiated based on some threshold - either time based (e.g. once every 24 hours) or queue based (e.g. when there are 100 modified parses). This asynchronous server side component allows discontinuity between the user-centric review process and the computational impact of large scale re-parsing.

Our active learner uses committee-based sampling (Dagan and Engelson, 1995) using differing supertagging and parsing statistical models as committee members. Where there are many annotation options, the most popular alternatives from the committee will be passed to annotators to help select the correct annotation more efficiently. This will minimise the cognitive load of selecting between too many alternatives.

8.2 Grid Sub-Module

The purpose of the Grid component is to manage all aspects of interaction with the distributed computational environment in which the parser itself is running. The Grid submodule handles low level execution including the queuing, dispatch and execution of analysis tasks, and fetching the results.

The experimental environment is setup with compute infrastructure in Sydney and in Melbourne. At the Sydney node, the system environment is a cluster of 9 dual-CPU Linux machines running MPI middleware. At the Melbourne node the system environment is a cluster of machines running Linux, managed by the NorduGrid Advanced Resource Connector (ARC). On each node, the CCG parser is installed.

The CCG toolkit is installed on the respective clusters and simply instantiated by the active learning framework as threshold boundaries are reached.

The Grid sub module selects the relevant compute node for execution of the current parse task. (In the simplest case, perhaps a round robin approach to selecting the compute facility for subsequent re-parsing runs would appear to be sufficient, although for more intensive human annotation sessions, batch mode parser execution with probe-based load measurement is probably desirable for a scalable and robust implementation). A job description is then created specific to the node requirements. The job is then passed to the head node of the cluster.

9 Discussion

The previous sections describe an architecture for distributed, computationally intensive, mixed initiative linguistic analysis. We believe this contribution is notable for a number of reasons including:

- a completely modular systems architecture, in contrast with tightly bound end-to-end systems which typically dominate this application space;
- coordinated yet distinctly decoupled computational and human effort, allowing both parties to contribute to the overall effort with maximum efficiency;
- re-usable, open sourced components which are sufficiently flexible to allow other interested parties to build from an established base, rather than the ground up
- an instantiation of service oriented NLP via open standards

We are motivated to modularise the overall system as much as possible to allow maximum flexibility for future extensions. In particular, our selection of the CCG parser is relatively arbitrary; any parser should be able to be swapped in for the CCG parser (e.g. an HPSG parser) with the only overhead being support for parallelisation and an API which can be functionally mapped to our SOAP based interface. Correspondingly, we envisage that the GUI component should be generalised sufficiently to allow for the rendering of a variety of different parse tree representations.

10 Status

At the time of writing the status of the components required is as follows:

- Visualization and Analysis module

  A prototype GUI has been implemented which can render CCG derivations in several formats. Lexical categories can now be modified in the GUI and the parse regenerated directly with the new constraints. This does not currently use SOAP for getting next parse.

- Workflow Management module

  The 7 SOAP methods (5 for Visualization and Analysis module interface, and 2 for Computational Management module interface) are implemented as a CGI application in Python. Basic user and task management implemented.

- Computational Management module

  The parameterisation and brokering framework for grid execution is deployed in production.

  Production grids are operational (building over existing infrastructure) at both Melbourne and Sydney sites. The CCG parser on these systems; in the Melbourne case, the active learning framework can be instatiated by a web services / SOAP based interface to NorduGrid's native job brokering system.

11 Conclusion

We have proposed an architecture for performing distributed annotation of CCG derivations. This architecture attempts to solve three key problems in the efficient preparation of large scale NLP resources: 1) selecting sentence to annotate which creates the most *useful* corpus for statistical parsers. 2) maximising the *annotator efficiency* and *minimising error*; 3) allowing distributed annotators to *share expertise*.

We have attempted to address these problems using a combination of machine learning techniques and grid computing infrastructure. In particular, Active Learning will identify the best sentences to annotate; interactive correction will make the most of our annotators time; and our workflow manager will allow (even remote) annotators to share their expertise more effectively.

While our implementation is relatively immature at this point, we believe the architecture proposed in this paper, along with the specific components, will be able to be reused in multiple contexts.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback, and to David Vadas and Toby Hawker for testing the CCG GUI. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of the EMNLP Conference*, pages 9–16, Barcelona, Spain.

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

Marcus Becker, Ben Hachey, Beatrice Alex, and Claire Grover. 2005. Optimising selective sampling for bootstrapping named entity recognition. In *Proc. of the ICML-2005 Workshop on Learning with Multiple Views*.

David Carter. 1997. The treebanker: a tool for supervised training of parsed corpora. In *Proc. of the Workshop on Computational Environments for Grammar Development and Language Engineering*, Madrid, Spain.

Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proc. of the 20th COLING*, pages 282–288, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 103–110.

Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proc. of the EMNLP Conference*, pages 111–118, Barcelona, Spain.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1995. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. MIT Press.

James Curran. 2003. Blueprint for a high performance nlp infrastructure. In *Proc. of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, Edmonton, Canada.

Ido Dagan and Sean P. Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Proc. of the ICML*, pages 150–157.

David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. 1997. Mixed-initiative development of language processing systems. In *Proc. of the 5th conference on Applied NLP*, pages 348–355.

Robin Dunn. 2005. wxPython toolkit. `http://www.wxpython.org`.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the EMNLP Conference*, pages 167–202, Pittsburgh, PA.

Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. 2003. SOAP version 1.2 part 1: Messaging framework. `http://www.w3.org/TR/2003/REC-soap12-part1-20030624/`.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the 3rd LREC Conference*, pages 1974–1981, Las Palmas, Spain.

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

Baden Hughes and Steven Bird. 2003. Grid-enabling natural language engineering by stealth. In *Proc. of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*, Edmonton, Canada.

Baden Hughes, Steven Bird, Ewan Klein, and Haejoong Lee. 2004a. Experiments with data intensive nlp on a computational grid. In *Proc. of the 2004 Hong Kong International Workshop on Language Technology*.

Baden Hughes, Srikumar Venugopal, and Rajkumar Buyya. 2004b. Grid-based indexing of a newswire corpus. In *Proc. of the 5th IEEE Workshop on Grid Computing*.

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.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods Treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1253–1257, Taipei, Taiwan.

2005. ReportLab toolkit. `http://www.reportlab.org/rl_toolkit.html`.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th ACL Conference on Applied NLP*, pages 88–95, Washington, DC.

Julian Smart, Kevin Hock, and Stefan Csomor. 2005. *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall.

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

Min Tang, Xiaoqing Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proc. of the 40th Annual Meeting of the ACL*, pages 120–127, Philadelphia, PA USA.

Multi-document Summarisation and the PASCAL Textual Entailment Challenge

Nicola Stokes
NICTA Victoria Laboratory,
Department of Computer Science
and Software Engineering,
University of Melbourne.
nicola.stokes@nicta.com.au

Eamonn Newman
School of Computer Science
and Informatics,
University College Dublin,
Ireland.
eamonn.newman@ucd.ie

Abstract

A fundamental problem for systems that require natural language understanding capabilities is the identification of instances of semantic equivalence and paraphrase in text. The PASCAL Recognising Textual Entailment (RTE) challenge is a recently proposed research initiative that addressed this problem by providing an evaluation framework for the development of generic "semantic engines" that can be used to identify language variability in a variety of applications such as Information Retrieval, Machine Translation and Question Answering. This paper discusses the suitability of the RTE evaluation datasets as a framework for evaluating the problem of redundancy recognition in multi-document summarisation, i.e. the identification of repetitive information across documents. This paper also reports on the development of an additional dataset containing examples of informationally equivalent sentence pairs that are typically found in machine generated summaries. The performance of a competitive entailment recognition system on this dataset is also reported.

1 Introduction

The aim of multi–document summarisation (MDS) is to generate a concise and coherent summary given a cluster of related documents. Although this process is a natural extension of single–document summarisation, MDS poses a number of unique challenges such as, how to manage contradictory and repetitive information in the cluster, and how to order extracted information in the resultant summary. A popular approach to the MDS problem is to first identify and cluster repetitive information units across documents, then select representative sentences from the "dominant" clusters, and finally generate an extractive summary from these sentences. This approach assumes, like many others in text summarisation, that the repetition of information is an indication of information importance and consequently summary relevancy. The simplest method for determining commonality across documents is to group text units (e.g. sentences, paragraphs) that exhibit a high concentration of word overlap. However, this approximate method for recognising similar semantic content is often insufficient due to instances of language variability such as paraphrase and synonymy. Figure 1 shows two sentences (A and B) that are semantically equivalent but syntactically different.

Text A: Agassi's dream run is ended by world's number one player. Text B: Federer beats Agassi.

Figure 1: Paraphrases with minimal word overlap

In this paper we discuss the suitability of the recently proposed PASCAL Recognising Textual Entailment (RTE) challenge (Dagan et al., 2005a) as an evaluation methodology for determining the performance of redundant information identification techniques in the context of MDS. The aim of the RTE challenge is to aid the development of generic "semantic engines" that can be used in a number of applications such as Information Retrieval, Information Extraction, Text Summarisation and Machine Translation. Two types of language variability were investigated in this year's challenge: exact paraphrases and textual entailment or subsumption. The evaluation was defined as a binary classification problem where participating systems were required to identify entailment relationships between sentence pairs, i.e. a sentence A entails another sentence B if the meaning of B can be inferred from the meaning of A (Dagan et al., 2005b). During the data collection effort for the challenge, annotators were asked to limit the number of "difficult" cases of entailment

Proceedings of the Australasian Language Technology Workshop 2005, pages 215–223,
Sydney, Australia, December 2005.

that they included in the dataset. The entailment pairs shown in Figure 2 are representative of the level of difficulty of the subsumption relationships found in the data, where entailment, in the majority of cases, requires syntactic matching, and synonym/paraphrase recognition rather than complex logical inference. In this way techniques that recognise redundant information in MDS and entailment in the RTE challenge have a lot in common. This point will be discussed in more detail in Section 2 of the paper.

The RTE development and test sets are composed of entailment examples taken from seven distinct application settings. The "Comparable Documents" portion of the collection is intended to be representative of the types of entailment and semantic equivalence found in multi-document summarisation. In general, participating systems at the RTE workshop performed significantly better (achieving as high as 87% accuracy) on this portion of the corpus. This result suggests that the types of entailment and semantic equivalence found in MDS are significantly less challenging than entailment found in other application settings. In this paper we will show that this result is misleading, and that the difficulty of identifying language variability in MDS is comparable with the level of difficulty observed in the other application domains explored by the RTE initiative.

The following section motivates the need for evaluating sub–tasks in MDS such as redundant information identification, and provides a brief overview of the techniques that have been used to identify language variability in MDS and at the RTE challenge. Section 3 discusses the RTE framework in the context of MDS and argues for the inclusion of additional examples of language variablity that frequently require identification in MDS but are not represented in the current RTE evaluation dataset. Section 5 describes the University College Dublin (UCD) RTE system, which detects entailment between sentence pairs using linguistic and statistical language analysis techniques[1]. Section 6 discusses the performance of this system at the RTE workshop. In addition, the results of some initial experiments are provided that support the assertion that the performance of a competitive RTE system in an MDS application is comparable with its performance in other RTE applications settings. Finally Section 7, discusses some conclusions and future directions for this work.

Task=Comparable Documents; Paraphrase Example; Judgement=TRUE
Text A: Satomi Mitarai died of blood loss.
Text B: Satomi Mitarai bled to death.

Task=Comparable Documents; Textual Entailment Example; Judgement=TRUE
Text A: The Kota (Fort), or Old City, for example, sometimes called the downtown section, is the central business district and Indonesia's financial capital.
Text B: The Kota is the country's business center.

Task=Comparable Documents; Syntactic Variation Example; Judgement=TRUE
Text A: Jakarta floods easily during the rainy season.
Text B: Jakarta is easily flooded during the rainy season.

Figure 2: Examples of syntactic variation, paraphrase and information subsumption in the RTE dataset

2 MDS and RTE

So why is the RTE challenge an attractive sub–task evaluation methodology for MDS? Firstly, identifying semantic relationships and correctly clustering informationally equivalent sentences is a critical analysis component in many MDS systems for the following reasons: if sentences are incorrectly clustered then the commonality between the documents is harder to determine, and redundant (i.e. repetitive) information will be included in the summary – an outcome that summarisation systems want to avoid at all costs. Secondly, there are inherent limitations with the current summarisation evaluation standard provided by the Document Understanding Conference (DUC)[2], where both automatic and manual evaluation strategies are used to measure summary quality in terms of coverage, information redundancy, readability,

[1]The author was involved in the development of this system before moving to the NICTA Victoria Research Laboratory.

[2]DUC is an annual NIST sponsored workshop that provides participants with summarisation tasks and a corresponding evaluation framework, i.e. corpora, gold standard summaries and evaluation metrics.
`http://duc.nist.gov`

coherence and grammaticality. Since its creation in 2001, the DUC initiative has helped to ensure that real and transparent progress is being made in summarisation research; however, because the DUC evaluation methodology is determining the performance of many difficult natural language processing (NLP) components concurrently (i.e. semantic analysis, content selection, sentence ordering and natural language generation), it is often difficult to establish which techniques employed by a particular high performing summarisation system have contributed most to its overall success. As such summarisation researchers are recognising the need for distinct evaluation frameworks for each of these sub-components. For example, researchers at Columbia University have separately evaluated their sentence clustering algorithm, SimFinder, which is employed in their NewsBlaster summarisation system (McKeown et al., 2002). More recently Barzilay and Lapata (Barzilay and Lapata, 2005) describe an evaluation methodology for text coherence techniques, which are commonly used by summarisation systems to improve text readability. The following subsection provides a flavour of the Entailment and Semantic Equivalence techniques presented at the PASCAL RTE–2005 challenge, followed by a description of two important contributions made by Text Summarisation researchers in this area.

2.1 Language Variability Recognition Techniques

The 2005 PASCAL RTE challenge is described by the organisers as "an initial attempt to form a generic empirical task that captures major semantic inferences across applications" (Dagan et al., 2005b). Sixteen groups submitted their RTE system results to the workshop. The systems used a broad range of linguistic knowledge resources, statistical association metrics and logical inference mechanisms. As already stated, the simplest type of semantic equivalence measure that can be used to identify entailment is a measure of vocabulary overlap. Consequently, nearly all of the systems at the workshop considered uni-gram or n-gram overlap metrics when classifying entailment. A number of more sophisticated methods were also proposed. These measures either used statistical cooccurrence metrics (e.g. latent semantic indexing), lexical resources for detecting semantic relationships between verbs, nouns, and adjectives (e.g. WordNet (Millar, 1995)) or a combination of both. Syntactic–based overlap measures, which involves calculating the degree of match between parse tree representations of the sentence pair, were also popular. A few groups also incorporated a logical prover with some additional world knowledge resource such as a geospatial ontology or a semantic taxonomy. Many of the submitted systems, such as the UCD submission described in the following section, considered more than just one of these measures during the entailment recognition process. More specifically, these lexical, syntactic, semantic or logical–based inference measures were used as partial (rather than conclusive) evidence of the presence or absence of an entailment relationship between two sentences.

Overall the entailment recognition *accuracy* (see Section 6 for definition) of the participating systems at the workshop ranged from 50-60% where accuracy measures greater than 0.535 and 0.546 are better than chance at the 0.05 and 0.01 level, respectively (Dagan et al., 2005b). The general conclusion of the workshop was that relatively simple metrics used in combination performed better than more complex, "deeper" metrics such as logical inference or the incorporation of world knowledge into the classification computation. An obvious explanation for this outcome is that deep linguistic analysis methods are more prone to errors than simple term overlap metrics due to additional complexities such as word sense disambiguation.

So how do RTE techniques compare to the repetitive information detection methods used by the text summarisation community? Well as already stated, summarisation researchers have tended to favour simple similarity metrics based on the number of shared words. There are a couple of notable exceptions, however, which have been investigated by researchers at Columbia University.

Possibly the most well-known and successful approach to similarity detection in automatic summarisation is the SimFinder (Hatzivassiloglou et al., 2001) algorithm. This algorithm clusters sentences that share thematic content determined by a set of similarity features based on word, stem and Wordnet concept overlap as well as more complex features that capture match at a syntactic level such as subject-verb and verb-object relations. The subsequent clustering of sentences is then performed using a non-hierarchical clustering tech-

nique. Representative sentences from these clusters are then used to generate a summary.

(Barzilay and McKeown, 2005a) describe a revision strategy for improving the readability of the summary output of the SimFinder algorithm. Their revision system, MultiGen, searches for semantically equivalent textual units in the dependency tree graph representations of the summary sentences. Semantically similar words and phrases are identified using the WordNet taxonomy and a paraphrase dictionary, automatically constructed from parallel monolingual corpora. So once an overlapping paraphrase has been detected in the dependency trees this analysis then facilitates "information fusion", i.e. the generation of a single sentence that represents the information in the overlapping sentences. This text generation technique has been integrated into the Columbia NewsBlaster multi–document summarisation system (McKeown et al., 2002).

It is clear from this discussion that the Text Summarisation community has much to gain from, and contribute to, the advancement of Entailment and Semantic Equivalence recognition research.

3 RTE and language variability in MDS

In this section of the paper we comment on the coverage of the RTE evaluation corpora with respect to the type of real-world examples of semantic equivalence that require detection during multi-document summarisation. For the RTE 2005 challenge two development collections and one test collection where released to participants[3]. In each case, the datasets consisted of an even number of positive and negative examples of entailment between sentence pairs. During the development of these datasets annotators were asked to collect relevant examples that corresponded to typical success and failure settings in seven different applications, i.e. Information Retrieval (IR), Information Extraction (IE), Machine Translation (MT), Question Answering (QA), Paraphrase Acquisition (PP), Reading Comprehension (RC) and Comparable Documents–style tasks (CD) such as multi–document summarisation. A more detailed discussion of the annotation process can be found in (Dagan et al., 2005b).

[3]The RTE datasets can be downloaded from: `http://www.pascal-network.org/Challenges/RTE/Datasets`

As already stated, the motivation behind this paper is to establish whether or not these examples of language variablity are reflective of the types of information redundancy found in an MDS setting. Particularly in the case of the CD sentence pairs which are reportedly representive of the MDS task. To answer this question we considered Mani's analysis of this problem in his review of MDS methods, where he defines 4 distinct types of redundancy between text elements in MDS (Mani, 2001):

1. Two text elements are string identical when they are exact repetitions, i.e. the same sentence is repeated in multiple articles.
2. Two text elements are semantically equivalent when they are exact paraphrases of each other.
3. Two text elements are informationally equivalent if they are judged by humans to contain the same information.
4. A text element A informationally subsumes text element B if the information in element B is contained in A.

A manual examination of the RTE datasets shows that string identity and informational equivalence are not represented in these collections. Figure 2 provides examples of paraphrase and informational subsumption, i.e. textual entailment in the RTE data. The exclusion of string identical examples isn't considered critical as the detection of exact repetition is trivial. However, the lack of Mani's informational equivalence type examples is more troublesome. An example of informational equivalence is shown in Figure 3. What differentiates this example of language variablity from those in Figure 2, is that the common information unit is an embedded paraphrase surrounding in both sentences by additional information. More specifically, while Text A and B share the information unit: "American Airlines laid off flight attendants", they also contain additional non-overlapping information units, i.e. the federal judge turned aside a union bid to block the job losses; unions warned travellers to expect long delays due to protests. From our analysis we can conclude that examples of *exact* paraphrase and entailment are the exception rather than the rule in MDS and other CD–type applications. More often than not these systems will be required to deal with noisier instances of semantic equivalence where sentences repeat embed-

Task=MDS; Embedded Paraphrase Example; Judgement=TRUE
Text A: *American Airlines began laying off hundreds of flight attendants on Tuesday*, after a federal judge turned aside a union bid to block the job losses.
Text B: Unions have warned travellers that they can expect long delays this weekend as protests begin after *American Airlines let a large number of flight attendants go last week.*

Figure 3: An example of informational equivalence and embedded paraphrase

ded information units rather than exhibit complete semantic overlap (i.e. exact paraphrase) or subsumption.

In MDS, if the system can successfully detect these fuzzier examples of information redundancy it can make an informed decision on whether to: (a) substituted one sentence for another in the summary without any critical loss of information or (b) fuse these sentences together as proposed by (Barzilay and McKeown, 2005a). Sentence fusion would probably be the most appropriate option in the case of the embedded paraphrase example shown in Figure 3. With this type of natural language generation application in mind, it would be beneficial if the RTE classification task also required systems to explicitly identify and return the common information unit(s) between each sentence pair, i.e. the system must justify its classification decision.

4 An MDS-based Informational Equivalence Dataset

This section describes the development of a complementary RTE-style corpus of sentence–pairs that are more reflective of the types of information redundancy observed during multi-document summarisation.[4]. Annotators were asked to use Columbia's online NewsBlaster summarisation system[5] (a consistent top-performer at the annual DUC summarisation evaluation workshop) to aquire relevant sentence pairs. This curation strategy was employed to ensure that the MDS dataset was representive of the types of informational equivalence that are problematic in MDS. A subsequent analysis of the official DUC summary submissions to the multi-document summarisation task defined for the 2004 challenge (i.e. DUC task 2) indicates that these NewsBlaster examples are consistent with the types of repetitive information that were missed by sentence clustering strategies employed by other top performing summarisation systems at the workshop.

In line with the task-specific subsets in the RTE collection, the MDS dataset consists of 100 sentence pairs: 50 positive and 50 negative instances of informational equivalence. Figure 4 shows an example of each classification type. In the previous section it was explained that in order for a sentence pair to be tagged as a positive instance of informational equivalence it had to share an information unit; however, no formal definition of what constitutes such as unit was provided. The formulation of such a definition is a challenge in itself, and is currently receiving significant attention from the Text Summarisation community in the context of summarisation evaluation (Nenkova and Passonneau, 2004; Amigo, 2004). In the context of this task, an information unit is defined as a unit of text that contains at least one subject-verb relationship, (i.e. a noun phrase like "Air France Flight 358" is not a large enough information unit but "Air France Flight 358 crashed" is). In addition, when choosing these examples annotators were asked to be mindful of the underlying classification task in the context of a summarisation application, i.e. would the inclusion of both sentences result in unnecessary repetition in a summary. Any disagreement between annotator regarding the classification of certain pairs was discussed and resolved before experimentation on the corpus began.

From the MDS examples in Figure 4 it can also be seen that these sentences often make reference to vague temporal expressions such as "deadline...set for Monday" and "Monday deadline". In order to ground these temporal references to points in time the full text of the original source document would need to be analysed. However, temporal resolution is not necessary in this classification task since examples were carefully chosen to ensure that if an event (such as a "suicide bomb attack") is mentioned in both sentences, then the system can assume that this information unit is referring to the

[4]The MDS corpus can be downloaded from: `http://www.cs.mu.oz.au/~nstokes/TE/MDS_corpus_1.0.xml`

[5]The NewsBlaster summarisation system: `http://newsblaster.cs.columbia.edu`

same instance of the event in time.

Task=MDS; Pair Id=4; Judgement=TRUE;
Text A: The United States ratcheted up its pressure Saturday on Iraqi negotiators who are trying to meet a deadline for writing a draft constitution set for Monday.
Text B: With Iraq's parliament facing a Monday deadline to approve a new constitution, President Bush said Saturday that the document "is a critical step on the path to Iraqi self-reliance".

Task=MDS; Pair Id=62; Judgement=FALSE;
Text A: Discovery was loaded with nearly 7,000 pounds of garbage that had accumulated in the space station since it was last visited by a shuttle in December 2002.
Text B: The Discovery crew spent nine of their first 13 days in orbit transferring supplies to the space station.

Figure 4: Pair 4 and Pair 62 are examples of positive and negative informational equivalence in the MDS dataset.

With regard to the negative examples of information overlap in the MDS corpus, sentence pairs were picked from summaries that contained some word overlap, but which would still be considered unique information contributors to a summary. This helped to ensure that these negative sentence pairs were non–trivial.

During the creation of this corpus a number of examples of "contradiction" (i.e. conflicting news reports on the details of a specific event) between potential informationally equivalent sentence pairs were found. Although these examples represent another important problem in MDS, they were not included in the final version of the corpus because they frequently occur in the RTE challenge datasets in the form of negative entailment examples as shown in Figure 5.

In the following sections we describe the UCD RTE system, and compare its performance on the MDS dataset to its performance on the RTE test set. As already stated, this experiment is used to investigate our claim that the CD task data in the RTE challenge is unrepresentative of language variability in MDS.

Task=Comparable Documents; Judgement=False;
Text A: Jennifer Hawkins is the 21-year-old beauty queen from Australia.
Text B: Jennifer Hawkins is Australia's 20-year-old beauty queen.

Figure 5: An example of contradiction in the RTE data collection.

5 The UCD Textual Entailment Recognition System

In this section, we present an overview of the UCD Textual Entailment Recognition system, which was originally presented at the PASCAL RTE workshop (Newman et al., 2005). This system uses a decision tree classifier to detect an entailment relationship between pairs of sentences that are represented using a number of difference features such as lexical, semantic and grammatical attributes of nouns, verbs and adjectives. This entailment classifier was generated from the RTE training data using the C5.0 machine learning algorithm (Quinlan, 1993). The features used to train and test the classifier were calculated using the following similarity measures:

- The ROUGE (Recall–Oriented Understudy for Gisting Evaluation) (Lin and Hovy, 2004) n-gram overlap metrics, which have been used as a means of evaluating summary quality at the DUC summarisation workshop. The Rouge package provides measurement options such as uni-gram, bi-gram, tri-gram and 4-gram term overlap, and a weighted and unweighted longest common subsequence overlap measure.
- The Cosine Similarity metric calculates the cosine of the angle between the respective term vectors of the sentence pair.
- The Hirst–St-Onge WordNet–based measure (Millar, 1995), is an edge counting metric that estimates the semantic distance between words by counting the number of relational links between them in the WordNet taxonomy (Budanitsky and Hirst, 2001). This metric also defines constraints on the length of the path and the types of transitive relationships that are allowed between concepts (nodes) in the taxonomy. These constraints are important because unlike other WordNet–based

semantic relatedness measures (which only consider IS–A relationships) the Hirst–St Onge metric searches for paths that traverse the IS–A and HAS–A hierarchies in the noun taxonomy. Hence, this metric provides better coverage at an increased risk of detecting spurious relationships if unrestricted paths were allowed between concepts. This feature was implemented using the Perl Wordnet Similarity modules developed by (Patwardhan et al., 2003).

- A verb–specific semantic overlap metric, that uses the VerbOcean semantic network (Chklovski and Pantel, 2004b; Chklovski and Pantel, 2004a) to identify instances of antonymy and near-synonym between verbs. The relationships between verb–pairs in VerbOcean were gleaned from the web using lexico–syntactic patterns. Although WordNet provides a verb taxonomy, the VerbOcean data was used because it appears to provide better coverage of the types of relationships needed for detecting entailment.
- A Latent Semantic Indexing (Deerwester et al., 1990) measure, like the WordNet measure, attempts to calculated similarity beyond vocabulary overlap by identifying latent relationships between words though the analysis of cooccurrence statistics in an auxiliary news corpus.
- The final similarity measure is based on a more thorough examination of verb semantics. This measure finds the longest common subsequence in the sentence–pair, and then detects evidence of contradiction or entailment in the subsequence (such as verb negation, synonymy, near-synonymy, and antonymy) using the VerbOcean taxonomy. An example is shown in Figure 6.

A more detailed description of the UCD system can be found in (Newman et al., 2005).

6 Language Variability Recognition Experiments and Results

This section of the paper reports on the performance of the UCD RTE system on the RTE and MDS datasets. The RTE challenge defined two evaluation metrics:

- An *accuracy score* which is calculated as the number of correctly classified sentence pairs (positive and negative) returned by the system divided by the number of sentence pairs in the dataset.
- A *confidence-weighted score (CWS)* that ranges between 0 (no correct judgements at all) and 1 (perfect score), and rewards the system when it assigns a higher confidence score to correct judgements rather than to incorrect ones.

Task=Paraphrase Acquisition; Judgement=FALSE
Text A: *France on Saturday* flew *a planeload of United Nations aid into eastern Chad* where French soldiers prepared to deploy from their base in Abeche towards the border with Sudan's Darfur region.

Text B:*France on Saturday* crashed *a planeload of United Nations aid into eastern Chad*

Figure 6: The Longest Common Subsequence is highlighted in italics.

The UCD RTE and MDS results are shown in Table 1. The entailment classifier in the MDS and RTE experiments was trained using the RTE corpus training sets (`dev1` and `dev2`). The average accuracy and CWS scores (0.565 and 0.6 respectively), and the task results listed below this row in the table represent the official UCD results reported at the RTE 2005 workshop. A manual analysis of these results showed that many of the misclassified errors made by the UCD system could be attributed to the occurrence of equivalence phrasal and compositional paraphrases e.g. "X invented Y" = "Y was incubated in the mind of X". As explained in Section 5 the system can only identify word–level, atomic paraphrase units (e.g., child = kid; eat = devour) that are defined in the VerbOcean and WordNet lexical resources. A more detailed discussion of system misclassifications is provided in (Newman et al., 2005).

Out of 16 groups UCD's average accuracy and CWS scores were ranked 4th and 5th respectively, where system accuracy results ranged from 0.586 to 0.495 and CWS scores from 0.686 to 0.507. In general, systems performed significantly better on the CD entailment examples, and for many it was this score that added some respectability to their average accuracy score. The most plausible explanation for these high CD scores (as high as 87% accuracy), accord-

ing to (Dagan et al., 2005b), is that vocabulary overlap metrics performed very well on this task because sentence pairs containing common terms were more likely to have the same meanings than in the other tasks. This implies that MDS systems need nothing more than vocabulary overlap metrics, and that the negative effect of errors from this component of an MDS system is minimal. However, a comparison of the UCD system results on the CD and MDS language variablity examples suggests that redundant information detection is as difficult as the other tasks investigated, and that further research effort is also required in this area.

Task	**Accuracy**	**CWS**
MDS	0.5400	0.6006
RTE Average	*0.5650*	*0.6000*
CD	0.7400	0.7764
IE	0.4917	0.5260
IR	0.5444	0.6130
PP	0.5600	0.5006
MT	0.5083	0.5130
QA	0.5385	0.5006
RC	0.5286	0.5685

Table 1: RTE and MDS Accuracy and CWS results for the UCD entailment classifier.

7 Conclusions

This paper evaluates the RTE challenge as a potential evaluation framework for comparing the performance of redundant information recognition strategies used in multi–document summarisation (MDS) to detect informational equivalence across documents. Most MDS systems use simple word counts to identify repetitive information. The problem with this approach is that many sentences that convey the same information show little surface resemblance due to linguistic phenomenon such as paraphrase and synonymy. The RTE challenge provides an opportunity for summarisation researchers to evaluate more sophisticate redundancy identification techniques independent of the summarisation task. However, an analysis of the RTE development and test sets show that this data is not representative of the types of informational equivalence that require detection during the MDS process. More specifically, although subsumption relationships are a natural occurrence in applications such as Question Answering and Information Retrieval (where the answer/relevant document will always entails the question/query) this is not the case for Comparable Documents-style tasks. The results of an experiment on a complementary dataset of MDS informational equivalence examples using a competitive RTE system showed that identifying redundancy in MDS is more challenging than the results on the Comparable Documents portion of the RTE test set would suggest. Consequently, if the ultimate aim of the PASCAL RTE challenge is to build "generic semantic engines" then future evaluations will also have to consider the identification of embedded (semantic and syntactic) paraphrases across sentences.

An obvious extention of this work would be to incorporate the UCD RTE system into an MDS system, and compare its effect on summary performance against a baseline semantic equivalence measure such as cosine similarity. It would also be interesting to further investigate how well the RTE evaluation framework simulates the process of identifying repetitive information in MDS and other applications. In a paper by Barzilay and Elhadad (2003), on sentence alignment for monolingual comparable corpora, it was shown that the effectiveness of the alignment process increased when the context surrounding sentences was also considered. This conclusion suggests that future RTE evaluations should also consider evaluating the role of context in the entailment detection process, where additional context is provided by the document in which the sentence occurred.

8 Acknowledgements

The support of Enterprise Ireland and NICTA (National ICT Australia) is gratefully acknowledged.

References

E. Amigo. 2004. An empirical study of information synthesis tasks. In *Association for Computational Linguistics (ACL'04)*.

R. Barzilay and N. Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Empirical Methods in Natural Language Processing (EMNLP'03)*.

R. Barzilay and M. Lapata. 2005. Modeling local coherence: An entity-based approach. In *Association for Computational Linguistics (ACL'05)*.

R. Barzilay and K. McKeown. 2005a. Sentence

fusion for multidocument news summarization. *Computational Linguistics*, 31(3).

A. Budanitsky and G. Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *the Workshop on WordNet and Other Lexical Resources, NAACL'01.*

T. Chklovski and P. Pantel. 2004a. Global path-based refinement of noisy graphs applied to verb semantics. In *the International Joint Conference on NLP (IJCNLP-05)*, pages 11–13.

T. Chklovski and P. Pantel. 2004b. VerbOcean: Mining the web for fine–grained semantic verb relations. In *Empirical Methods in Natural Language Processing (EMNLP-04).*

I. Dagan, O. Glickman, and B. Magnini (eds). 2005a. In *the PASCAL Recognising Textual Entailment Challenge Workshop, April 11th-13th 2005, Southampton, UK.*

I. Dagan, O. Glickman, and B. Magnini. 2005b. The PASCAL recognising textual entailment challenge. In *the PASCAL Recognising Textual Entailment Challenge Workshop 2005*, pages 1–8.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science.

V. Hatzivassiloglou, J. Klavans, M. Holcombe, R. Barzilay, Min-Yen Kan, and K. McKeown. 2001. SimFinder: A flexible clustering tool for summarization. In *the Workshop on Automatic Summarization, NAACL-01.*

C.-Y. Lin and E. Hovy. 2004. Automatic evaluation of summaries using n-gram co–occurence statistics. In *the Document Understanding Conference (DUC'04), National Institute of Standards and Technology.*

I Mani. 2001. *Automatic Summarization.* John Benjamins (Natural language processing series, edited by Ruslan Mitkov, volume 3), Amsterdam.

K. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. 2002. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *the Human Language Technology Conference (HLT'02).*

G. Millar. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

A. Nenkova and R. Passonneau. 2004. Evaluating content selection in summarization: The Pyramid Method. In *HLT–NAACL'04.*

E. Newman, N. Stokes, J. Dunnion, and J. Carthy. 2005. UCD IIRG approach to the Textual Entailment Challenge. In *the PASCAL Recognising Textual Entailment Challenge Workshop*, pages 53–56.

S. Patwardhan, J. Michelizzi, S. Banerjee, and T. Pedersen. 2003. *WordNet::Similarity Perl Module* `http://search.cpan.org/dist/wordnet-similarity/lib/wordnet/similarity.%pm`.

J.R. Quinlan. 1993. C5.0 machine learning algorithm. `http://www.rulequest.com`.

Design and development of a speech-driven control for an in-car personal navigation system

Ying Su, Tao Bai, and Catherine I. Watson
Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand
c.watson@auckland.ac.nz

Abstract

The paper outlines the development and design of a speech driven control for a personal in-car navigation system which runs on a standard Pocket PC. The modified system enables speech driven menu navigation, speech shortcut commands and interactive dialogues. The speech recognition method is presented, sources of inaccurate recognition are identified, and solutions are presented. Speech recognition accuracies of 96% and 88%, depending on the task, are achieved in an in-car environment. One draw back is the time taken to perform the recognition. The speech driven control module which interfaces with the in-car navigator is designed to be flexible. These features are discussed.

1 Introduction

This paper presents the design and development of a proto-type speech-driven control for a personal in-car navigation system. The navigation system is currently on the market, it is an as automatic navigation software application integrated into a Pocket PC operating environment. Like most software applications on a Personal Digital Assistance (PDA), it requires manual user control via the hardware interface of the device, which consists of the touch screen. This has obvious limitations for in-car use, and a hand-free speech driven solution to control the navigator is being investigated.

With the navigation system, the user is able to access automatically extracted map information, GPS localization, and navigation instructions via a graphical interface (GUI). The application also performs analysis of the trip and automatic routing to the destination address entered by the user, which helps improve the efficiency of travelling. Besides the navigation features, the application also allows the user to customize the preference settings using the menus as part of the GUI.

In order to develop a prototype solution to enable speech-driven control for the in-car navigation system (hence forth called the Drive Router) there were two major functionality requirements; 1. the acquisition and recognition of user speech signals, and 2. the controlling the in-car navigation system according to the recognized speech commands. The control features specified in this prototype were speech-driven menu navigation, speech shortcut commands, and interactive dialogs.

The menu navigation feature allows the user to navigate through the different menus and preference setting screens of the GUI of the Drive Router by saying the names of the buttons on the GUI. The shortcut command triggers a transition that normally requires a series of GUI control actions. For example when the application is displaying the map screen, upon the recognition of the phrase “GPS status”, the GPS status screen will appear, this originally required the user to navigate through two menus to get it. Interactive dialog provides an efficient way for complex information retrieval. In this prototype, it is used to retrieve a destination address from the user.

These three features cover a relatively broad range of speech-driven control on the Drive Router system, as well as forming a foundation for further development. Since the features are targeting users, it is required that usability is taken into account when developing the features. The solution must be operational in the operating environment of the Drive Router, which is a standard Pocket PC with an integrated microphone. A Hewlett-Packard IPAQ h2200 Pocket PC (henceforth the HP Pocket PC) running the Microsoft Pocket PC 2003 Premium Operating System is used as the platform for the implementation.

Proceedings of the Australasian Language Technology Workshop 2005, pages 224–232,
Sydney, Australia, December 2005.

2 Speech acquisition and recognition

2.1 Embedded speech recognition system

A large amount of effort has been gone into the development of robust speech recognition solutions. However, most solutions are designed to operate on a PC based platform. The Driver Router, runs on a pocket PC and is in an embedded system. An embedded system is one that has CPU and is programmable but is not a general-purpose Personal Computer. Few speech recognition solutions are suitable for an embedded environment due to the limited amount of memory and computation power. Typically, embedded system does not have large enough hardware capacity to process large recognition vocabulary [1] and store statistical parameters [2]. Other limitations specific to speech related applications include poor speech acquisition, mainly due to the quality of the microphone, and internal noise generated within the device. The quality of the received speech signal directly affects the recognition accuracy.

Some embedded speech recognition solutions, such as the IBM personal speech assistant [3], require additional hardware support to overcome the constraints. Some others, such as the one mentioned in [4], make use of wireless communication to delegate the responsibility of recognition processing to more powerful remote servers. Within the scope of this project, the Driver Router is designed to operate in a standard Pocket PC with no extra hardware or remote server support. Thus an embedded speech recognition solution developed in software is desired. Among the few speech recognition software solutions available for embedded systems, the ScanSoft Automatic Speech Recognition (ASR) Embedded Development System (EDS) is used for experimental and evaluation purposes in the development of the prototype.

2.2 The ScanSoft ASR system

The ScanSoft ASR EDS is designed for the development of software based speech enabled features into Windows based applications. In particular, it can be incorporated into a Microsoft Windows CE environment, which the Microsoft Pocket PC operating system is configured upon.

The speech acquisition and recognition system recommended by [5] (see Figure 1) comprised of two major components, which are the AudioIn driver and the Vocon3200 Speech Recognition Engine. These components can be developed and configured using the Application Programming Interfaces (APIs) provided in the package, which are a range of functions in the C++ programming language, allowing the developer to construct and configure different modules of the system.

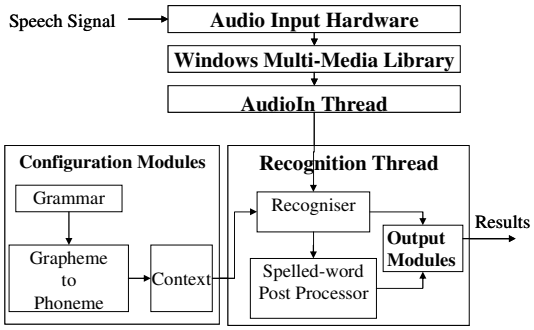


Figure 1: Speech acquisition and recognition using the ScanSoft ASR EDS. Produced based on [5] and [6].

The AudioIn driver is in charge of streaming analogue audio signals from the audio input hardware and supplying the samples of the signals to the recognition thread of the Vocon3200 Speech recognition Engine.

The engine performs recognition on 16-bit digital speech signal samples taken at a sampling frequency of 16 kHz. The engine performs continuous recognition. The speech signal is recognized against a limited set of commands, each specified as a vocabulary item in text and converted to a phonetic transcription. The spoken command vocabulary and the grammar rules can be specified in the text grammar file. The phonetic transcriptions of the text commands can then be generated by a Grapheme to Phoneme object, which utilizes a language model for the conversion rules. The language model used is the standard American English Model.

The phonetic transcriptions are then loaded into the context. On detection of the end of a spoken phrase, the signal is recognized with consultation to the context. The recognition algorithm is based on Hidden Markov Models [7]. With these models, each phoneme in a phonetic transcription is associated with a probability distribution. By analyzing an utterance, the transcriptions with higher probability of matching the actual speech can be found and separated from the less probable

ones. The recognition results generated on one utterance are the text vocabulary items associated with the most probable phonetic transcriptions selected by the Hidden Markov Model. Each result is also assigned a confidence level or confidence score, which indicates the likelihood of the match of result. The results with confidence levels above the Acceptance Threshold, which is defined by the developer, are then ready to be used by the consumer of the results. ScanSoft [5] claims that since the engine performs phoneme-based recognition and works with a standard language model to generate the phonetic transcriptions, it is able to perform speaker independent recognition without any training from the user.

The package also allows special development to perform recognition on spelling. Recognition by spelling imposes a big challenge because the utterance for a single letter is short and likely to be similar to other letters, hence leading to incorrect spelling recognition. The Spelled-word Post-Processor can be constructed using the package to improve spelling recognition accuracy. After the recognition engine is configured with a spelled-word specific grammar, a sequence of intermediate recognition results generated by the recognition thread can be fed into the post-processor. Each intermediate result corresponds to a character in the spelling and its confidence level. The post-processor then analyzes the results against a limited set of possible spelling defined in a data structure, namely a "spell tree", and produces the final recognition results, which contains the possible spelled words, each with a corresponding error score.

2.3 Accuracy testing

2.3.1 Testing and results

A Pocket PC based test application was developed to facilitate testing of the performance of speech acquisition and recognition system developed using the ScanSoft package in the actual operating environment of the Drive Router. The AudioIn driver and the recognition engine were incorporated in the test application, which was then run on a HP Pocket PC, equipped with an integrated microphone. The recognition accuracy is tested in both a laboratory acoustic environment, with a Signal-to-Noise Ratio (SNR) of 49 dB, and an in-car environment, with a SNR of 21 dB. To simulate the normal operating environment of the Drive Router, the in-car acoustic environment is setup with the Pocket PC placed 70cm away from the user. The noise in the environment mainly consists of engine noise and traffic noise.

Two sets of vocabularies are used in the spoken word accuracy testing, one set being 200 words randomly chosen from an English dictionary, the other being 40 spoken commands to be used during the operation of the Drive Router, including all menu navigation and shortcut commands developed in this prototype. To test the spelled word recognition accuracy 100 words were spelt and matched to a spell tree consisting of 10000 road names in the Auckland region. The test results are summarised in Table 1. The results of special interests are the accuracy for recognizing the 40 speech commands and the spelled road names in the in-car environment, which are 89% and 80% respectively. These directly relate to the quality of the speech-enabled features of the Drive Router when used in practise.

Signal to noise ratio	Accuracy for 200 spoken words	Accuracy for 40 spoken words	Accuracy for 100 Spelled words
49 dB	87%	94%	95%
21 dB	80%	89%	80%

Table 1: Test results on the accuracy of the ScanSoft recognition system.

2.3.2 Sources of inaccurate recognition

The main sources of inaccurate recognition include phonetic similarities between the spoken words, spelled word recognition errors and noise.

Since the recognition is phoneme based, it is difficult for the recognition engine to distinguish words with similarities in their pronunciations. For example, the English words "bit" and "pit" are only different by one phoneme in their pronunciations. These words form a confusable set. When confusable sets are present in the vocabulary, recognition errors are likely to occur if an item in the set is spoken. The probability of having similarities between the items increases as the size of the vocabulary increases, resulting in lower recognition accuracy in general. This is indicated by the decrease of accuracy as the vocabulary size changes from 40 to 200, shown in Table 1.

Spelled word recognition performed by the ScanSoft system is essentially phoneme based and character based, since the recognition engine is used to recognize the characters by their phonemes and the spelled word post processor is used to search for the better matching character sequences. Therefore the recognition accuracy is affected by the phonetic similarities between the characters and the number of characters in the spelling. Like the similarities between some spoken words, it is difficult for the engine to distinguish phonetically similar characters. Examples of such characters are the famous "E set", including the letters 'b', 'd', 'p' and 't' in the English alphabet [8]. With these characters, the engine is like to make substitution error, meaning one character being mistaken to another. Due to the short durations of the pronunciations of the characters are spelled with no obvious pauses in between, two consecutive characters are likely to be mistaken as one, resulting in deletion recognition errors. Finally signal inherited from the previous character or noise presented in the environment can be mistaken as additional characters, leading to insertion recognition errors.

Apart from recognition errors, there is always the possibility for spelling mistakes made by the user, which also include substitution, insertion and deletion errors. Insertion and deletion errors cause the length of the spelling to be different from the correct form, making it difficult for the spelled word post processor to match the input sequence with the correct spelling in the spell tree. From testing results, deletion errors are more likely to produce inaccurate recognition result, because less information is provided in the input sequence for processing.

Noise present in the environment can corrupt the information contained in the speech signal and introduce unwanted elements into the signal, which will be mistaken as part of the speech. It also affects the accuracy of detecting the trailing silence indicating the end of the utterance. As indicated in Table 1, the recognition performance is worse with a lower SNR. The test environment is created to consist mainly of random background noise in an in car environment. As commented in [9] speech recognition in an in-car environment is fragile and depends on driving conditions, further conversations between passengers significantly increase the level of confusion.

2.3.3 Representation of confidence level

As mentioned before, the ScanSoft recognition system associates a numerical confidence score with a spoken phrase recognition result to indicate how reliable the result is, and a numerical error score with a spelling recognition result to indicate how inaccurate the result is. Although it is clear that the higher the confidence score, the more reliable the spoken recognition result is, and the lower the error score, the more reliable the spelling recognition result is, the exact accuracy of the result and its relationship with the acoustic environment are not obvious from the raw scores. To suit the application, the raw score is converted to percentage accuracy that indicate the actually reliability of the result in specific to the in-car acoustic environment. The conversions are done using the following formulae:

%Spoken word accuracy = (100-cutoff_percentage) *(spoken_raw_score-spoken_cutoff_score) / (spoken_max_score-spoken_cutoff_score) +cutoff_percentage

%Spelled word accuracy = (100-cutoff_percentage) *(spelled_cutoff_error - spelled_raw_error) / (spelled_cutoff_error-spelled_min_error) +cutoff_percentage

The cutoff_percentage parameter is defined by the developer so that a result with a percentage accuracy below which will be rejected. To maintain a reasonable sensitivity of recognition, the cut-off percentage is set to be 90 percent. The maximum spoken score, the minimum spelling error score and the cut-off scores are obtained from the accuracy testing result in the in-car environment. With the cut-off percentage being 90 percent, the cut-off score for spoken word recognition is the score that is less than the raw confidence score of the correct recognition result 90 percent of the time. On the other hand, the cut-off error for spelled word recognition is the error score which is larger than the raw error score of the correct result 90 percentage of the time. The new representation of accuracy makes further development easier, allowing efficient evaluation of the result and comparison of confidence level between spoken and spelling recognition results since they are using the same measurement standard.

3 Recognition improvement strategies

The original configuration of the ScanSoft recognition engine, the recognition accuracy is

below 90 % in the in-car acoustic environment (see Section 2). Inaccurate recognition will lead to undesired control actions, which will affect the functionality and usability of the speech-enabled features. Some configuration requirements of the engine also impose challenges to achieve certain recognition features, which are important for the application. For example, to perform spelling recognition, the engine needs to be configured with a spelling specific grammar. This means with one configuration, an utterance can only be treated as either spelling or spoken phrases, but not both. Therefore, some strategies have been developed to improve the accuracy and flexibility of recognition.

3.1 Improving accuracy

3.1.1 State-dependent vocabulary

Throughout the operation of the Drive Router, the amount of possible speech commands can be large, and confusions are likely to be present in this large vocabulary. Instead of having the entire global speech command vocabulary active all the time, a state-dependent vocabulary configuration is used. With this approach, the vocabulary items activated and deployed by the engine at any point in time is limited to only the valid speech commands used by the Drive Router at the time. For example, when the Drive Router is in the map display state only the vocabulary items related to this state are activated and used for recognition, so these would include the phrases “Zoom in”, “Zoom out”, “Show me the menu”, and “GPS status”. With this smaller set of vocabulary, the chance of having similarities between the words is reduced. It also allows the commands to be chosen more easily, because phonetically similar phrases can now coexist in the vocabulary, as long as they are in different states. A reduced vocabulary size also helps reduce the setup time and processing time.

3.1.2 Result confirmation

Another strategy adopted is prompting for user confirmation if there are uncertainties in the results. If the engine has produced a list of recognition results with similar confidence levels on one utterance, an interactive dialog is used to inform the user of the possible options and ask the user to choose the correct one from the list.

3.1.3 Address recognition

The recognition accuracy also depends on the type of words to be recognized. As one of the desired control features, the system must be able to handle input of location names via speech when the user wants to specify the destination address of the journey. However a location name can be difficult to be recognized by its pronunciation, because it can be a foreign name or a proper name and its pronunciation is not as suggested by its written form. The pronunciation of such a location name cannot be estimated by the recognition engine, which relies on the grapheme to phoneme translation rules in a standard language model.

Another problem with recognizing a location name by pronunciation is the large vocabulary with lots of phonetically similar words. Therefore recognition by spelling is used for location name recognition. However, the engine is prone to make substitution, insertion and deletion mistakes for spelling recognition, and it is always possible for the user to make spelling mistakes. If address entering by speech solely relies on spelling, the usability of the feature will not be optimum. To improve address recognition accuracy we used a strategy which by fully utilized the possible input forms. Spelling of the location name was used as the dominant input form because it is more reliable. It is also taken into account that some location names do have conventional pronunciations and prompt the user for the pronunciation of the location name if the spelling recognition result has a low confidence level due to possible recognition errors or spelling mistakes made by the user.

Although asking for pronunciation as additional information does add to the complexity of the address entering task for the user, usability is sacrificed for recognition accuracy, which makes the feature more competitive to manual address look up. In addition, the user can always skip the steps if they feel it is troublesome (see Section 4.2. for more details) The representation of reliability of a recognition result using percentage accuracy, as mentioned in Section 2, allows the comparison of the confidence of a spoken recognition result and a spelling recognition result. If a location name occurs in both the spelling and spoken recognition results, the levels of confidence or percentage accuracy in the two forms are summed up, giving more confidence on the matching of the name with the utterance.

In order to recognize the location name by pronunciation as a backup, the speech recognition engine needs to be configured with the valid location names as the spoken vocabulary. Although the user will be asked to provide the area name first, which can be used as an address search constraint, the resulting road names within an area can be over 10000. This figure is only for the Auckland region, and may be worse for a European or the American map. With this large vocabulary size, the problems include not only poor recognition accuracy, due to the chance of having similarities between the words, but also the resource consumption of setting up and processing the vocabulary for recognition.

Building a recognition context consisting of 10000 road names takes about 10 seconds processing time on a HP Pocket PC with a 200 MHz processor. Since the road name set is determined at run time, the delay is simply not acceptable for a user oriented application. To reduce the road name set, a partial spelling indexing method is used. With this approach, the spelling of the road name provided in the utterance is stored in a buffer. The first three characters of the spelling are extracted from the spelling and are used as an index string to extract from a subset of road names starting with the same letters. This subset is then configured as the spoken vocabulary for the engine before the user is prompted for the pronunciation of the road name. With the New Zealand map data used in the development of the prototype, using up to the first three letters of the spelling as index is sufficient to narrow the road name set down to less than 30 names. The validity of this approach is based on the assumption that the user does not make spelling mistakes in the initial part of the spelling.

3.1.4 Improved accuracy

With the accuracy improvement strategies, the accuracy testing for the in-car environment is repeated. For the state-dependent vocabulary configuration, the 40 speech commands used throughout the operation of Drive Router are divided into 8 states. Together with result confirmation, the recognition accuracy is increased by 7% to 96%, (see Table 1). With spoken location name as backup, the accuracy for spelled address recognition is increased by 8 % to 88% (see Table 1).

Our results compare favourably with [10] who got 91.3 % word accuracy rate and a 10.1% word error rate, and [11] who got a 7.4% word error rate. The vocabulary for both these studies were the digits. Unlike our study these speech recognition in-car studies were able to use large numbers of speakers to test the system, and the speech recognition platform was not an embedded system

3.2 Improving flexibility

As mention previously, some of the configuration requirements of the ScanSoft system discourage flexible processing of an utterance. Apart from processing an utterance as spoken and spelled words, the partial spelling indexing method requires one spelling to be processed more than once, first as the full spelling of the location name, then to extract the partial spelling. In order to enhance flexible utterance processing, an interactive recognition mode is introduced into the ScanSoft recognition system. With the interactive mode, an utterance is saved in a buffer. The engine is then configured multiple times to perform different types of processing on the utterance.

4 Control feature realization

By specification there were three major control features, including speech-driven menu navigation, speech shortcut commands, and interactive dialogs. In order to achieve portability of the control features and minimise the changes needed in the Drive Router to integrate with the speech control features, the solution was in favour of using the existing control mechanisms and data access interfaces in the Drive Router. Whenever manipulation to low level internal Drive Router data was necessary, additional interfaces were developed into the Drive Router to introduce a high level of abstraction and to avoid direct access to low level details.

4.1 Basic control mechanisms

4.1.1 Graphical user interface control

The menu navigation feature requires a mechanism to control the GUI of the Drive Router. The GUI was constructed using Microsoft Foundation Class Library and is controlled with a typical Windows message system. Windows messages containing control information, such as a button click or a set focus event, are dispatched in

the main message loop of the receiving application, and are directed to the control component. The control component then responds to the event. With the knowledge of the identifiers of the available menu items in the Drive Router, the menu navigation control feature can be achieved using this mechanism. As the Drive Router uses the same system for GUI control, and the mechanism is applicable to any Windows-based applications, including PC based Windows, Windows CE and Pocket PC applications, adopting this mechanism allows the menu navigation feature to be portability and platform independent.

4.1.2 Accessing internal functions

Shortcut behaviour essentially requires automation of internal data processing and events triggering. The realization of these behaviours can be done by calling internal Drive Router functions. Since the Drive Router implementation is object-oriented, the functions can be accessed via an object of the class in which the functions are defined.

4.1.3 Sharing data

As one of the desired features, the user should be able to specify a destination address using speech. Within the Drive Router system, the address data can be extracted from its map data engine, and a set of current location results is kept internally. The Drive Router map data engine has a global interface available to any external module, but the current location result data structure was originally only used by the graphical user interface layer of the Drive Router, which accepts and analyzes manual input of address via the virtual key board. Since the speech interface is an additional plug-in to the system, it is desired that the address input from both input mechanisms valid at the same time. Therefore an interface is developed to allow access to the internal current location result data from an external module. The resulting system enables the Drive Router to have updated location input results from both the speech and manual address input interfaces.

4.2 Dialog management system

The dialog management system is designed as an advanced control component to achieve interactive dialogs between the Drive Router and the user. The design aims at achieving flexibility for modification, expansion and maintenance of a dialog. Usability issues are also considered.

4.2.1 State-oriented design

An interactive dialog can be viewed as a sequence of question and answer pairs. Each pair is a task of getting a particular type of information from the user. The dialog can be very complex if there are a large number of tasks and the sequence of them depends on the information provided by the user, which is exactly the case for a dialog that handles user information intelligently. In recent years, the Extensible Markup Language (XML) has been frequently used in the development of interactive dialogs [12]. The language allows the dialog to be dynamically configured and provides a simple interface for the developer to modify the dialogs. However, an XML parser must be developed and run with the system to process the dialog specification described in the XML file, which introduces unnecessary overhead.

As a lightweight alternative to achieve a simple and flexible solution with the desired functionalities, the dialog was modelled as a finite state machine. Figure 2 shows simplified address entering dialog state diagram. Each state, represented by the boxes, performs a subtask of the dialog, which involves prompting for the type of information expected, getting the user response to the prompt, and determining the next desired state. The links between the tasks, represented by the arrows, are modelled as the state transitions, which depend on the current state and the user response. The transitions can be executed by the dialog management system, which oversees all the states. This model allows the order of tasks to be easily and dynamically arranged based on the user response, as well as allowing easy modification and expansion of the dialog, which simply involves the addition of new state objects and the possible transitions from the state.

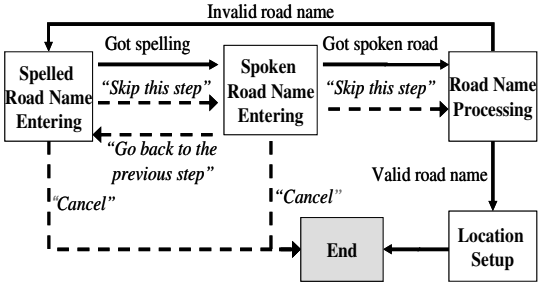


Figure 2: Finite state machine for a dialog

4.2.2 Usability

There are two interaction styles a dialog can adopt. They are the system-driven style and the user-driven style. The system-driven style is necessary for an unconventional dialog, in which the user has little knowledge about what information is required [13]. The address entering dialog, which requires the user to provide spoken and spelled area and road names, can be categorized as such systems. The user-driven mode, on the other hand, allows the system to be user-friendly. For usability, every dialog is designed to have a multi-initiative style, which is the combination of the two [13]. The flow of the dialog is mainly driven by the system, as indicated by the solid arrows in Figure 2, with the addition of several transition commands, as indicated by the dotted arrows in Figure 2, to allow the user to have certain level of control of the flow and to change the current subject. With the development of the interactive recognition mode (see Section 3), which allows one utterance to be processed as different types of speech, the user is able to place a spoken transition command, even when the system is expecting the spelling of a location name.

Features have been added to improve the responsiveness of the dialogs. A timeout event is triggered when there is no response from the user for a certain period of time and the prompt is repeated as a reminder. Also when the user is asked to choose an item from a list of available options, such as the similar recognition results in a confirmation dialog, the user does not have to wait till the end of the list until they have a chance to respond. Any valid choice between the option prompts will be accepted and the dialog status will change accordingly.

5 Solution implementation

The prototype solution was programmed using C++. The language is chosen for its object oriented nature, and consequently allows easy modification and maintenance, as well as portability. Besides its natural integration with the Drive Router, the C++ is also supported by many platforms so that the solution is platform independent. The Microsoft Pocket PC 2003 Software Development Kit is used to configure the solution for the Microsoft Pocket PC operating environment. The development is done using Microsoft Embedded Visual C++ 4.0 Integrated Development Environment.

5.1 Solution structure

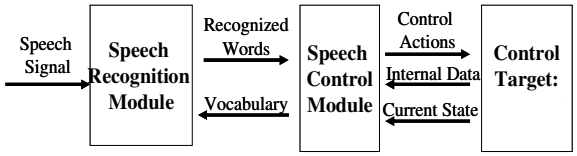


Figure 3: Implementation structure

The solution is implemented in two modules, as illustrated in Figure 3. Each module has a set of APIs to allow external modules to interact with it and configure it. The speech recognition module is a wrapper of the ScanSoft speech acquisition and recognition system. Speech signals are captured and analyzed by the module and recognition results are produced. The interactive recognition mode is also implemented in the module, with a set of functions to allow external modules to activate or deactivate the mode and to decide what type of processing should be done on the utterance. The recognition results gathered in a customized result structure is sent to the speech control module in the form of a Windows message.

The speech control module controls the Control target (the Drive Router), according to the recognition results. The dialog management system residing in the control module enables interactive dialogs for result confirmation and address entering. The implementation also incorporates the state dependent vocabulary approach (see Section 3). The Drive Router was modified to notify the speech control module of its current state, the control module then configures the speech recognition module to activate the vocabulary items related to the valid commands in the state.

6 Testing on integrated system

The implemented solution is integrated with the Drive Router and tested on a HP Pocket PC, with the Intel PXA 255 200 MHz processor and 64 Mb RAM. The device is running the Microsoft Pocket PC 2003 Premium Operating System. On correct recognition of the speech commands, the desired menu navigation or shortcut events are triggered. The confirmation dialog is activated when the user has spoken a phrase that is confusable with other words in the active state. The peak memory

consumption overhead with the speech enabled features is 4 Mb, mainly due to the recognition processing. The major performance limitation of the solution is the processing time. On average, the response time for menu navigation and shortcut commands is 2 seconds, and the maximum response time for address recognition can be up to 6 seconds. This is mainly due to the frequent reconfiguration of the recognition engine at run time. The most time consuming parts of the configuration is the destruction of the engine.

7 Discussion

The accuracy of speech recognition ultimately determines the functionality of the speech-driven control features. More tests need to be done on the performance of the ScanSoft recognition system, especially its ability to handle different speakers. Throughout the project, a lot of effort has gone into getting around the configuration constraints of the ScanSoft system to achieve the desired functionality and accuracy, e.g.. the state-dependent vocabulary approach and the interactive recognition mode. However, these approaches require frequent reconfiguration of the engine at run time, which significantly increases the response time. In order to achieve the desired functionalities and accuracy without compromising the performance, a fundamental solution is a more flexible speech recognition package.

Our investigation focused on the feasibility enabling the Driver Router to have speech-driven control. This we demonstrated, but we did not investigate improving the speech recognition via noise adaptation techniques, microphone placement and/or microphone arrays, Speaker adaptation. The studies [10] and [11] demonstrated that any combination or optimization of these will increase the speech recognition rates .

8 Conclusions

A prototype solution is developed to enable speech-driven control on the Drive Router navigation system. The solution included state-dependent vocabulary configuration, confirming uncertain results with the user, and using both the spelling and the pronunciation of a location name to improve the recognition of an address, and resulted in an accuracy of 96% for recognizing the spoken commands developed in the prototype and 88% for address recognition. Recognition flexibility was also achieved by the development of the interactive recognition mode. The cost of the accuracy and flexibility improvement is the increase in response time due to the constraints of the speech recognition system used.

The desired control features, including speech-driven menu navigation, shortcut commands and interactive dialog for result confirmation and address entering, are developed, with flexibility and usability taken into consideration.

9 References

[1] Dusan, S., Gadbois, G, and Flanagan, J. “Multimodal Interaction on PDA’s Integrating Speech and Pen Inputs”, Proceedings of EUROPSPEECH, Geneva, Switzerland, pp: 2225-2228, 2003

[2] Filali, K., Li, X., and Bilmes, J. "Data-Driven Vector Clustering for Low-Memory Footprint ASR", International Conference on Spoken Language Processing, Denver, Colorado, 2002

[3] Comerford, L., Frank, D., Gopalakrishnan, P., Gopinath, R., and Sedivy, J., “The IBM Personal Speech Assistant”, Proc. of the ICASSP, 2001.

[4] Huang, X. et al., “MIPAD: A Multimodal Interaction Prototype”, Proc. of the ICASSP, 2001.

[5] ScanSoft VoCon 3200Software Development Kit Version 2.0 Developer’s Guide. (2004). ScanSoft Inc.

[6] Lopez, G. and Vanpoucke, P. (2004). “ScanSoft AudioIn Component – AudioIn API”, Version 2.0. ScanSoft Inc.

[7] Automatic Speech Recognition at CSLU. (2003). Center for Spoken Language Understanding. Retrieved from http://cslu.cse.ogi.edu/asr/ on 1st, September, 2005.

[8] Loizou, P., and Spanias, A., “High-Performance Alphabet Recognition”, IEEE Transactions on Speech and Audio Processing, Vol 4, No.6, November 1996.

[9] Hansen, J.H.L, Zhang, X.,Akbakac, M., Yapenal,U., Pellom, B, and Ward, W. “Robust Speech Processing for In-Vehicle Voice Navigation Systems,” ICA-2004: Inter. Congress on Acoustics, vol. 4, pp. 2603-2606, Kyoto, Japan, April 2004.

[10] U. Yapanel, X. Zhang, J.H.L. Hansen, "High Performance Digit Recognition in Real Car Environments”, ICSLP-2002:Inter. Conf. on Spoken Language Processing, vol. 2, pp. 793-796, Denver, CO USA, Sept. 2002

[11] Delphin-Poulat, L. “Robust speech recognition techniques evaluation for telephony server based in-car applications”, IEEE International Conference on Acoustics, Speech, and Signal Processing, Proceedings. (ICASSP '04). 65-8 vol.1, May 2004.

[12] Voice XML Forum. (2004). IEEE Industry Standard and Technology Organization. Retrieved from http://www.voicexml.org/, on 2nd September, 2005.

[13] Stallard, D. “Flexible Dialogue Management in the Talk ‘n’ Travel System”, International Conference on Spoken Language Processing, Denver, Colorado, pp:2693-2696, 2002

Combining Confidence Scores with Contextual Features for Robust Multi-Device Dialogue*

Lawrence Cavedon
National ICT Australia, Victoria Research Lab
and CS&IT, RMIT University
Melbourne VIC, Australia
lawrence.cavedon@nicta.com.au

Matthew Purver, Florin Ratiu
CSLI, Stanford University
Cordura Hall, 210 Panama St. Stanford
CA 94305, USA
{mpurver,fratiu}@stanford.edu

Abstract

We present an approach to multi-device dialogue that evaluates and selects amongst candidate dialogue moves based on features at multiple levels. Multiple sources of information can be combined, multiple speech recognition and parsing hypotheses tested, and multiple devices and moves considered to choose the highest scoring hypothesis overall. The approach has the added benefit of potentially re-ordering n-best lists of inputs, effectively correcting errors in speech recognition or parsing. A current application includes conversational interaction with a collection of in-car devices.

1 Introduction

In this paper, we describe recent enhancements to the *CSLI Dialogue Manager* (CDM) infrastructure to increase robustness, in particular in (but not exclusive to) multi-device settings. Dialogue contributions may be processed using multiple information sources (e.g. deep syntactic parsing and shallow topic classification), scored at multiple levels (e.g. acoustic, semantic and context-based), and bid for by multiple agents, with the overall highest-confidence bid chosen.

The CDM provides a multi-device infrastructure, with customization to new applications and addition of plug-and-play devices eased by a declarative dialogue-move scripting language (Mirkovic and Cavedon, 2005). However, deciding which device an utterance is directed at is not always straightforward. One of our current application areas is a conversational interface to in-car devices, including entertainment, restaurant recommendation, navigation and telematic systems (Weng et al., 2004); in such an environment, a request such as *"Play X"* might be directed at an MP3 player or a DVD player. Eye-gaze (useful in multi-human dialogue) is not available, and we cannot rely on explicit device naming. One option is to use the resolution of NP arguments as disambiguating information (in our *"Play X"* example, whether X is a song or a movie). However, the NP-resolution process itself is often device-specific (see below), preventing NPs from being properly resolved until device has been determined.

Our proposed solution, inspired by approaches to multi-agent task allocation such as Contract Net (Smith, 1980), is to allow all devices to perform shallow processing of the incoming utterance, each producing multiple possible candidate dialogue moves. Potential device-move combinations are then scored against a number of features, including speech-recognition and parse confidence, discourse context, current *device-under-discussion*, and NP argument analysis. The device associated with the highest-scoring dialogue move is given first option to process the utterance. A disambiguation question may be generated if no device is a clear winner, or a confirmation question if the winning bid is not scored high enough.

Device choice, move choice, and selection of best ASR/parser hypothesis are thereby made simultaneously, rather than being treated as independent processes. As well as allowing for principled device identification, this has the benefit of scoring hypotheses on the basis of multiple information sources, including context. The highest scoring result overall may not correspond to the highest-confidence result from the ASR or parser n-best list alone, but n-best lists are effectively re-ordered based on device and dialogue context, allowing parsing errors such as incorrect PP-attachment to be automatically corrected. Confirmation and clarification behaviour can also be governed not only by ASR or parse confidence, but by the overall score.

* This work was performed while all the authors were employed at CSLI, Stanford University, and was partially supported by the US government's NIST Advanced Technology Program.

Proceedings of the Australasian Language Technology Workshop 2005, pages 233–240,
Sydney, Australia, December 2005.

Related Approaches Rayner et al. (1994) combine speech recognition confidence scores with various intra-utterance linguistic features to re-order n-best hypotheses; Chotimongkol and Rudnicky (2001) also include move bigram statistics. Walker et al. (2000) use similar feature combination to identify misrecognised utterances. More recently, Gabsdil and Lemon (2004) also include pragmatic information such as NP resolution, and simultaneously choose from an n-best list while identifying misrecognition. They also divide misrecognised utterances into two overall confidence ranges, one for outright rejection and one for confirmation/clarification. Similarly Gabsdil and Bos (2003) combine acoustic confidences with semantic information, and Schlangen (2004) with bridging reference resolution, in order to allow clarification on an integrated basis. All of these approaches assume a single-device setting and hence no ambiguity of move type once the correct word string or parse has been identified. Here we extend these approaches to allow a principled choice of move/device pairing.

2 Background

2.1 Dialogue Manager Architecture

Our focus is on *activity-oriented dialogue*, discussing tasks or activities that are jointly performed by a human and one or more intelligent devices or agents. By "joint activity", we mean that the human participates in specifying the activity, clarifying requests, interpreting observations, and otherwise supporting the agent in the performance of the activity. Systems engaging in such dialogue characteristically require deep knowledge about the task domain and the devices/agents they provide access to, in order to know what information is critical to the tasks, and know what information about task performance is appropriate to provide to the user. CSLI has been developing activity-oriented dialogue systems for a number of years, for applications such as multimodal control of robotic devices (Lemon et al., 2002), speech-enabled tutoring systems (Fry et al., 2001), and conversational interaction with in-car devices (Weng et al., 2004).

The dialogue system architecture (Figure 1) centers around the CSLI Dialogue Manager, which can be used with various different external components: speech-recognizer, NL parser, NL generation, speech-synthesizer, as well as connections to external application-specific components such as ontologies or knowledge bases, and the dialogue-enabled devices themselves. Clean interfaces and representation-neutral processes enable the CDM to be used relatively seamlessly with different NL components, while interaction with external devices is mediated by *Activity Models*, declarative specifications of device capabilities and their relationships to linguistic processes.

The CDM uses the *information-state update* (ISU) approach to dialogue management (Larsson and Traum, 2000). The ISU approach extends the more traditional finite-state-based approaches used for simple dialogues (in which dialogue context is represented as one of a finite number of states, and each dialogue move results in a state transition), maintaining a richer representation of information-state. This includes the dialogue context as well as e.g. device and activity status, together with a set of update rules defining the effect of dialogue moves on the state (e.g. adding new information and referents for anaphora resolution, and triggering new tasks, activities and system responses). This approach allows more complex dialogue types with advanced strategies for context-dependent utterance interpretation (including fragments and revisions), NP resolution, issue tracking and improved speech-recognizer performance (Lemon and Gruenstein, 2004).

2.2 The CSLI Dialogue Manager

Generic ISU toolkits (e.g. TrindiKit (Traum et al., 1999), DIPPER (Bos et al., 2003)) provide general data structures for representing state and a language for specifying update rules, but the specific state and rules used are left to the individual application. The CDM is a specific implementation of an ISU dialogue-management system, providing data structures and processes for update specifically designed as suitable to activity-oriented dialogue, but adaptible to different applications and domains.

The two central components of the CDM information state are the *Dialogue Move Tree* (DMT) and the *Activity Tree*. The DMT represents the dialogue context and history, with each dialogue move represented as a node in the tree, and incoming moves interpreted in context by attachment to an appropriate open parent node (for example, `WhAnswer` moves attach to their corresponding `WhQuestion` nodes). This tree structure specifically supports *multi-threaded, multi-topic* conversations (Lemon et

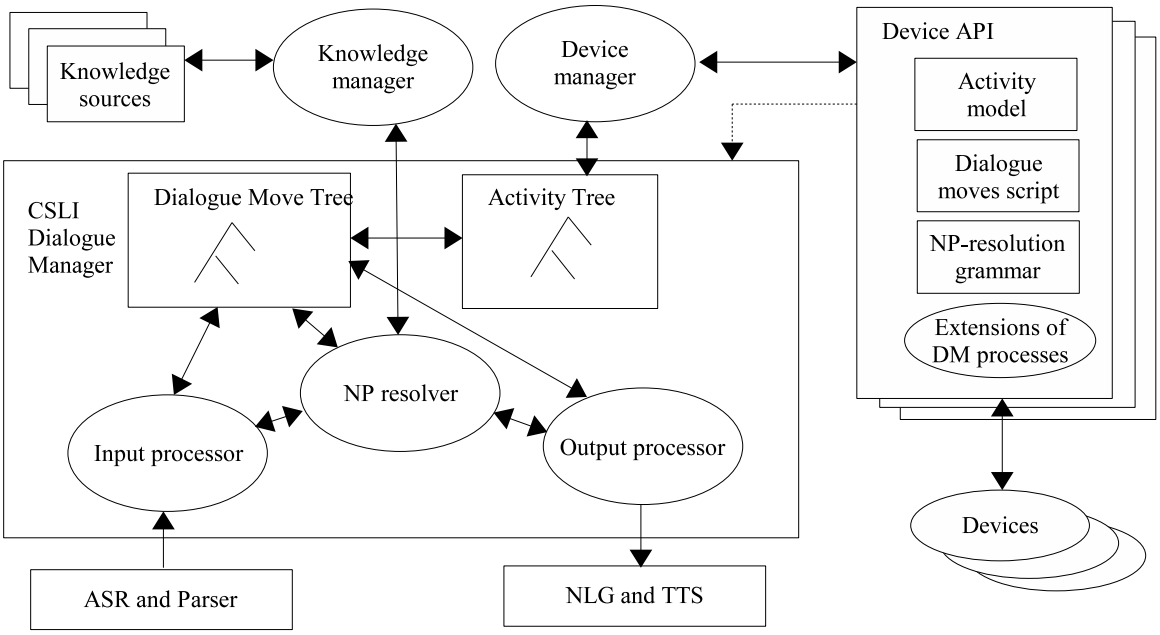


Figure 1: Dialogue System Architecture

al., 2002), with branches representing topics or threads: a dialogue move that cannot attach itself to the most recent active node may instead attach to another open branch (corresponding to a resumed conversation) or open a new branch (a new conversation thread) by attaching itself to the root node. The DMT also serves as context for interpreting fragments, multi-utterance constructs, and revisions, and provides discourse structure for tasks such as NP-resolution. In tandem, the Activity Tree manages the underlying activities, fully instantiating new activities via their Activity Models (e.g. resolving NP referents or spawning subdialogues to fill missing arguments), editing existing ones as a result of revisions or corrections, and monitoring their execution (possibly generating system moves notifying completion or failure).

Other data structures that are part of the information state include: the *salience list* (NPs and their referents for anaphora resolution); *multimodal input buffers* (semantic interpretations of GUI events); and the *system agenda* (potential system outputs scheduled by the dialogue manager). See (Lemon et al., 2002) for details.

2.3 Dialogue Move Scripting

In early versions of the CDM, dialogue moves were coded completely programmatically (in Java). While libraries of general-purpose dialogue moves (e.g. `Command`, `WhQuestion`, etc.) were re-used wherever possible, customization to new domains generally required significant programming effort in defining both new dialogue moves and their effects, and processes such as reference resolution. More recently, Mirkovic and Cavedon (2005) describe a dialogue-move scripting language designed to expedite customization to new domains. Each script serves a number of purposes:

1. hierarchical definition of dialogue moves, allowing inheritance and re-use of existing moves, while allowing customization to a specific domain;
2. mapping of utterance representations to appropriate dialogue moves, including argument values for devices' activity models;
3. definition of attachment rules for information-state update;
4. dialogue move-specific specification of output to be generated, for disambiguation or requests for required information.

Listing 1 shows the skeleton of a sample dialogue-move script for a `play Command` move for an MP3 player. The specific syntax of the *Input* and *Output* fields can be ignored for now: they simply match the interfaces of the parser and generator respectively. Variables in the dialogue move script correspond to variables in the Activity Model (AM) for the corresponding device. The AM for the MP3 device contains

```
User Command:play {  // inherits from generic Command dialogue move
   Description "play something"
   Input {           // templates for matching parser output
      // full parse match: ``play/start X''
      1.0   SYN{ s( features(mood(imperative)), predicate(#play/vb|#start/vb),
                 ?arglist(obj:_playable-object,?sbj:*)) }
      // full parse match: ``I want to play/hear X''
      1.0   SYN{ s( features(mood(indirect)), predicate(#play/vb|#hear/vb),
                 ?arglist(obj:_playable-object,?sbj:*)) }
      // topic classifier match
      0.1   TOPIC{ play_item }
      // topic classifier match with argument
      0.25  AND{ TOPIC{ play_item },   SYN{ arglist(obj:_playable-object,*) } }
   ... }
   Producing {       // templates for system output: questions
      System WHQuestion:disambiguate
      System WHQuestion:fill:play:_playable-object {
         Output {avs (e1 / play :question (q1 / what) :agent I)
         }
   ... }             // templates for system output: reports
         CloseOn System Report:play:playing {
            Output {avs (e1 / play :patient (p1 / [song]) :aspect continuous)
         }
   ... }
... }
```

Listing 1: Sample dialogue move script for a `play Command` for an MP3 device

a `play` operation with a (required) *playable-object* argument. When an incoming utterance matches an *Input* template from Listing 1, the *playable-object* variable is filled by unification, and resolved to an object from the device's domain which then fills the corresponding slot in the activity. For details, see (Mirkovic and Cavedon, 2005).

2.4 Multi-Device Dialogue

The CDM has also been extended to multi-device dialogue, with the scripting approach allowing easy dynamic plug-and-play specification of new "dialogue-enabled" devices. Note that this does not constitute multi-party dialogue: interaction is still mediated by a single dialogue manager, between a user and a *Device Manager* with which devices register themselves. However, the plug-and-play requirement (necessitated by the in-car application (Weng et al., 2004)) has resulted in important extensions to the dialogue management infrastructure.

Mirkovic and Cavedon (2005) describe a framework for encapsulating devices with information required to "dialogue-enable" them. Each device has associated with it the following components:

1. a set of dialogue-move scripts;
2. an Activity Model describing any device functionality accessible by dialogue;
3. a device-specific ontology and knowledge base (KB);
4. rules for device-specific NP-resolution.

Any significantly different forms of interaction requiring device-specific dialogue management processes must still be specified as new Java classes (referred to as *DM process extensions* in Figure 1), but in general the above four components contain the device-specific information required for dialogue-enabling new devices.

Note that NP resolution rules are included in the device definition; while pronoun resolution tends to be domain-independent, resolving definite descriptions and demonstratives is often device-dependent, and resolving named referents often requires constructing appropriate queries to a device-specific knowledge-base.

Devices can now be added dynamically to the DMT, registering themselves with the Device Manager and becoming associated with their own nodes to which new conversation threads can attach; "current device" becomes part of the information-state and interpreting incoming utterances is performed in this context.

In this context, device selection—determining which device an utterance is associated with—becomes a further complication: an utterance may (on the surface) be potentially applicable to multiple devices: e.g. "Play X" could be applicable to either an MP3 player or a DVD

player. Our original proposal was to create a dialogue move consistent with each such device and then score its applicability based on other factors, e.g. ability to resolve the object reference (the MP3 player would resolve a song-name, the DVD player a movie name). The rest of the paper generalises this approach to a wider range of possible disambiguities, involving a greater number of scoring features, and results in more interesting behaviours than simple device-disambiguation.

3 Multiple Interpretation Methods

The first new extension to the CDM described here is the use of multiple information sources in parallel to classify dialogue move type and produce an activity-specific representation. In most systems (and previous incarnations of the CDM) a single interpretation mechanism is chosen which is best suited to the application at hand, be it e.g. an open-domain statistical parser, a domain-specific constraint-based grammar, or keyword-spotting techniques. We extend this approach here to allow arbitrary multiple interpretation mechanisms, each producing its own (independent) interpretation hypothesis and associated confidence. In the current application, we use both a statistical parser producing relatively deep dependency structures, and a shallow maximum-entropy-based topic classifier.

Dialogue move scripts, such as the one sketched in Listing 1, are used to construct instantiations of candidate dialogue moves for a device, based on incoming user utterances (and planned system outputs, although we focus on the former here). This is governed by the *Input* field for each move type, which specifies a set of patterns: when an utterance representation matches an *Input* pattern, a candidate node of the appropriate type can be created. As Listing 1 shows, patterns can now be defined in terms of interpretation method as well as the interpreted form itself: `SYN` patterns match the output of the statistical parser, `TOPIC` patterns match the output of the topic classifier, while `AND` patterns match combinations of the two. Further general pattern types are available (e.g. `LF` for semantic logical forms, `STRING` for surface string keyword-matching) but are not used in the current application.

Each pattern is associated with a weight, used in the overall move scoring function described in Section 4 below. This allows moves created from matches against deep structure to be scored highly (e.g. `SYN` patterns in which predicate and arguments are specified and matched against), shallow matches to be scored low (e.g. simple `TOPIC` matches), and combined matches to have intermediate scores (e.g. a combination of an appropriate `TOPIC` classification with a `SYN` parser output containing a suitable NP argument pattern). Depending on other elements of the scoring function (e.g. the ASR confidence associated with the hypothesised string being tested) and on competing move hypotheses, low scores may lead to clarification being required (and therefore clarification will be more likely when only low-scoring (shallow) patterns are matched). Behaviour can therefore be made more robust: when deep parsing fails, a shallow hypothesis can be used instead (clarifying/confirming this specific hypothesis as necessary depending on its confidence) rather than resorting to a rejection or general clarification. Scores are currently set manually and determined by testing on sample dialogues; future work will examine learning them from data.

4 Dialogue Move Selection

In the general case, multiple possible candidate dialogue moves will be produced for a given user utterance, for a number of reasons:

1. multiple hypotheses from ASR/parser output;
2. multiple interpretation methods (deep parsing vs. shallow classification);
3. multiple possible move types for a candidate interpretation;
4. multiple antecedent nodes (active dialogue threads), including multiple devices, for a particular move type.

These are not independent: it is important to consider all factors simultaneously, to allow an integrated scoring function for each candidate and thus consider the best overall. The skeleton algorithm for instantiating and selecting a dialogue move is therefore as follows:[1]

[1]Note that we will not create $O \times N \times M$ candidates: only a subset of script entries (if any) will match for each node and n-best entry.

```
foreach open node O
   foreach n-best list entry N
      foreach matching script entry M
         create candidate move
score all candidates
if (score(top) >> score(second))
then
   select top candidate
else
   generate question to disambiguate
if (score(selected-node) < threshold)
    generate question to confirm
```

The interesting aspect of the above process is the scoring function. Dialogue-move candidates are scored using a number of weighted features, ranging from speech-recognizer confidence, through to pragmatic features such as the "device in focus" and recency of the DMT node the candidate would attach to. The full list of features currently considered is shown in Table 1. Note the inclusion of features at many levels, from acoustic recognition confidences through syntactic parse confidence to semantic and pragmatic features.

4.1 Reordering n-best candidates

This integrated scoring mechanism therefore allows n-best list input to be re-ordered: dialogue-move candidates are potentially instantiated for each n-best list entry and the highest-scoring candidate chosen. While the n-best list rank and confidence are factors in the overall score, other features may outweigh them, resulting in an initially lower-ranked n-best entry becoming the highest-scoring dialogue move.

Evaluation so far has been limited to initial testing on a manually constructed set of test inputs, using only a subset of the features: those shown italicised in Table 1 are not currently available due to either implementational issues (for full domain referent resolution and KB queries) or lack of domain data (for move bigram frequencies). Our test set includes 400 sentences, of which 300 have been used in training the statistical parser and 100 are unseen variations; it currently covers only utterances related to a single device (a restaurant recommendation system) and does not include speech recognition hypotheses (we are therefore testing parse n-best reordering only). We are currently working towards evaluation on a full set of features, with user-generated multi-device speech input.

However, even with the restricted set of features, preliminary testing on this set shows encouraging results: the percentage of sentences for which the correct parse is chosen increases from 90% to 94%, a 41% reduction in error with several common parse errors being corrected. One example is incorrect PP-attachment (a notoriously difficult challenge for statistical parsers). The example below (from a restaurant recommendation scenario), shows the top two n-best list entries for a sentence as produced by our statistical parser:

```
1. how about [a restaurant
              [in Grant]] [on Mayfield]
2. how about [a restaurant
              [in Grant] [on Mayfield]]
```

Here, the second is lower-ranked but correct, taking both PPs as modifying *restaurant*, while the first treats only one as modifying *restaurant*, one as a sentential modifier. As the second allows two database-query constraints to be filled (city and street name), and the first just one, this boosts its overall score enough to overcome its lower parse confidence, and it is selected and used in DMT attachment. Similar improvements are gained with nominal modifiers:

```
1. how about [a
           [[cheap] chinese] restaurant]
2. how about [a
           [cheap] [chinese] restaurant]
3. how about [a
           [cheap chinese] restaurant]
```

Here the second is correct, treating *cheap* and *chinese* as both independently modifying *restaurant*; the first takes *cheap* as modifying *chinese*, and the third takes *cheap chinese* as a single multi-word unit. Again, as the second fills two database-query constraints (price level and cuisine type), its overall score becomes highest. Evaluation of the improvement achieved is currently in progress.

4.2 Move type comparison

The scoring function for feature combination is currently manually defined. When comparing between candidate moves of the same type, this is relatively straightforward, although hardly trivial and inherently done to a high extent by subjective expertise. However, it becomes much less straightforward when comparing candidates of different types, as some move types and some DMT attachment contexts will allow only a subset of the features to have meaningful values. However, comparison between move types is essential, as two ASR hypotheses with

Recognition features:	recognition and parse probabilities; recognition and parse n-best ranks;
Semantic features:	topic classification for the parse (with score); for dialogue moves spawning activities: - number of slots filled by input pattern; - *number of resolved/unresolved slots after NP resolution;* - *number of ambiguously resolved slots after NP resolution;* for queries about database objects: - set of constraints sent to the knowledge base; - *cardinality of the set of knowledge base query results;*
Contextual features:	current most active node; current activity; position and recency of the parent node in the active node list; *bi-gram frequencies of the dialogue moves:* - *DMT attachments - pairs of child and parent node types;* - *pairs of chronologically consecutive user nodes.*

Table 1: Move Scoring Features

similar recognition scores may have very different possible move types:

1. Command: "Play a rock song by Cher"
2. Query: "What rock songs are there?"

We are therefore currently investigating the use of machine learning techniques to improve on our current manual definitions. With annotated data the optimal weights of a scoring function that combines all the features can be automatically learned (see (Gabsdil and Lemon, 2004)).

4.3 Dialogue-move disambiguation

In order for a winning bid to be unambiguously accepted, its score must exceed the next highest score by more than a predefined threshold. If not, we take the choice of winning bid to be within our margin of error, and the dialogue manager asks a disambiguating clarification question. For example, if the pair of sentences in the previous section result in hypothesis dialogue moves with scores within the margin of error, then the dialogue manager generates a question of the form:

> "Did you want to play a rock song by Cher or did you ask about rock songs?"

Alternatively, in some cases there may be a clear highest-scoring bid (i.e. one of high *relative* confidence) which is itself of low *absolute* confidence. In such cases, rather than act on the move unconditionally we ask the user for clarification. If the score is below a certain confidence threshold T_1 we treat the highest bid as a reasonable hypothesis, but ask for confirmation of the intended move; following the previous example, this would result in a question such as:

> "I'm not sure I understood that. Did you want to play a rock song by Cher?"

If the score is below a second critical minimum threshold T_2 we take this as a failure in interpretation, and prompt for general clarification. As even the best hypothesised move is likely to be incorrect in this case (being of such low confidence), asking for specific confirmation is likely to be counter-productive or annoying (see e.g. (San-Segundo et al., 2001)).

Threshold values are currently specified as part of dialogue-move definitions; a future direction is to automatically learn optimal values for the thresholds.

5 Discussion and Conclusions

We have described a number of strategies implemented in the CSLI Dialogue Manager to more robustly handle ambiguous or misunderstood utterances, and low-confidence interpretations. Features from multiple sources of evidence are combined to rate the possible dialogue move candidates as interpretations of a user utterance. Features include confidence scores from ASR and parser, as well as semantic and pragmatic criteria, and measures related to the dialogue context itself. As well as selecting dialogue move, in our multi-device setting the approach has the benefit of selecting the device being addressed. Although we have not yet performed a full evaluation of the ef-

ficacy of this approach, we have observed several examples of the n-best list of inputs being (correctly) re-ordered—i.e. after misclassification by the statistical parser, the candidate dialogue-move corresponding to the correct (though lower-confidence) parse can still be selected. We are currently gathering data in order to provide a concrete evaluation.

Confidence thresholds (upper and lower bounds) set by the dialogue designer specify the levels at which a candidate move is rejected, requires explicit confirmation by the user, or simply accepted. Future work includes automatically learning optimal values for these thresholds and optimal weights on the features for scoring candidate dialogue-moves, applying the techniques of e.g. Gabsdil and Lemon (2004) to our multi-device setting.

References

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. DIPPER: Description and formalization of an information-state update dialogue system architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*.

A. Chotimongkol and A. Rudnicky. 2001. N-best speech hypotheses reordering using linear regression. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH)*.

J. Fry, M. Ginzton, S. Peters, B. Clark, and H. Pon-Barry. 2001. Automated tutoring dialogues for training in shipboard damage control. In *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*.

M. Gabsdil and J. Bos. 2003. Combining acoustic confidence scores with deep semantic analysis for clarification dialogues. In *Proc. 5th International Workshop on Computational Semantics (IWCS-5)*.

M. Gabsdil and O. Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proc. 42nd Annual Meeting of the ACL*.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6.

O. Lemon and A. Gruenstein. 2004. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction*, 11(3).

O. Lemon, A. Gruenstein, and S. Peters. 2002. Collaborative activities and multi-tasking in dialogue systems. *Traitement Automatique des Langues*, 43(2).

D. Mirkovic and L. Cavedon. 2005. Practical plug-and-play dialogue management. In *Proceedings of the Annual Meeting of the Pacific Association of Computational Linguistics (PACLING)*.

M. Rayner, D. Carter, V. Digalakis, and P. Price. 1994. Combining knowledge sources to reorder n-best speech hypothesis lists. In *Proceedings of the ARPA Human Language Technology Workshop*.

R. San-Segundo, J. M. Montero, J. Ferreiros, R. Córdoba, and J. M. Pardo. 2001. Designing confirmation mechanisms and error recover techniques in a railway information system for spanish. In *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*.

D. Schlangen. 2004. Causes and strategies for requesting clarification in dialogue. In *Proc. 5th SIGdial Workshop on Discourse and Dialogue*.

R. G. Smith. 1980. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

D. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio. 1999. A model of dialogue moves and information state revision. In *Task Oriented Instructional Dialogue (TRINDI): Deliverable 2.1*. University of Gothenburg.

M. Walker, J. Wright, and I. Langkilde. 2000. Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *Proceedings of the 17th International Conference on Machine Learning*.

F. Weng, L. Cavedon, B. Raghunathan, D. Mirkovic, H. Cheng, H. Schmidt, H. Bratt, R. Mishra, S. Peters, L. Zhao, S. Upson, E. Shriberg, and C. Bergmann. 2004. A conversational dialogue system for cognitively overloaded users. In *Proc. 8th International Conference on Spoken Language Processing (INTERSPEECH)*.

Automatic Utterance Segmentation in Instant Messaging Dialogue

Edward Ivanovic
Department of Computer Science and Software Engineering
University of Melbourne
edwardi@csse.unimelb.edu.au

Abstract

Instant Messaging (IM) chat sessions are real-time, text-based conversations which can be analyzed using dialogue-act models. Dialogue acts represent the semantic information of an utterance, however, messages must be segmented into utterances before classification can take place. We describe and compare two statistical methods for automatic utterance segmentation and dialogue-act classification in task-based IM dialogue. It is shown that IM messages can be automatically segmented and classified to a very high accuracy using statistical machine learning.

1 Introduction

Dialogue acts are a useful first level of analysis for describing discourse structure as they represent the illocutionary force of utterances such as assertions and declarations. Early work on speech act theory by Austin (1962) and Searle (1979) has been extended in dialogue acts to model the conversational functions that utterances can perform. Table 1 shows an example dialogue with utterance segments and dialogue acts.

As illustrated in Table 1, some messages contain multiple utterances and thus require segmentation before each utterance can be classified as a dialogue act. Once utterances are classified, the dialogue-acts may then be used for subsequent tasks such as machine translation (Tanaka and Yokoo, 1999), dialogue game detection (Levin et al., 1999), and, in the case of spoken dialogue, speech recognition (Stolcke et al., 2000).

Speaker	Message
Sanders	[Hello Customer]$^{\text{CONVENTIONAL-OPENING}}$, [thank you for contacting MSN Shopping]$^{\text{THANKING}}$. [This is Sanders and I look forward to assisting you today]$^{\text{STATEMENT}}$
Sanders	[How are you doing today?]$^{\text{OPEN-QUESTION}}$
Customer	[good]$^{\text{STATEMENT}}$, [thanks]$^{\text{THANKING}}$
Sanders	[How may I help you today?]$^{\text{OPEN-QUESTION}}$

Table 1: An example of the beginning of a dialogue in our corpus showing utterance boundaries and dialogue-act tags in superscript.

Instant Messaging (IM) consists of two or more people typing messages to each other in real time on a line-by-line basis. Although IM dialogue can take place with a group of people simultaneously writing to each other, for the purposes of this study we assume only two-party dialogue. As described in Ivanovic (2005), sequences of words are grouped into three levels: the first level is a *Turn*, consisting of at least one *Message*, which consists of at least one *Utterance*, defined as follows:

Turn: A dialogue participant normally writes one or more message then waits for the other participant to respond, hence taking *turns* in writing messages.

Message: A message is defined as a group of words that are sent from one dialogue participant to the other participant as a single unit. This is usually achieved by typing a message and pressing the Enter key or a 'Send' button on the client program. A single turn can span multiple messages.

Utterance: An utterance can be thought of as one complete semantic unit with respect to dialogue acts. This can be a complete sentence or as short as an emoticon (e.g. ":-)" to smile). Messages contain one

Proceedings of the Australasian Language Technology Workshop 2005, pages 241–249,
Sydney, Australia, December 2005.

or more utterances. Although it is possible to send a message mid-utterance, resulting in utterances spanning messages, so such instances occur in our corpus, which our model assumes. Example utterances, enclosed within brackets, are shown in Table 1.

Most utterance segmentation research to date has focussed on transcribed speech. The aim of speech segmentation, however, is different to that required by dialogue act classification. That is, large-vocabulary speech recognisers segment speech into *acoustic* segments for more efficient processing, using criteria such as non-speech intervals and turn boundaries in dialogue. These methods are not appropriate for IM utterance segmentation because the acoustic segmentation methods rely on the recorded waveform of speech, which does not exist in IM dialogue.

We show that utterance segmentation for dialogue act classification requires very different criteria to transcribed speech segmentation. Our methods for dialogue act utterance segmentation are based on linguistically and statistically motivated approaches.

The rest of this paper is organised as follows. The data collection and dialogue act tag set are described in Section 2. The methods and language models used in our experiment are explained in Section 3. Evaluation techniques we use are in Section 4. Our experimental results and discussion are in Section 5, with the conclusions and future work in Section 6.

2 Data and Dialogue Act Tag Set

Our data was collected in previous work (Ivanovic, 2005) from an online IM-based support service and consisted of nine chat sessions, totalling 550 utterances, 6,500 words, with a mean message length of 10 words. The chat sessions were manually segmented into utterances by one person and used as a gold standard. These utterances were then annotated by three people

Table 2 shows the dialogue act tag set we use, which was also taken from previous work as described in Ivanovic (2005). The tag set was chosen by manually labelling our corpus using tags that seemed appropriate from the 42 tags used by Stolcke et al. (2000), which in turn were based on the Dialog Act Markup in Several Layers (DAMSL) tag set (Core and Allen, 1997).

Tag	Example	%
STATEMENT	I am sending you the page now	36.0
THANKING	Thank you for contacting us	14.7
YES-NO-QUESTION	Did you receive the page?	13.9
RESPONSE-ACK	Sure	7.2
REQUEST	Please let me know how I can assist	5.9
OPEN-QUESTION	how do I use the international version?	5.3
YES-ANSWER	yes, yeah	5.1
CONVENTIONAL-CLOSING	Bye Bye	2.9
NO-ANSWER	no, nope	2.5
CONVENTIONAL-OPENING	Hello Customer	2.3
EXPRESSIVE	haha, :-), grr	2.3
DOWNPLAYER	my pleasure	1.9

Table 2: The 12 dialogue act labels with examples and frequencies given as percentages of the total number of utterances in our corpus.

A Kappa analysis used to compare inter-annotator agreement normalised for chance (Siegel and Castellan, 1988), resulted in a value of 0.87 with 89% agreement (Ivanovic, 2005). A Kappa statistic of 0.8 and above is considered a satisfactory indication that our corpus can be labelled reliability using our tag set (Carletta, 1996).

A complete list of the 12 dialogue acts we used is shown in Table 2 along with examples and the frequency of each dialogue act in our corpus.

3 Methods

Our first goal was to determine which features obtained from IM transcripts would be useful in detecting utterance segments within messages. The data available from IM chat transcripts are the *speaker*, *message text*, and *time stamp* of each message. Unlike regular written prose, IM chats are often very informal—omitting usual punctuation such as commas, periods, question marks, and initial capital letters for proper names and new sentences. Spelling mistakes, acronyms for common phrases, and ungrammatical messages are also quite common.

The observation that utterances in our data do not cross message boundaries allows us to focus on segmenting one message at a time. We use two ap-

A: [$_{INTJ}$ hello $_{UH}$] [$_{NP}$ customer $_{NN}$] , $_{O}$ <s> [$_{VP}$ thank $_{VB}$] [$_{NP}$ you $_{PRP}$] [$_{PP}$ for $_{IN}$] [$_{VP}$ contact $_{VBG}$] [$_{NP}$ Msn $_{NNP}$ Shopping $_{NNP}$] . $_{O}$ <s> [$_{NP}$ this $_{DT}$] [$_{VP}$ be $_{VBZ}$] [$_{NP}$ Sanders $_{NNP}$] [$_{O}$ and $_{CC}$] [$_{NP}$ I $_{PRP}$] [$_{VP}$ look $_{VBP}$] [$_{ADVP}$ forward $_{RB}$] [$_{PP}$ to $_{TO}$] [$_{VP}$ assist $_{VBG}$] [$_{NP}$ you $_{PRP}$] [$_{NP}$ today $_{NN}$] . $_{O}$
A: [$_{ADVP}$ how $_{WRB}$] [$_{O}$ be $_{VBP}$] [$_{NP}$ you $_{PRP}$] [$_{VP}$ do $_{VBG}$] [$_{NP}$ today $_{NN}$] ? $_{O}$
B: [$_{ADJP}$ good $_{JJ}$] , $_{O}$ <s> [$_{NP}$ thanks $_{NNS}$]

Figure 1: Sample tagged and chunked data.

proaches to segment the messages: Hidden Markov Models (HMMs) and a probabilistic model based on parse trees. We discuss each of these in turn.

3.1 HMM Method

In the absence of reliable punctuation cues, we looked at approaches based on the available lexical information. One such method was to use an HMM to find the most likely segment boundaries. We experimented with three versions of the HMM approach, based on sequences of: (i) lemmas, (ii) part of speech tags, and (iii) head words of chunks.

The rationale behind using chunks is that the number of possible segments is reduced since utterance boundaries do not lie within chunks. The data was assigned POS tags and segmented into chunks via the FNTBL Toolkit (Ngai and Florian, 2001), which is an efficient implementation of Eric Brill's Transformation-based learning algorithm (Brill, 1995). Lemmatisation on our corpus was performed using the morphological tools described in Minnen et al. (2001).

Figure 1 illustrates some characteristics of the data. Utterance boundaries are marked by <s> tags, chunk boundaries are enclosed within brackets, and words' POS tags are shown in subscript after the word. The actual chunks in the data use IOB tags similar to that described in Ramshaw and Marcus (1995).

We first trained an n-gram statistical language model with add-one smoothing and Katz backoff (Katz, 1987) to hypothesize the most probable locations of utterance boundaries for each individual message. The resulting segmentations were then evaluated using the WindowDiff metric as described in Section 4.

Elements used to represent the segments were lemmas, POS tags, and chunks. Segment beginnings in our training data were marked with a <s> tag. This allowed each element to be in one of two states: S or NO-S depending on whether it had a <s> tag before it. We build two probability distributions P_S and P_{NO-S} representing the probability that token t_k is at the beginning of a segment or not, respectively. Using this state information permits us to use an HMM with the following forward computation for the likelihoods of the states at each position k as described by Stolcke and Shriberg (1996):

$$\begin{aligned} P_{NO-S}(t_1...t_k) &= P_{NO-S}(t_1...t_{k-1}) \times \\ &\quad p(t_k|t_{k-2}t_{k-1}) \\ &\quad + P_S(t_1...t_{k-1}) \times \\ &\quad p(t_k|\texttt{<s>}t_{k-1}) \\ P_S(t_1...t_k) &= P_{NO-S}(t_1...t_{k-1}) \times \\ &\quad p(\texttt{<s>}|t_{k-2}t_{k-1})p(t_k|\texttt{<s>}) \\ &\quad + P_S(t_1...t_{k-1}) \times \\ &\quad p(\texttt{<s>}|\texttt{<s>}t_{k-1})p(t_k|\texttt{<s>}) \end{aligned}$$

where t is a lemma, POS or chunk token. A corresponding Viterbi algorithm is then used to find the most likely sequence of S and NO-S states given the lemmatised words. Note that this model treats segment marks, <s>, as tokens.

3.2 Parse Tree Method

Parse trees generally contain nodes of clauses as illustrated in Figure 2. We assume that utterance boundaries only occur at major syntactic boundaries. This is similar in principle to the use of chunks as described in Section 3.1, where we hypothesise that a segment boundary exists before each token. The notion of a token, however, changes from representing chunks to sub-trees within a parse tree. Since a token in this context represents multiple words, and utterance segments may only occur in between tokens, this method significantly reduces the possibility of obtaining false-positive segment boundaries

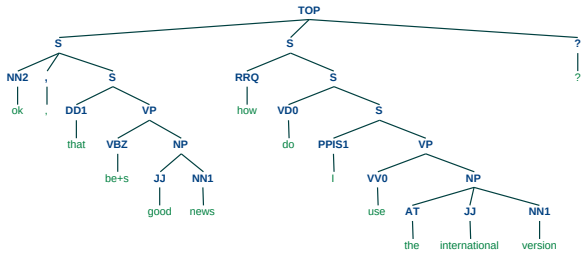


Figure 2: Parse tree of a message showing utterances separated into sub-trees as generated by RASP (Briscoe and Carroll, 2002).

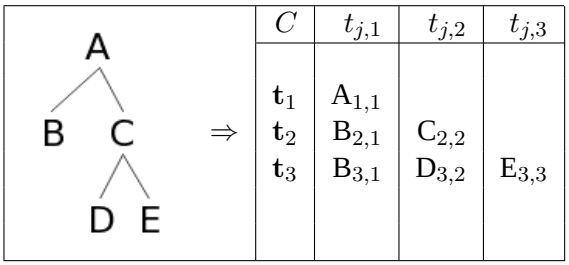


Figure 3: Proper analyses, C_1^3 from a parse tree.

when compared with using word or chunk tokens assuming correct parse trees. If the parse trees are not correct, however, this technique will have the opposite effect. This is discussed in more detail in Section 5.3.

To produce the parse trees, we use the RASP (Robust Accurate Statistical Parsing) parser described in Briscoe and Carroll (2002). RASP is designed to be domain-independent in order to handle text from different genres. Given that our data comes from instant messaging, which exhibits less predictable prose than that typically found in newspapers, we chose RASP over other parsers such as Collins (1999) and Charniak (2000) that are optimised on the Wall Street Journal treebank.

Utterance segments in our data always occur within a maximum depth of 2 nodes from the root of the parse tree. Using this depth limit, we first build a table of possible "cuts" through the tree. These cuts, or proper analyses as described in Chomsky (1965), contain every combination of sub-trees, as illustrated in Figure 3, resulting in a sequence C of nodes:

$$C = \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \ldots, \mathbf{t}_h \quad (1)$$

where each combination $\mathbf{t}_i$ is a sequence of tree nodes such that:

$$\mathbf{t}_i = t_1, t_2, t_3, \ldots, t_n \quad (2)$$

where the leaves of each tree node t_i represent a possible utterance.

We then calculate the most likely dialogue act for the leaves (words) within each node, independently in the combination table. The result and its corresponding dialogue-act are stored with the node t_i. Next, we calculate the probability of a correct sequence of utterances based on the product of the dialogue-act classification probabilities, using the following formulae:

$$\langle \mathbf{t}^*, \mathbf{d}^* \rangle = \arg\max_{\langle \mathbf{t} \in C, \mathbf{d} \rangle} \prod_{t_i \in \mathbf{t}} \hat{P}(d_i | t_i, d_{i-1})$$

$$\hat{P}(d_i | t_i, d_{i-1}) = P(d_i | d_{i-1}) \prod_{v \in \text{leaves}(t_i)} P(v | d_i)$$

where $\mathbf{t}^*$ is the best node combination (or segments), C is the set of proper analyses, $P(d_i|t_i)$ is the probability of node $t_i \in \mathbf{t}$ being dialogue-act d based on its leaves (words), d_{i-1} is the previously assigned dialogue-act (using bigrams), and v is a word in node t_i.

Using this method has the effect of evaluating the classification and segmentation tasks at the same time, taking the most probable combination. Algorithm 1 shows the process used to find the best proper analysis in C. The `classify` method returns the highest probability of all dialogue acts given the words in node n using the naive Bayes method. It also returns the corresponding dialogue act which is then stored with the respective node n.

Importantly, the naive Bayes algorithm uses a bag-of-words as its features, taking the product of each word's probability of being in any given dialogue act. This allows the product in line 6 of Algorithm 1 to be used as a ranking score amongst the proper analyses even though the number of nodes n in $\mathbf{t}$ may vary within C. If a different classification algorithm were used, then line 6 may have to be modified to preserve mathematical tractability.

Algorithm 1 Find best utterance segmentations $\mathbf{t}_i \in C$. The `classify` method also returns the best dialogue acts and probabilities which are stored with their nodes n.

```
1: max_p ← 0 {stores the best probability}
2: max_t ← None {best tree node}
3: for all t in C do
4:    p ← 1
5:    for all n in t do
6:       p ← p×classify(leaves(n))
7:    end for
8:    if p > max_p then
9:       max_p ← p
10:      max_t ← t
11:   end if
12: end for
```

4 Evaluation

The segmentation algorithms described in Section 3 were evaluated via 9-fold cross-validation where eight of the chat sessions in our corpus were used for training and one for testing. This process is repeated for all dialogues and the mean result is presented.

In this section, we first discuss why the standard information retrieval evaluation metrics of recall and precision are not appropriate for this type of segmentation, and then discuss the WindowDiff metric, which is used instead.

4.1 Using the Recall and Precision Metrics for Segmentation

The standard information retrieval metrics of recall and precision are not well-suited to evaluating segmentation tasks. Recall is the ratio of correctly hypothesised segment boundaries to the total number of actual boundaries. Precision is the ratio of correct boundaries detected to all hypothesised boundaries.

There are two main problems with using these metrics for segmentation tasks: the first is related to the inherently subjective nature of segmentation. An example is with the message "ok - that's great, thanks" in which "ok - that's great" could be segmented and tagged as a single ACKNOWLEDGEMENT or as the two utterances: "[ok]$^{\text{ACKNOWLEDGEMENT}}$ - [that's great]$^{\text{STATEMENT}}$". Deciding which segmentation should be considered correct depends largely on how the utterances will be used, that is, the downstream task. The traditional recall and precision metrics will regard the alternative segmentation as an error.

Similarly, if our corpus has a message that is manually segmented into two or more adjacent utterances with the same dialogue-act, the system should not necessarily be penalised for regarding the span of text as one segment. For example, "[Goodbye]$^{\text{CONVENTIONAL-CLOSING}}$ and [take care]$^{\text{CONVENTIONAL-CLOSING}}$" could just be marked as one utterance.

The second problem with using recall and precision to evaluate segmentation tasks is the question of how to handle near-boundary misses, that is, a false-positive that occurs near a true boundary. Using recall and precision in the way described will penalise a system equally regardless of whether a hypothesised segment boundary is off by one or ten words.

4.2 The WindowDiff Metric

The manually segmented data is used as a gold standard with which to compare hypothesised segmentations using the WindowDiff metric. The WindowDiff metric, proposed by Pevzner and Hearst (2002), aims to improve segmentation evaluation by rewarding near-misses.

WindowDiff works by choosing a window size k that is typically equal to half of the average segment length in a corpus. This k-sized window then slides over the hypothesised segmentation data and compares segment and non-segment marks with the reference data. If the number of hypothesised and reference segments within the window size differ, a counter is incremented and the window continues to the next position. The final score is then divided by the number of scans performed. A perfect system would therefore receive a zero score.

In most segmentation tasks, segment lengths are uniformly distributed, so using a fixed value for k is appropriate. However, because utterance lengths in our data vary considerably, as shown in Figure 4, we evaluate for different values of k. We adjust k from 1 to 20 for each message, taking the mean result for each value of k. The maximum allowable value of k is the message length on a per-message basis. This technique provides a fair evaluation given the varied utterance lengths.

Another question for our experiment is whether

allowing any deviation from our reference segmented data is acceptable, such as inserting a boundary somewhere near an actual boundary. Depending on where a boundary is inserted, this may result in two incomplete utterances as in example (1) below:

(1) a. Ref: [thanks] [you've been very helpful]
b. Hyp: [thanks you've] [been very helpful]

The segments in (1) differ only by one word, but the resulting utterances in (1-b) are confusing, especially when taken in isolation. In this case, we would not want to allow any deviation from the reference data. However, there are cases where a near-miss is acceptable, such as in (2):

(2) a. Ref: [Thank you for waiting], Customer. [I have found a page that lists a wide variety of Rock Climbing Shoes]
b. Hyp: [Thank you for waiting, Customer]. [I have found a page that lists a wide variety of Rock Climbing Shoes]

Here, the hypothesised segmentation (2-b) is just as acceptable as the reference (2-a).

The difference between examples (1) and (2) is that (2-b) has maintained the clause boundaries. The word *Customer* in (2-a) is not part of either segment, so including it in the utterance does not affect the adjacent utterance. Since an utterance is a complete phrase, this is the only way a near-miss may still be considered correct. Some other exceptions exist involving single-word utterances which will not be considered here.

5 Experimental Results and Discussion

The WindowDiff results for the various models and window sizes are shown in Figure 5 along with the baseline WindowDiff scores. A lower score indicates higher accuracy. The best result was achieved by the parse tree method. The worst result was given by the HMM POS tag model, but it still exceeded the baseline.

The relative difference between the models varies little as the window size changes. The WindowDiff score begins to taper off as k increases past 20 words, which is at approximately the 90th percentile

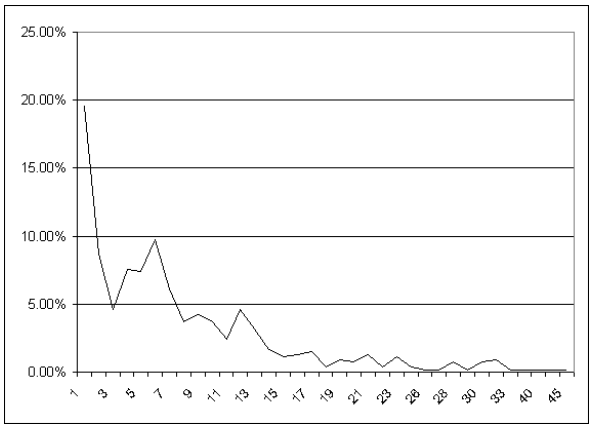


Figure 4: Frequency distribution of utterance length in words. The mean length is 7.6 words and the median is 6 words.

of utterance lengths in our corpus. This plateau is due to window lengths having no effect on shorter messages as a result of the adjustment we make to k when k is greater than the message length.

The better evaluation scores for small values of k are simply due to the way the WindowDiff algorithm compares segments within a window. An equal penalty is applied regardless of whether there are five or two segments within a window that should only contain one. Therefore, a window length spanning the entire message will at most return only one penalty if the hypothesised segments differ at all from the reference segments. Since the window spans the entire message, only one comparison is performed which results in the equivalent of a 100% error rate. Conversely, when k is small, the number of unequal windows between the reference and hypothesised segmentations will also be small since we have so few false positives. At the same time, the number of comparisons will be high, leading to a low WindowDiff score.

A perfect score of 0 is never achieved since there are always some misaligned segments. We never see a score of 1 since many of the single-utterance messages are accurately detected, as discussed in Section 5.1 below. Likewise, none of the models approach the baseline as the window size increases, which indicates that some of the multi-utterance messages are also accurately detected.

Although no individual value of k can be used to judge performance because of the varying segment

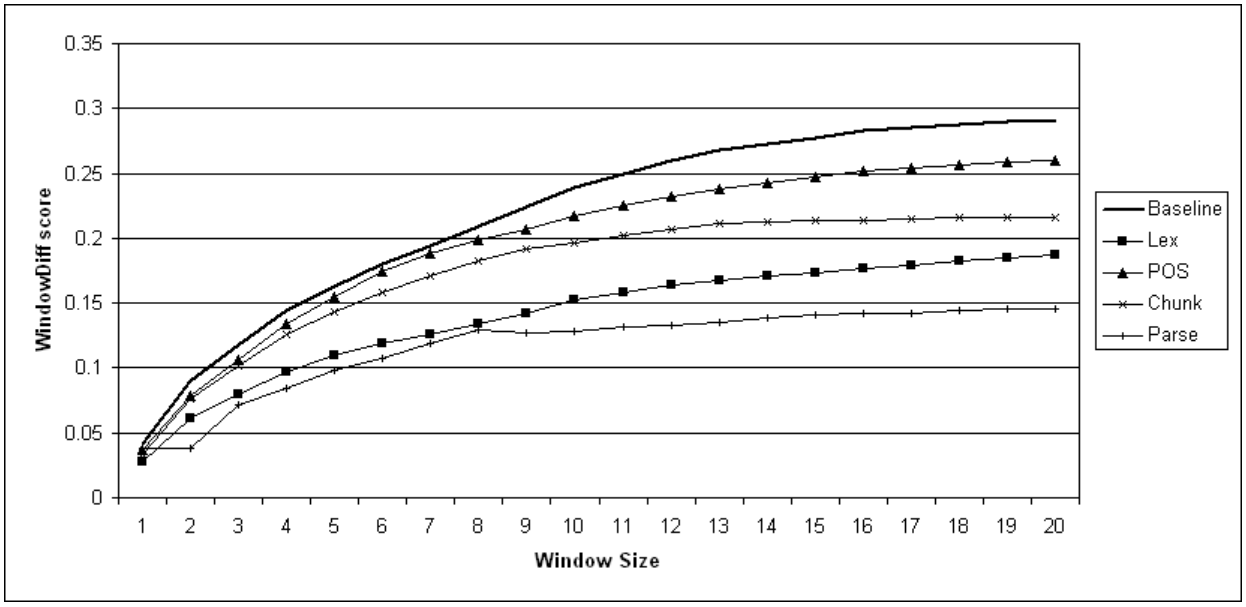


Figure 5: WindowDiff results of various models used and varying window size k from 1 to 20. A lower score indicates better accuracy.

(utterance) lengths in our corpus, we can confidently gauge the performance of each method relative to each other method since their respective rankings remain constant for all values of k.

5.1 Baseline

An analysis of our data revealed that messages contain up to three utterances. Of these messages, 60% contain only one utterance, 20% contain two utterances, and the remaining 20% consist of three utterances.

The baseline is calculated by assuming that each message contains only one utterance since this is the majority class.

5.2 HMM Results

We used three types of features with the HMM: lemmas, POS tags, and the head word of chunked data. The POS tag model performs the worst, whereas the lemma model is the best of the HMM models. This indicates that cue words play a major part in determining utterance segment boundaries. Replacing the words with their respective POS tags loses this information.

Using POS tags can sometimes help overcome data sparseness problems as it has the effect of generalising words. However, in this case it overgeneralises, resulting in poorer performance.

The rationale behind using chunks is that it reduces the number of possible boundaries as we hypothesise boundaries between chunks rather than words. Since utterance boundaries do not lie within chunks, this may have increased the probability of correct segment boundary detection. However, the results show that the HMM benefits from using all words rather than only the chunks' head words.

The main types of errors produced by the HMM are false positives based on words that commonly occur at the start of an utterance, such as "what", occurring mid-sentences as in (3):

(3) but I'm not sure what to get her

The reference data has this as one utterance, but the HMM detects a false positive starting at "what".

5.3 Parse Tree Results

The parse tree method gives the best results. A qualitative evaluation of the dialogue act classifications assigned to detected utterances gave an accuracy of 84%. The baseline for the dialogue act classification task was 36%, which was the majority class being STATEMENT.

The most common type of error the parse tree method makes is to separate words near the root of

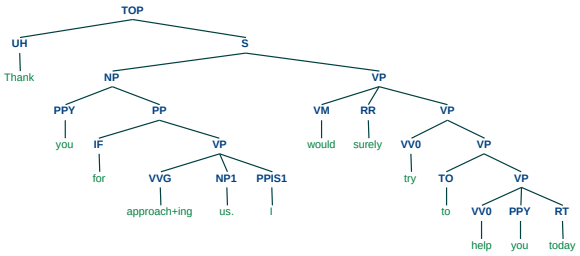


Figure 6: Erroneous parse tree of sentence (4) as produced by RASP.

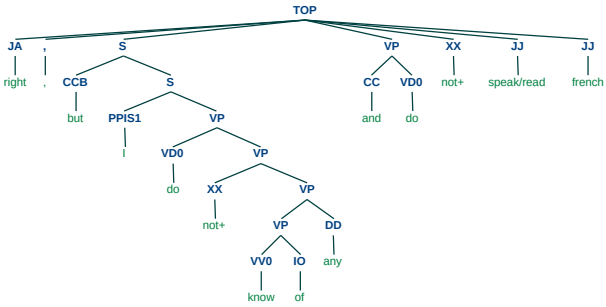


Figure 7: Parse tree of sentence (5) as produced by RASP.

a parse tree away from a deeper right node. Figure 6 shows a parse tree produced by RASP for (4) below:

(4) Thank you for approaching us. I would surely try to help you today

The parse tree for (4) is problematic. The first word, "thank", is detached from the S node that contains the rest of the sentence. Our model treats (4) as a sequence of word tokens $W = w_1, w_2, w_3, ..., w_{13}$ and finds that $P(\text{THANKING}|w_1) \times P(\text{STATEMENT}|W_2^{13}) > P(d|W)$, where d is any dialogue act. In this instance, RASP failed to segment the two sentences in this message, which prevented our model from evaluating the correct utterances. This illustrates the high dependency our model has on the quality of the generated parse trees.

Another type of error is that the model does not detect any segmentations within a message where there ought to be. An instance of this is in (5) below:

(5) right, but I do not know of any and do not speak/read french

The reference data has the word "right" segmented and tagged as RESPONSE-ACK and the rest of the message as one STATEMENT. However, our model does not evaluate that possibility as the corresponding parse tree in Figure 7 does not combine the words as would be expected.

6 Conclusion and Future Work

Finding utterance boundaries in IM dialogue is a critical step for aiding utterance classification and downstream language processing modules such as dialogue response planning. We have shown that the parse trees model obtains the best results. Of the HMM models, the HMM over lemmas in messages performs better than using chunked data and POS-tags, which lose too much information and impede accuracy.

The parse tree method performed best overall and has the advantage of combining both segmentation and classification tasks in one step to give the optimal combined result. It is based on the linguistic intuition that utterances are complete constituents, which are modelled well by parse trees. However, this heavy reliance on the quality of the parse trees is also a weakness. Most of the errors obtained using the parse tree method may be attributed to poor, or at least unexpected, parse trees being produced. That notwithstanding, the preliminary results using the RASP parser are very encouraging.

In future work, we intend to focus more on parsing IM messages, taking into account some of its distinct characteristics. Some obvious steps to produce better parse trees are to perform spelling corrections and expand acronyms, such as "idk" for "I don't know". Existing parsers will thus be able to produce more accurate parse trees, which will in turn result in higher segmentation accuracy.

We will also investigate the subjectiveness of utterance segmentation by performing Kappa (Siegel and Castellan, 1988) analysis on our segmentation

boundaries. The Kappa analysis will give an indication as to the meaningful upper bounds of the performance of our system.

Acknowledgments

Thanks to Steven Bird, Timothy Baldwin, and the Language Technology Group at Melbourne University for their constructive comments. Thanks also to Trevor Cohn, Phil Blunsom, and the anonymous reviewers for their very helpful feedback. The data used in this study was POS tagged, chunked, and parsed by Timothy Baldwin using the tools described with some modifications.

References

John L. Austin. 1962. *How to do Things with Words*. Clarendon Press, Oxford.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

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, Gran Canaria.

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

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia. Supervisor-Mitchell P. Marcus.

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

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

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Lori Levin, Klaus Ries, Ann Thyme-Gobbel, and Alon Lavie. 1999. Tagging of speech acts and dialogue games in Spanish call home. *Towards Standards and Tools for Discourse Tagging (Proceedings of the ACL Workshop at ACL'99)*, pages 42–47.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, pages 40–47.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

John R. Searle. 1979. *Expression and Meaning: Studies in the Theory of Speech Acts*. Cambridge University Press, Cambridge, UK.

Sidney Siegel and N. John Castellan, Jr. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill, second edition.

Andreas Stolcke and Elizabeth Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In *Proceedings, ICSLP 96. Fourth International Conference on Spoken Language*, volume 2, pages 1005–1008, Philadelphia, PA, Oct. ICSLP.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Hideki Tanaka and Akio Yokoo. 1999. An efficient statistical speech act type tagging system for speech translation systems. In *Proceedings of the 37th conference on Association for Computational Linguistics*, pages 381–388. Association for Computational Linguistics.

Author Index