Using Minimal Recursion Semantics in Japanese Question Answering

A thesis presented

by

Rebecca Dridan

to

The Department of Computer Science and Software Engineering

in partial fulfillment of the requirements

for the degree of

Master of Engineering Science

University of Melbourne

Melbourne, Australia

September 2006

©2006 - Rebecca Dridan

All rights reserved.

Thesis advisors
Timothy Baldwin and Steven Bird

Author
Rebecca Dridan

Using Minimal Recursion Semantics in Japanese Question Answering

Abstract

Question answering is a research field with the aim of providing answers to a user's question, phrased in natural language. In this thesis I explore some techniques used in question answering, working towards the twin goals of using deep linguistic knowledge robustly as well as using language-independent methods wherever possible. While the ultimate aim is cross-language question answering, in this research experiments are conducted over Japanese data, concentrating on factoid questions. The two main focus areas, identified as the two tasks most likely to benefit from linguistic knowledge, are question classification and answer extraction.

In question classification, I investigate the issues involved in the two common methods used for this task—pattern matching and machine learning. I find that even with a small amount of training data (2000 questions), machine learning achieves better classification accuracy than pattern matching with much less effort. The other issue I explore in question classification is the classification accuracy possible with named entity taxonomies of different sizes and shapes. Results demonstrate that, although the accuracy decreases as the taxonomy size increases, the ability to use soft decision making techniques as well as high accuracies achieved in certain classes make larger, hierarchical taxonomies a viable option.

For answer extraction, I use Robust Minimal Recursion Semantics (RMRS) as a sentence representation to determine similarity between questions and answers, and then use this similarity score, along with other information discovered during comparison, to score and rank answer candidates. Results were slightly disappointing, but close examination showed that 40% of errors were due to answer candidate extraction, and the scoring algorithm worked very well. Interestingly, despite the lower accuracy achieved during question classification, the larger named entity taxonomies allowed much better accuracy in answer extraction than the smaller taxonomies.

Declaration

This is to certify that:

i the thesis comprises only my original work towards the Masters except where indicated in the Preface,

ii due acknowledgement has been made in the text to all other material used,

iii the thesis is approximately 30,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Signed,

Rebecca Dridan
19th March 2007

Contents

Title Page i
Abstract iii
Declaration iv
Table of Contents v
List of Figures viii
List of Tables x
Citations to Previously Published Work xi
Acknowledgments xii
Conventions Used in this Thesis xiii

1 Introduction **1**
1.1 Information Access 1
1.2 Question Answering 3
1.3 Focus and Scope of this Thesis 4
1.4 Thesis Overview 6

2 Question Answering **9**
2.1 Introduction 9
2.2 Question Answering Architecture 12
2.3 Evaluation Methods 16
2.3.1 Correctness Judgements 17
2.3.2 System Evaluation Metrics 19
2.3.3 Modular Evaluation 21
2.4 Developments in Question Answering Tasks 21
2.5 Issues in Japanese Language Processing 26
2.6 Recent Results in Question Answering 28
2.7 Question Processing 30
2.8 Answer Extraction 33
2.9 Summary 36

3 Question Classification **40**
3.1 Named Entity Taxonomies . . . 41
3.2 Pattern Matching . . . 45
3.3 Machine Learning . . . 48
3.3.1 Features . . . 49
3.4 Results . . . 53
3.5 Conclusion . . . 59

4 Similarity **61**
4.1 Recognising Textual Entailment . . . 63
4.2 Word-based Metrics . . . 64
4.3 Syntax-based Metrics . . . 66
4.4 Semantics-based Metrics . . . 68
4.5 Summary . . . 71

5 Robust Minimal Recursion Semantics **72**
5.1 Introduction to RMRS . . . 73
5.2 Linguistic Processing Tools . . . 77
5.2.1 Japanese Tools . . . 78
5.3 RMRS Manipulation . . . 81
5.3.1 Merging . . . 81
5.4 Summary . . . 83

6 Answer Extraction using Similarity **84**
6.1 Comparison Algorithm . . . 85
6.1.1 Finding predicate matches . . . 87
6.1.2 Co-referentially indexed predicates . . . 89
6.1.3 Semantic relations through argument features . . . 90
6.1.4 Deriving the best match set . . . 93
6.1.5 Processing unmatched predicates . . . 94
6.1.6 Output . . . 94
6.2 Paraphrase detection . . . 96
6.2.1 Paraphrase Data . . . 98
6.2.2 Method . . . 99
6.2.3 Results . . . 100
6.3 Answer Sentence Selection . . . 102
6.3.1 Question Answering Data . . . 103
6.3.2 Method . . . 103
6.3.3 Results . . . 103
6.4 Answer Extraction . . . 108
6.4.1 Method . . . 108
6.4.2 Results . . . 109

6.4.3 Effect of different named entity taxonomies 110
6.5 Summary 111

7 Conclusion **113**

A Named Entity Taxonomies **131**
A.1 NE4 131
A.2 NE5 131
A.3 IREX 131
A.4 NE15 Taxonomy 132
A.5 NE28 Taxonomy 132
A.6 Sekine's Extended Named Entity Hierarchy 133
A.6.1 NUMEX branch 133
A.6.2 TIME TOP branch 134
A.6.3 NAME branch 135

List of Figures

2.1 Question answering architecture 13
2.2 Potential changes to the QA architecture for CLQA 16
2.3 Nuggets for definition question *What is a golden parachute?* 19
2.4 Developments in Question Answering Tasks 23
2.5 Examples of inconsistency in Japanese tokenisation 27
2.6 Alphabets used in Japanese text 28
2.7 Sekine's Named Entity Hierarchy, version 4 (abridged) 31

3.1 Soft decision making strategies using a hierarchical Named Entity taxonomy 43
3.2 Question types used for question classification 46
3.3 Examples of patterns used to identify WHEN questions 46
3.4 List of Japanese question words used in question classification 49

4.1 Examples of sentence patterns using named entity, phrase chunk and part-of-speech abstractions 67
4.2 An example of a Basic Elements (BE) sentence representation 68
4.3 Representation of `kiss` in glue semantics and MRS 70

5.1 RMRS elementary predicate representing "inauguration" 74
5.2 Full RMRS representation of "when was JR inaugurated?" 76
5.3 Underspecified RMRS representation of "when was JR inaugurated?" 77
5.4 Examples of the default output of the CaboCha dependency parser . 79
5.5 CaboCha RMRS representation of "when was JR inaugurated?" . . . 79
5.6 RMRS output from SProUT 80
5.7 Output of merging RMRS from JACY and from SProUT 82

6.1 RMRS structures for the example sentences 86
6.2 An example of content EPs 88
6.3 Match list after the predicate matching stage 89
6.4 Co-referentially indexed EPs of match 2 EPs 90
6.5 Indexed ARG2 EPs of match 2 91

6.6 Match list after predicate matching stage for examples (23) and (24) . 92
6.7 Match list after all argument processing is finished 93
6.8 Final match list 96
6.9 Final match result 97
6.10 Modified pih output 106
6.11 RMRS created from the pih output shown in Figure 6.10 107

List of Tables

3.1 Named Entity Taxonomy Summary 43
3.2 Question Classification Results 54
3.3 Lenient Question Classification Results for Hierarchical Taxonomies . 57

6.1 Default parameters for RMRS comparison 87
6.2 Results from paraphrase detection 101
6.3 Paraphrase results by class 101
6.4 Results from paraphrase detection, without ontology 102
6.5 Paraphrase results by class, without ontology 102
6.6 Results from the answer selection task 104
6.7 Question answering results using similarity 110
6.8 Results showing the effect of named entity taxonomies on question answering 111

Citations to Previously Published Work

Material from sections 6.1, 6.2 and 6.3 of this thesis has been previously published in:

Dridan, Rebecca, and Francis Bond. 2006. Sentence comparison using Robust Minimal Recursion Semantics and an ontology. In *Proceedings of the Workshop on Linguistic Distances*, pp 35–42, Sydney, Australia. Association of Computational Linguistics.

Acknowledgments

Almost 3 years ago I knocked on the door of Steven Bird, enquiring about a return to study. It was through his encouragement that my research began and Steven's ongoing advice has been much appreciated. Also highly appreciated is the support of my supervisor, Timothy Baldwin. Tim's suggestions, guidance and comments have helped me form my research ideas and, thus, create this thesis. Moreover, he has introduced me to a wide-reaching research community, both within and beyond Australia. I will be forever grateful for the valuable contacts he has made for me.

A number of other students and academics have served to guide and motivate my research through their comments, questions and ideas. I acknowledge particularly the contributions from Alistair Moffat, James Curran and the Melbourne University Language Technology group.

This research has benefited immensely from the chance to travel in Australia and Japan, which would not have been possible without funding from the University of Melbourne CSSE department, and NTT in Japan who funded an internship for me in their Communication Science Laboratories, Kyoto. The Pam Todd Scholarship, awarded by St Hilda's College, allowed me to purchase a laptop which greatly increased my productivity while in Japan.

The time in Japan really helped me solidify my research ideas and I am grateful for the assistance and friendship of everyone I met at NTT, especially Francis Bond who was instrumental in arranging the internship. I am also indebted to Francis for his ongoing help and advice which has made a huge difference to my thesis.

Throughout this research I have used tools developed by many people and I extend my gratitude to those who gave me access to their software, and to those who answered my cries for help when I couldn't make things work—Satoshi Sekine, Ulrich Schaefer, Kathrin Spreyer, Eric Nichols and Melanie Siegel. The resources available under the research agreement between The University of Melbourne and the NTT Communication Science Laboratories have also been of tremendous assistance.

I am very grateful for the support of friends and family throughout this whole process, with special thanks to Celia Colquhoun-King for proof-reading, to Fincina Hopgood and Christian Thorn for their support at crunch time, and to Martin Strauss for ideas, feedback, support, and for believing in me the whole way—thank you.

Conventions Used in this Thesis

Throughout this thesis Japanese will be presented in a Japanese script, with a phonetic realisation in italics and an English gloss in double quotes, thus: 運転 *unten* "drive". The phonetics will use the Hepburn romanisation style.

Abbreviations that will be used for glossing Japanese linguistic examples are listed below.

- ACC accusative
- DAT dative
- GEN genitive
- NML nominaliser
- NOM nominative
- QM question marker
- TITLE-HON honorific title
- TOP topic marker

Chapter 1

Introduction

1.1 Information Access

The amount of information publically available in electronic format has increased enormously in the past 50 years, which has brought about a change in information seeking strategies. Where once the focus was on finding as much information as possible, now it has become much more important to filter information, in order to pinpoint exactly the knowledge we are after. So how do we access the specific knowledge we need out of the vast collection of information we have? The field of information access covers a broad spectrum of techniques and applications that aim to answer this question.

The ideal information access application would be able to determine the information need of a user as expressed in their native language, search all information available in any language, and return the requested information, possibly compiled from multiple sources, in a language and format the user can easily understand. An application of this nature would need full language understanding, including implicature understanding, just to determine the information need. For example, if a user asks *Does an Australian tourist need a visa to visit Monaco?*, the ideal system would need to understand that *Yes* probably wouldn't satisfy the user's need, since they would then want to know what sort of visa and where it could be obtained. Once the information need is understood, still more inference and logical reasoning power

would be required in compiling the answer, since this could include resolving information such as *To visit Monaco, an Australian citizen needs the same visa they would require to visit France* and *Citizens from the following countries do not need a visa to travel as a tourist to France, provided their visit is less than 90 days: Angorra, Australia,...*. An ideal answer compilation system would also be able to draw the user's attention to, or resolve, conflicting information, and give some indication of the reliability of each information source.

The information access application described above is beyond the capabilities of current techniques. The other end of the information access application spectrum, within current capabilities, is the ubiquitous search engine, driven by techniques from the information retrieval (IR) field. In IR, the information need is interpreted as a bag of keywords without structure, and the answer is returned in the form of a list of documents whose topics are supposed to reflect the need expressed in those keywords. In the case of the visa information need mentioned above, a typical search request for an IR system might be {*Australian*, *tourist*, *Monaco*, *visa*}, but in a web search context this request may return many documents about visiting Australia. That is, we would expect the information about the direction of travel to be lost. To find the required information, a user would have to come up with the correct set of words to describe a page that is likely to contain the information, placing the burden on the user to know what sort of document that is, and perhaps be aware of documents that might appear relevant to the topics, but not contain the necessary information. This is even more difficult if the user is searching for information that is generally expressed in very common words, or words with more than one meaning. Once relevant documents are found, the user still needs to read or search the documents in order to find the appropriate information, which in the visa example, might be the information that they now need to start a new search for information on French visas.

The question answering (QA) field fits somewhere between IR and the ideal information access system, aiming to provide a more satisfactory response to a user's information need than in IR, but within the capabilities of current research (Maybury 2004). A QA system takes the information need of a user expressed in the form of a natural language question, allowing the user to specify exactly the information

they require, in the same natural way they might to a human. It then attempts to return an exact answer, using text from one or more documents. An example input that could be used for a QA system—a more explicit version of the earlier visa question—is: *What are the visa requirements for an Australian tourist visiting Monaco?* This clearly specifies both the information need—*visa requirements*—and the relevant constraints—*Australian tourist* and *visiting Monaco*. By employing a question as input, the information search can be more precisely directed without causing an extra burden on the user. The natural language phrasing provides information in the structure of the input, as well as the words, and also gives more information for disambiguation of polysemous words.

Cross-language question answering, sometimes known as cross-lingual question answering, takes the question answering a step further towards the ideal information access application. An cross-language question answering (CLQA) system would be able to take a question phrased in any natural language, search information sources in all languages, and return an answer in the original language used by the questioner (Magnini *et al.* 2003). To get to this level, issues from machine translation, such as semantic equivalence and transfer level, need to be addressed.

With the focus of this thesis being on addressing issues in question answering, the next section gives an overview of the field, including the issues that make it an interesting topic.

1.2 Question Answering

As will be explained in Chapter 2, a standard QA system is made up of three modules: question processing, passage retrieval, and answer extraction. The main contribution of the question processing module is to identify the question type (WHO, WHEN, WHERE,...), and the *Expected Answer Type* (EAT) which predicts the type of entity that question requires as an answer. Passage retrieval attempts to return answer bearing passages in a very similar way to an IR system, as described above, and is often implemented as such. Answer extraction uses the information provided by the other modules to pinpoint the answer within a passage.

The basic form of question answering takes a factoid question such as *Who is the King of Tonga?* and returns an answer in the form of a name or a number. These sort of answers are known as named entities and might be, for example, the name of a person, a country or a type of animal, or else perhaps a date, a time or an amount. More complex questions are also possible, and some examples of these are described in Chapter 2. While it is possible to use question answering to query information stored in a database, or some other form of structured knowledge source (eg, Frank *et al.* 2006), all the question answering systems discussed in this research use unstructured text as an information source.

Question answering is interesting from a natural language processing point of view, precisely because of the extra information available from using a question as input. While this extra information is immediately apparent to a human reader, if the computer system continues to treat the question as a bag of words, nothing is gained in using a question. If, on the other hand, natural language processing techniques, such as syntactic and semantic parsing and named entity extraction are used to analyse the question, it is possible to constrain answer sources to be, for example, those where Monaco is the entity that is to be visited.

1.3 Focus and Scope of this Thesis

This research explores the current state of the question answering field, with the ultimate aim of developing a full cross-language question answering system. As an initial step, this thesis focusses on techniques that would benefit a cross-language system, however these techniques are only evaluated as this stage using Japanese. Questions are restricted to factoid questions that can be answered independently of any other question.

As mentioned above, question answering systems need to use linguistically motivated methods to gain any benefit from using a question as input. A review of recent QA systems shows that all the better performing ones do indeed use techniques that involve high levels of linguistic processing. As will be seen in Chapter 2, however, Japanese systems lag behind those built for English and other European languages,

possibly because, in general, their answer extraction modules use a very low level of linguistic information. One of the goals of this thesis is to migrate the linguistically motivated techniques used in the better systems for use in Japanese question answering.

The problem with using these linguistically motivated techniques, when looking at the ultimate goal of cross-language question answering, is that many of them are language specific, and would require a substantial amount of manual effort to be reused with another language. Ideally, adding a new language to a cross-language system would require very little tweaking for each language, but at present many techniques depend on peculiarities of the language they are designed for such as, for English, the importance of the first words in a question. This motivates another goal of this research, which is to maintain language independence wherever possible. While there may appear to be a conflict between the twin goals of using linguistically motivated techniques and maintaining language independence, the solution to this clash of priorities is to separate the methods and the data as much as possible, developing methods that can use whatever linguistic data is available.

One factor to be considered in any linguistically motivated technique is the level of linguistic processing that will be used. Linguistic processing tools range from those that provide shallow linguistic knowledge such as word boundaries and parts of speech, to the very deep linguistic knowledge such as the way words relate together to produce a meaning. Many systems use only shallow knowledge because this is easy to produce, with a high level of accuracy. The argument against using deep processing is that because it is difficult, it can be brittle, often producing no information. This leads to a refinement of the goal of using linguistically motivated methods—to use deep linguistic information wherever possible, but in a robust fashion so that shallow information can be used where the deep information cannot be produced.

Robust Minimal Recursion Semantics (RMRS) is a semantic formalism that describes words and their senses, the relationships between words within a sentence (in a formal semantics sense), and the scoping of quantifiers within a sentence. RMRS helps towards the goal of language independence, by providing a level of abstraction above word order or syntax. While the RMRS representation of a sentence is still not

language independent, Copestake *et al.* (1995) showed that this abstraction greatly simplifies transfer in machine translation. The flat structure of RMRS, representing each piece of information about words or the links between them, as an atomic fact, not only makes computation easier, but contributes to the other main benefit of using RMRS, namely robustness. Hence, in this thesis, RMRS will be the method of using robust linguistic information in a language independent manner.

1.4 Thesis Overview

Chapter 2 provides an outline of the question answering field, including the standard 3-part architecture commonly used in QA systems, and some of the common evaluation methods used at recent QA evaluation forums. The evolutions in the QA task requirements at these forums are summarised, along with recent evaluation results. Since this research focusses on Japanese QA, a section describing Japanese text processing peculiarities is included, before describing current techniques used for question processing and answer extraction in QA systems for a variety of languages.

Chapter 3 explains how a named entity (NE) taxonomy is used in question answering and then reports experiments on question classification, using a variety of differently-sized NE taxonomies. Pattern matching, the most common approach to question classification, is evaluated by manually constructing rule sets to classify Japanese questions. This proves to be labour-intensive and more complicated than expected. I compare this to machine learning, a question classification technique used much less frequently, possibly due to a perceived lack of training data. In a set of experiments I evaluate the effectiveness of machine learning when a small (2000 question) training data set is available. Results from these experiments show that the machine learning technique, which is language independent, does significantly better than pattern matching, with much less manual effort required. The experiments also show that classification accuracy decreases as taxonomy size increases, but that some of the more common named entity types are more accurately identified when a larger taxonomy is used.

Chapter 4 looks at the concept of sentence similarity, and how it can be used

for detecting entailment and finding answers in QA. Sentence similarity is examined both from a word similarity perspective, and also according to how the words are related, referred to here as relational similarity. Methods for measuring relational similarity are described according to the sentence representation they use, and the complexity of the linguistic tool required to produce that representation. Examples are given of sentence similarities that can be decided from word-based, syntax-based and formal semantics-base representations, as well as demonstrating sentence pairs that are difficult to compare accurately using any of these representations.

As RMRS is the representation used for deciding similarity in later experiments, Chapter 5 describes the formalism in detail including what it looks like, what it means and how it can be underspecified. Some of the pre-existing language processing tools that produce RMRS output from different languages are summarised and then RMRS manipulation tools specifically produced for this research are described, including an algorithm for merging RMRS output from different sources.

Tying the previous two chapters together, Chapter 6 describes a novel algorithm for comparing text using RMRS as the representation format. This algorithm is then used in three different experiments. The first experiment evaluates the comparison algorithm directly, by using it to identify paraphrases in the format of definition sentences for the same sense from different dictionaries. The results of this show that RMRS provides a better format for comparison than bag-of-words for judging this form of similarity. The RMRS comparison algorithm is then used in a question answering framework to select sentences that are most similar to the question, in order to evaluate similarity as a method for finding answer bearing sentences. When named entity information is added to the RMRS comparison, this method does significantly better than a bag-of-words comparison in finding an answer bearing sentence. The last experiment uses the RMRS similarity score and the question classification results from Chapter 3 to extract answers for the questions in our data set. Results show that the answer ranking method works well, although there are issues to resolve with extracting possible answer candidates that will be ranked. In addition, these results show that the larger taxonomy allows greater accuracy in answer extraction, despite its lower accuracy at the question classification stage.

Chapter 7 summarises the findings of this research and reiterates the lessons learnt. Suggestions for further work and other interesting research questions are also given.

Chapter 2

Question Answering

2.1 Introduction

A question answering (QA) system allows a user to use the most natural form of information request—a question posed in natural language (Maybury 2004). Using a question as input allows the user to be quite specific about the information they require so a system can satisfy their information need. This differs from an information retrieval system that can only use an unstructured bag of keywords as an indication of what the user wants to know. A question answering system attempts to provide answers rather than documents, so the user is given just the information they need.

Research in question answering has examined many types of question (Lehnert 1981), and the following describes the types tested at recent evaluation forums. The simplest variety is generally known as the factoid question, requiring just a number or name as an answer. This question type can be extended to require a list of names, which can be more complicated since it might involve collating information from different sources and recognising and eliminating multiple representations of the same entity. Some examples of factoid questions with their answers are shown below. Question 3 is an example of a list question.

(1) Q. How high is Mt Fuji?

A. 3,776 metres

(2) Q. In which museum are Leonardo Da Vinci's writings about perpetual motion?

A. Victoria and Albert Museum

(3) Q. Who were the Australian Prime Ministers during the Vietnam War?

A. Sir Robert Menzies, Harold Holt, John Gorton, Sir William McMahon and Gough Whitlam

More complicated questions are those that ask for a reason, manner, method or definition, since the information need may be more difficult to extract, and they may require longer (but still coherent) answers. Some examples of these more open ended questions are:

(4) Q. What is a fractal?

A. A mathematically generated pattern that is reproducible at any magnification or reduction.

(5) Q. How do I remove decals from the car, without damaging the paintwork?

A. Start from one end of the graphic and move the blowdryer in a circular or a back and forth motion, covering about a 6 inch area. Carefully try to grab the corner of the graphic and start pulling it off the surface of the vehicle and at the same time moving the blowdryer ahead so the vinyl can be hot as you are pulling it off the vehicle.

All the above questions are stand alone questions that could be asked in isolation, but in some situations it may be more realistic to envisage a user asking a series of questions on the same topic. The question series below are examples of a user drilling down to the information they need (6), or else in a browsing mode, following up on previous answers of interest (7). Processing these question series can require a system to understand the context of the previous questions and answers, including resolving coreferences (Kato *et al.* 2004).

(6) i. What sort of visa does an Australian citizen need to visit Monaco?

ii. Where do I apply?

iii. What documents will I need?

(7) i. Who is Michael Jackson?

ii. What is his best selling hit?

iii. When was it released?

iv. Has it been covered by other artists?

Interactive QA also moves beyond the stand alone question model, eliciting further information about the information need from the user (Gonzalo *et al.* 2005). This might mean asking clarification questions - "Did you mean Melbourne, Florida, or Melbourne, Victoria?", or making suggestions for other potentially interesting questions.

Cross-language QA (CLQA) can encompass all the previous question types, with the further complication that the question and the document collection searched may be in different languages (Magnini *et al.* 2003). This ties back to the ultimate goal expressed in Chapter 1 of global information access.

Research into question answering has been enlivened by the introduction of shared tasks related to question answering at three international evaluation forums—TREC, CLEF and NTCIR. The remainder of this chapter gives an overview of the question answering field, as it has been developed through these forums; describes some peculiarities of Japanese text processing that impact on Japanese question answering; and then details recent results from the QA evaluations and the question processing and answer extraction techniques used to achieve these results.

First in the field overview, the generic 3-part QA pipeline architecture is explained, giving details of each of the three modules, and also describing the modifications that can be made to assemble a cross-language QA system. The discussion of QA evaluation gives details of the standard metrics used for different question types at each of the forums, as well as examining the case for modular evaluation. Finally, the developments in the question answering tasks are summarised, with respect to a

QA roadmap put together by the ARDA Q&A Roadmap Committee in 2001 (Burger *et al.* 2001).

Japanese natural language processing involves a number of issues that are not relevant to English or many other languages. Some of the complexities of tokenisation, multiple alphabets and multiple character encodings are described in Section 2.5.

In Section 2.6 recent results from evaluation forums are used to evaluate current progress in question answering research, and how that differs across the three forums. Specific techniques used in question processing and answer extraction, the two question answering modules that are explored in this thesis, are then examined.

2.2 Question Answering Architecture

A standard question answering architecture has emerged since the introduction of the question answering shared tasks mentioned above. The architecture, described by Paşca (2003) and used by many QA systems (Amaral *et al.* 2005, Isozaki 2004, Cui *et al.* 2004, *inter alia*), consists of three main modules—Question Processing, Passage Retrieval and Answer Extraction. The relationships and data flow between these modules are shown in Figure 2.1; we consider each module in turn below.

Question Processing

The question processing module extracts useful information from a given question. This can include keywords to be used in passage retrieval, the question type (who, what, when, how etc.) and the Expected Answer Type (EAT). The EAT will be used to filter or rank potential answer candidates, and comes from the Named Entity (NE) type taxonomy used by the system in question. The type could be fairly generic such as PERSON, NUMBER or ORGANISATION or as specific as BASEBALL TEAM. For the example *Who was the US president during the Gulf War?*, a system may detect that this is a *who* question and the EAT might be determined as either PERSON, PRESIDENT or US PRESIDENT, depending on the granularity of the NE taxonomy being used. The size and shape of NE taxonomies can have a significant impact on the question answering system. A simple taxonomy might just consist of a list of

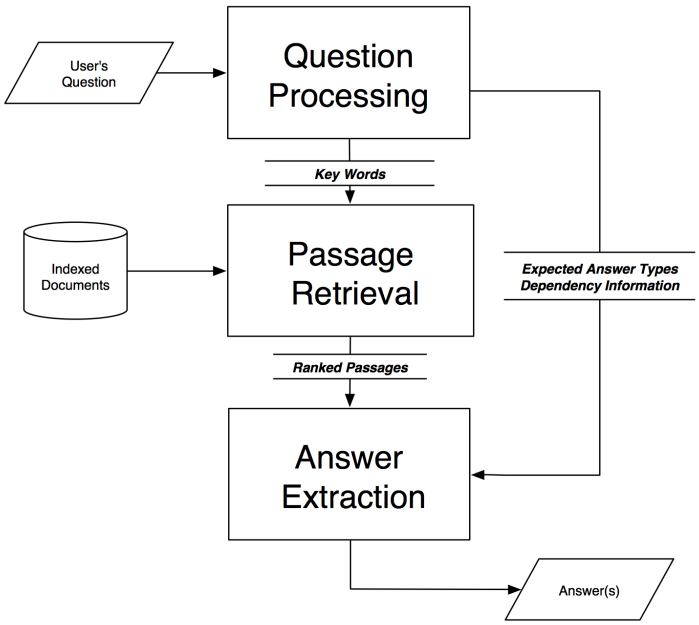


Figure 2.1: Question answering architecture

perhaps eight types, while large (100-200 type) hierarchical taxonomies are also used. Section 2.7 describes some of these taxonomies and how they are used, and the effects of the different sized taxonomies are evaluated later in this thesis.

While the question type can generally be determined by simple keyword matching, determining the EAT can require a deeper understanding of the question by the system, in order to narrow down the types of entities that would be possible answers. Often the question type can be used to help limit the possible EATs, but there is no one-to-one correspondence. For example, each of the questions below could have an EAT of COMPANY, despite having four different question types, while a *when* question, which often indicates an EAT of DATE could also be asking for an EVENT.

(8) Who is the world's largest producer of lithium-ion batteries?

(9) Where was Michael Eisner working before he became Disney CEO?

(10) Which company recently bought Skype?

(11) What is the name of the company that developed the computer animation artificial intelligence software, Massive?

The information about question type and EAT is predominantly used by the answer extraction module, but the question processing module also provides information for the passage retrieval stage, in the form of keywords. From our earlier example, the keywords extracted and used in passage retrieval might be *US*, *president* and *Gulf War*. Some of the methods currently being used in question processing are outlined in Section 2.7.

Passage Retrieval

The passage retrieval module takes information from the question processing stage and uses it to fetch a set of passages that may contain the answer to the given question from a set of indexed documents. The input may be limited to keywords, but some passage retrieval modules can make use of the EAT, or other syntactic information from the question. The passages returned might be full documents or shorter passages of text.

In many cases passage retrieval is implemented using a standard information retrieval engine. These are often based on tf.idf (term frequency–inverse document frequency) weightings which measure the importance of a word to a particular document. The tf.idf model works quite well in IR systems, since it highlights documents whose topics are reflected by the keywords. In QA systems, however, the answer to a question may be in a document without being the main focus of the document and hence many passage retrieval modules based purely on tf.idf perform poorly in QA.

While passage retrieval is obviously an important module in a QA system, it will not be examined in detail in this thesis. The focus of this research is on the use of linguistic knowledge to improve question answering accuracy, and passage retrieval has very little scope for using linguistic knowledge. Instead of implementing or sourcing a passage retrieval module, I will use what is termed 'oracle IR' for the evaluations. The 'oracle' is a list of document IDs for each question, where the answer is guaranteed to be in each document. For more information on passage retrieval methods that have worked for QA, see Isozaki (2004), Tiedemann (2005), and Soriano *et al.* (2005), among others.

Answer Extraction

The answer extraction module uses the information found in question processing to extract, score and rank potential answer candidates from within the returned passages. Answer candidates are generally identified by using named entity recognition to extract names of people and places, and numbers with their units. For the more open-ended questions, patterns might be used to extract, for example, definitions.

The answer candidates are then scored and ranked. The EAT determined in the question processing module can be used to eliminate candidates that are not of the right type, or to boost the rank of those that are. Beyond that, many systems use a hybrid scoring system that takes a variety of factors into account. Some use the score given to a passage in the passage retrieval module as part of the score for answers from that passage. The proximity of the potential answer to keywords from the question is another common scoring component.

Section 2.8 examines some of the more complex methods research groups have used to select the sentence from a passage that contains an answer, and then isolate the answer within that sentence.

Cross-Language Question Answering

Many cross-language question answering systems use the standard monolingual QA architecture shown in Figure 2.1, with machine translation (MT) inserted at one or more points. Some systems attempt to translate the question, others translate the entire document collection or just the keywords used for passage retrieval. While the document collection translation appears to involve much more work, the benefit for a fixed document collection is that the translation can be done off-line, minimising the translation required when questions are asked in real time. Translating the keywords requires the least translation of all, but does require that the question processing module and the answer extraction module operate on different languages. Translating the question is the most frequent method used, since this allows a pre-existing monolingual QA system to be used without change, by adding an MT engine to the front of the pipeline. While an ideal CLQA system would return the answer in the

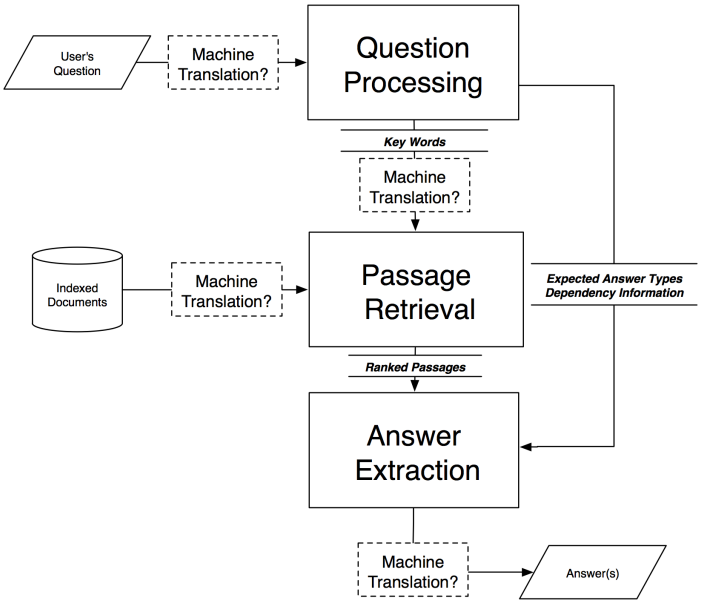


Figure 2.2: Potential changes to the QA architecture for CLQA

same language as the question was asked, most current CLQA systems return the answer in the language of the document collection. It is however possible to add machine translation in the final stage. Figure 2.2 shows the different places machine translation can be added to the standard QA architecture to form a CLQA system.

While there are QA systems that don't use the standard architecture, for example Clifton and Teahan (2004), the majority of systems at the three evaluation forums have used this basic framework. Standardising on an architecture enables more in-depth comparisons between systems and the techniques they use, and allows researchers that want to focus on one particular area to use off-the-shelf components for other modules, while still being able to evaluate the efficacy of their techniques on the overall system.

2.3 Evaluation Methods

Question answering evaluation is another area where the evaluation forums have helped to define standards that enable system comparisons. By defining key met-

rics, and in many cases, providing automatic evaluation tools, forum organisers have allowed researchers to compare their systems with past results to get a valid measure of the improvement (or otherwise) of new techniques. There are two elements to consider in QA system evaluation: how to judge the correctness or otherwise of any one answer, and how to combine these correctness judgements to give an overall performance score of the system.

2.3.1 Correctness Judgements

Judging the correctness of an answer to a factoid question appears straightforward in comparison to the corresponding task in information retrieval of judging relevance of a document with respect to keywords. The problems for factoid questions are usually concerned with the granularity and also the expression of the answer. Granularity is frequently a problem when the answer is either a date or a location. For example, *22nd November*, *November, 1956*, *November* and *1956* might all be possible answers to a question about an event date. The answers that would satisfy the information need of the user may depend on the type of event (*When is Australia Day?*), the duration of the event (*When was the Melbourne Olympics?* versus *When was the Opening ceremony of the Melbourne Olympics?*), how long ago it was, or even on the available information—is a less specific, but correct answer better than no answer? Equally, questions about locations might be more appropriately answered by a city, state or country, depending on the question. The evaluation forums have generally managed this issue by providing guidelines for their assessors, but there are often still disagreements over what constitutes a correct answer. The expression of an answer is a related problem and depends on the definition of an *exact answer*. At TREC-2002, for example, there was disagreement amongst assessors as to whether *Kidman* and *actress Nicole Kidman* were correct alternatives to the gold standard answer of *Nicole Kidman*. Assessment guidelines often spell out what is required or allowed in terms of units, articles and punctuation within an exact answer.

Another factor involved in judging the correctness of an answer is whether the answer is supported or not. In all of the forums, systems have to return the ID

of the document they extracted each answer from, and assessors have to check not only whether the answer is correct, but also whether the document supports this. Hence an answer of *Abraham Lincoln* for *Who was the 16th President of the United States of America?* may not be correct if the supporting document does not actually mention that he was a US President. Both TREC and CLEF have added `inexact` and `unsupported` to `correct` and `wrong`, as allowable judgements to cover some of these difficult to decide instances.

Once questions move beyond factoid to something that requires more than a named entity, date or amount as an answer, correctness judgements get more complicated. Definition questions are particularly difficult to judge since the scope of appropriate answer would depend on the user context—a student doing homework may require a different level of information to a scientist. CLEF avoided this problem by constraining definition questions to be about only people or organisations (eg, *Who is Kofi Annan?*) and allowing very general answers. TREC in 2003 began using a much more detailed judgement by first defining a typical user (Voorhees 2003):

> The questioner is an adult, a native speaker of English, and an "average" reader of US newspapers. In reading an article, the user has come across a term that they would like to find out more about. They may have some basic idea of what the term means either from the context of the article (for example, a bandicoot must be a type of animal) or basic background knowledge (Ulysses S. Grant was a US president). They are not experts in the domain of the target, and therefore are not seeking esoteric details (e.g., not a zoologist looking to distinguish the different species in genus Perameles).

The TREC assessors were then asked to research a particular definition question for themselves and create a list of "information nuggets" that might be appropriate parts of an answer, and then define which of those nuggets were vital. Nuggets should be defined in such a way that it is easy to make a binary judgement of the presence or absence of each nugget while judging an answer. An example of a definition question, and the nuggets selected, as described by Voorhees (2003), is shown in Figure 2.3.

1	vital	Agreement between companies and top executives
2	vital	Provides remuneration to executives who lose jobs
3	vital	Remuneration is usually very generous
4		Encourages execs not to resist takeover beneficial to shareholders
5		Incentive for execs to join companies
6		Arrangement for which IRS can impose excise tax

Figure 2.3: Nuggets for definition question *What is a golden parachute?*

2.3.2 System Evaluation Metrics

Once correctness judgements have been made on individual answers, an overall score can be calculated for each question and then the performance can be evaluated for each system over a question set. In evaluations that allow `inexact` and `unsupported`, as well as `correct` and `wrong`, two separate performance measures can be given. These have become known as "strict" and "lenient" evaluations, where the overall score from a strict evaluation counts only `correct` answers, but the score from a lenient evaluation also considers `unsupported` and `inexact` to be correct. Hence a lenient evaluation score will be higher, allowing answers that may be inexact or unsupported but are considered "good enough".

The evaluation metrics used for question answering systems differ according to whether systems are to return one answer or multiple answers. When only one answer is returned per question, the standard metric is accuracy—that is, percentage of questions in a test set that are answered correctly. Some variations on this, used at both TREC and CLEF, incorporate a confidence measure into the metric. The confidence-weighted score requires systems to return the answers to a question set in ranked order of confidence, so their most confident answer, regardless of which question it is for, is returned first. The total score then, for a set of Q questions is

$$\frac{1}{Q}\sum_{i=1}^{Q}\frac{\text{number correct in first } i \text{ ranks}}{i}$$

This gives more weight to correct answers ranked at the top than correct answers lower down. Other metrics that incorporate confidence scores are detailed in Herrera *et al.* (2004).

Tasks that require more than one answer are more complicated to evaluate. In the case of question answering systems that return a ranked list of answers for each question, Reciprocal Rank is used. Reciprocal Rank (RR) is the reciprocal of the rank of the first correct answer. Hence if the first answer is correct the RR is $\frac{1}{1}$, while if the fifth answer is the first correct answer returned, then the score is $\frac{1}{5}$ or 0.2. The system evaluation for a specific data set is then the Mean Reciprocal Rank (MRR), the mean of the RR score for each question in the set. The MRR is a somewhat lenient evaluation metric because it gives credit for returning the correct answer below incorrect answers. This kind of evaluation makes sense for information retrieval, where the judgement is on relevance, rather than correctness, because it is not unreasonable to expect a user to scan through a few documents looking for a relevant one. In this case, the user is able to judge relevance for themselves. In question answering, on the other hand, answers could be right or wrong and it is not reasonable to ask a user to decide which answer in a list is correct. While a user may be able to discard obviously false answers, it is quite likely a system could return more than one probable answer for a question like *How long is the border between France and Belgium?*. If a user already knew the answer, they probably wouldn't be looking for it!

Tasks that require a complete list of correct answers generally use the concepts of precision, recall and F-measure from information retrieval. These metrics are defined as follows: if C is the number of correct answers returned, N is the total number of answers returned, and T is the total number of correct answers that should have been returned, then

$$\begin{aligned} \texttt{precision} &= \frac{C}{N} \\ \texttt{recall} &= \frac{C}{T} \\ \texttt{F}_\beta\texttt{−measure} &= \frac{(1+\beta^2) \times \texttt{precision} \times \texttt{recall}}{(\beta^2 \times \texttt{precision}) + \texttt{recall}} \end{aligned}$$

When β is set to the default value of 1, the F-measure is the harmonic mean of precision and recall.

The nugget based method of evaluating definition questions uses a variant of

F-measure, where recall is calculated only for vital nuggets and precision is approximated by equating "answers returned" with a function of the response length, since a discrete value of answers returned is not possible. This gives more weight to shorter responses that contained the same information. Full details of the calculation are explained in the TREC 2003 QA overview (Voorhees 2003). The β value for the definition F-measure at TREC 2003 was 5, indicating that recall was considered 5 times more important than precision.

2.3.3 Modular Evaluation

The evaluation forums have had a huge impact on the field by allowing systems to be compared on the same tasks over the same data. However, given the pipeline architecture of most QA systems, an improvement often suggested is the idea of modular evaluation. Most systems follow the standard 3-part QA architecture outlined in Section 2.2 but tend to focus their efforts on one particular module. Without an evaluation of the effectiveness of each module, it is difficult to compare the effect of different techniques for individual modules. Many participants do their own modular evaluations and publish their results in their system description but so far the forums only provide end-to-end evaluations. This may be slowly changing—in the TREC 2005 QA Task systems were required to return a ranked document list along with every question answer, so that the passage retrieval effectiveness could be measured, but results from this are not yet publically available.

2.4 Developments in Question Answering Tasks

Question answering tracks at TREC, CLEF and NTCIR have provided researchers a way to evaluate their QA systems, with the tasks gradually becoming more difficult with each succeeding year. TREC, the Text REtrieval Conference held annually in the USA, was the first to introduce a question answering track in 1999 (TREC-8) (Voorhees 1999). The first non-English QA task was Question Answering Challenge (QAC1), held at NTCIR-3 in 2002, which evaluated Japanese monolingual QA (Fuku-

moto *et al.* 2002). The NTCIR (NII Test Collection for IR) Workshops are held every 18 months in Japan and focus on Japanese and other Asian languages. CLEF, the Cross Language Evaluation Forum, focusses on cross language issues and hence the first QA initiative at CLEF was the Multiple Language Question Answering Track at CLEF 2003 (Magnini *et al.* 2003). This track had monolingual and cross-language tasks involving English, Spanish, Dutch, German, Italian and French. While TREC QA tasks have remained focussed on English monolingual systems, NTCIR added a Cross-Lingual Question Answering task in 2005, using combinations of Japanese, Chinese and English.

In 2001, the ARDA Q&A Roadmap Committee produced a roadmap to guide research in question answering, defining milestones in areas such as question classification and analysis, answer extraction and formulation, reasoning and inference, multilingual QA and user modelling (Burger *et al.* 2001). While many of the milestones set have proven to be much too ambitious for the current level of progress, successive forums have gradually implemented some of these milestones as task requirements, meaning the question answering task that participants have been asked to attempt has become progressively harder. This increase in difficulty has been due to changes in question type, in the ranges of answers required, and in the accepted format of the answer. A summary of these developments is shown in Figure 2.4.

The initial QA task at TREC was designed to be a simplified version of question answering, to initiate interest in the QA task. Participants were expected to return up to five ranked 50 or 250 byte text snippets. The questions were factoid questions, hand written by the organisers who looked through the document collection of newspaper texts to find entities they could ask about. The answers were guaranteed to be in the document collection. Since up to 5 answers could be returned, evaluation was by mean reciprocal rank (MRR, see Section 2.3). The first increase in difficulty came the following year when the questions were assembled from real query logs, which meant there was much less word overlap between the questions and the phrasing used in the document collection.

In 2001, continuing the move towards a more realistic task, the TREC task no longer guaranteed that the answer existed in the document collection, and participants

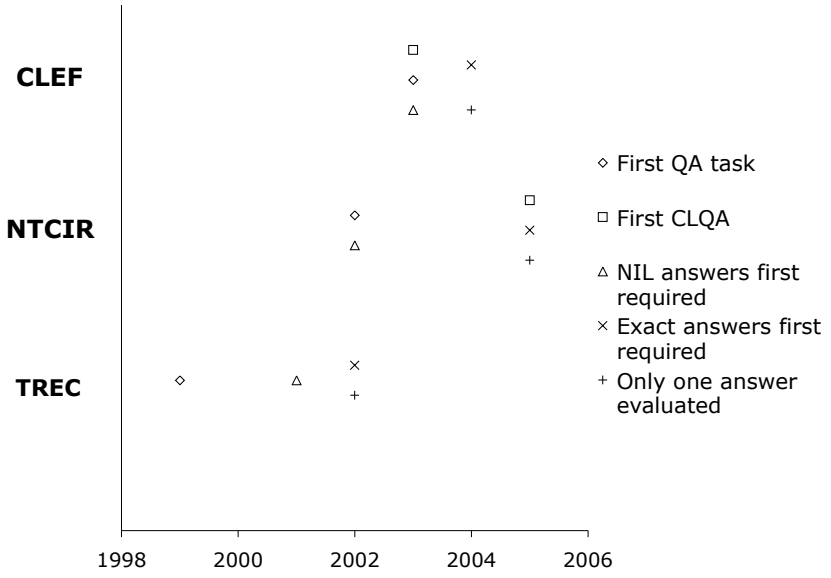


Figure 2.4: Developments in Question Answering Tasks

were required to return NIL if a correct answer could not be found. This requirement reflects the belief that no answer is better than an incorrect answer. Recognising the non-existence of an answer required more intelligent processing, since rather than just returning the most likely answer it found, a system needed to determine the confidence in that answer, and return NIL if the confidence was not high.

More complex question types also appeared in 2001. The first change was an unexpected increase in definition questions, due to the use of different query logs as question sources. While definition questions were on the QA roadmap as a future target, at this stage they were considered too difficult, and culled for the next year. A more deliberate extension was the addition of a list task, which required collating answers from multiple documents. Participants were told the number of answers required for each question, and the evaluation for this task was accuracy—the number of *distinct*, correct answers returned out of the number that was required. Recognising alternate expressions of the same answer was meant to be the complicating factor in this task, since only distinct answers would be counted, but the assessors found that incorrect answers were much more of a problem than duplicates. This may, however,

have been due to a lack of duplicates in the document collection. A pilot context task was also run, asking a series of questions where later questions depended on earlier questions and answers. At this stage, understanding context was not found to be the limiting factor for this task, since system performance depended more on the type of question than on the context needed.

NTCIR's QAC1 in 2002 was very similar to the TREC 2001 QA track, requiring 5 ranked answers to factoid questions, as well as having a list and a context task. However, this challenge required exact answers, rather than 50 byte strings. This was moving closer to the ideal QA system, isolating the answer, rather than an extent of text likely to contain the answer, which is an extra level of complexity for the participating systems. This initiative also made evaluation more complicated since it was not always clear what an exact answer should include, or how specific it should be. TREC-2002 also required exact answers in the QA task, and introduced a new judgement of `not exact` for answers that included too much, or not enough information.

The other change made in TREC-2002 was only accepting the first answer returned, rather than a ranked list of 5. This necessitated a change in scoring mechanism (away from MRR) and a new metric called confidence-weighted score was used (described in Section 2.3). This allowed systems to report how confident they were on any answer, which is important information for users of a QA system, who need to know how much they are able to trust answers they receive.

The first CLEF QA task was held in 2003. The monolingual tasks (in Dutch, Italian and Spanish) were equivalent to the TREC 2002 main task, except that up to 3 ranked answers were allowed instead of just one, and so MRR was used for the evaluation. It was also possible to submit a 50 byte answer instead of an exact string, but most participants elected to submit exact answers. The real complexity added by CLEF 2003 was the inclusion of the first cross-language tasks. These consisted of questions in Italian, Spanish, Dutch, French and German, with a document collection in English. Answers were returned in English.

The following year, CLEF not only moved to one exact answer, with confidence score, but also broadened the type of question asked. Question sets included definition

questions and *how* questions, in addition to the factoid questions that had been evaluated in the first event. The number of languages also increased, with question sets constructed for 9 languages and document collections for 7, so that 6 monolingual and 50 cross-language tasks were possible (although only 13 of the cross-language tasks were actually attempted).

TREC added definition questions to their test sets in 2003, which they evaluated using the concept of information nuggets that was detailed in Section 2.3. The list questions for that year changed slightly so that the number of answers to be returned was not specified. Both list and definition questions were evaluated by F-measure (see Section 2.3) and a combined overall score for the task was calculated by a weighted combination of factoid accuracy and F-measure of list and definition questions.

In 2004, TREC used the same sorts of questions and evaluations as the previous year, but divided the questions into 65 series of questions, with each series focussed around one topic. Each series had multiple factoid questions, up to two list questions and an OTHER question which was meant to be similar to a definition question. Answering each question in a series could depend on understanding context and coreferences from earlier questions, but not answers. Evaluation was the same weighted combination as the previous year.

In 2005, NTCIR introduced the Cross-Lingual Question Answering task (CLQA1) consisting of cross-language QA using Japanese, Chinese and English. The monolingual task they ran that year (QAC3) included many of the more advanced requirements introduced at TREC, such as list questions and questions which required understanding of context, but the CLQA task went back to simpler requirements. Five ranked answers could be returned and all answers were exact named entities. Evaluation was both by accuracy (looking at only the first answer returned) and MRR.

While not getting as far as the logical inference and answer formulation goals described in the roadmap, the trend over the past eight years has been moving towards a realistic question answering scenario, with real questions requiring either one answer, or notification of a lack of answer. Looking ahead, the extensions planned for QA tasks at all forums are primarily focussed on harder questions. This includes developments

like time-restricting questions (*Who was Uganda's President during Rwanda's war?*), asking about relationships between entities (*Who were the participants in this spy ring, and how are they related to each other?*) and *why* questions (*Why is the sky blue?*). In addition to the standard question answering tasks, other tasks related to QA have been introduced. Both TREC and CLEF have organised interactive experiments that aim to investigate aspects of user interaction and the benefits it can bring to QA. All three forums encourage research into evaluation of QA systems and, in 2006, an Answer Validation Exercise at CLEF was announced to provide a structured task for researchers to attempt automatic validation of QA responses in a very similar way to that used in the PASCAL Recognizing Textual Entailment Challenges (Dagan *et al.* 2005, Bar-Haim *et al.* 2006). The harder questions and the automatic evaluations both suggest the need for a deeper understanding of the questions and documents as they move beyond the capabilities of pattern matching, to a natural language processing level that requires recognising constraints, relationships and semantic equivalence.

2.5 Issues in Japanese Language Processing

Japanese has particular issues that complicate language processing. These issues often mean that techniques developed for other languages cannot be directly applied without a certain amount of pre-processing. The first issue is that, like Chinese and Thai, Japanese is a non-segmenting language. This requires that a tokenisation process be used to split text into 'words'. Various tokenisation tools exist for Japanese, but there is no definition for the ideal lexical unit, and so the segmentation unit or token varies between tools. Some Japanese processing systems will split text into *bunsetsu*, which are logically larger than English words, similar to phrase chunks, while most of the standard tokenisation tools create smaller units, generally splitting input into morphemes. Even within the morpheme-based tools, tokenisation is inconsistent and it is important to use the same tokenisation tool for a full system to avoid the systematic differences that occur when using different tokenisers. Figure 2.5 gives examples of tokenisation into *bunsetsu*, and two variations of morphemes (from

Original sentence:	彼は意図的に窓を割った "he broke the window intentionally"
bunsetsu:	彼␣は␣意図的に␣窓を␣割った
ChaSen:	彼␣は␣意図␣的␣に␣窓␣を␣割っ␣た
Juman:	彼␣は␣意図␣的に␣窓␣を␣割った

Figure 2.5: Examples of the ways a Japanese sentence can be tokenised into *bunsetsu* and two variations of morpheme, where "␣" indicates a token boundary

ChaSen and Juman).

Other issues arise from the characters used to write Japanese. Japanese can be written using 4 different alphabets—kanji, hiragana, katakana and latin. Kanji are Chinese characters which can have multiple pronunciations and are logographic in nature. That is, each character can represent a complete word or morpheme. There are thousands of kanji characters, though many are considered ancient or obscure. The Jōyō kanji are a set of 1,945 characters that all students in Japan are meant to know before finishing high school. About 1000 more are regularly used in people's names. Kanji characters outside the Jōyō set often have their phonetic reading written above them in publications. Hiragana and katakana are both phonetic alphabets with 46 basic characters each. Hiragana is used for inflected verb endings, particles and other words where the kanji is either not known, or too formal. Katakana, on the other hand, is generally only used for foreign loanwords, or sometimes (eg, in advertising) for emphasis. The latin alphabet is often used in Japanese, particularly for acronyms. While there are kanji representations for numbers, the Arabic numbers are also often used. Figure 2.6 shows the characteristics of each alphabet, and also a Japanese sentence that uses all four, as well as Arabic numbers.

Because of the huge number of characters that can be used, Japanese cannot be written using the ASCII character set. There are at least 3 different encoding systems in common use, all incompatible. Most programming languages used for text processing (such as Perl, Python, etc) now use UTF-8 encoding by default, since UTF-8 is capable of encoding the majority of characters used world wide. However, EUC-JP is the de facto standard in Unix-based computing in Japan, and so many

Alphabet	Characters	Common Usage	Examples
Kanji	1,945	native Japanese words	漢字 時
Hiragana	46	verb endings, particles	から あ
Katakana	46	foreign words	スーパー ル
Latin	52	acronyms	ARC MoD

(a) Characteristics of each alphabet

２０００年のＮＨＫ大河ドラマは何ですか
"What was NHK's large scale period drama in 2000?"

２０００	年	の	ＮＨＫ	大河	ドラマ	は	何	ですか
arabic	kanji	hiragana	latin	kanji	katakana	hiragana	kanji	hiragana

(b) An example sentence that uses all four alphabets and Arabic numbers

Figure 2.6: Alphabets used in Japanese text

Japanese processing tools and resources use EUC-JP. Conversion is possible between encodings, using `iconv` on the command line or the Encode module[1] in Perl, but this adds an extra layer of complexity to a processing system and requires advance knowledge of the encoding of any particular resource used.

All of these issues mean that Japanese language processing involves extra layers of complexity and overhead when compared to English and other European languages, even for simple pattern matching techniques. This may be a contributing factor to the lower level of performance of Japanese QA systems when compared to those at TREC and CLEF. These results are outlined in the next section.

2.6 Recent Results in Question Answering

The current state of question answering research is best demonstrated by the systems and results described in the recent proceedings of each of the evaluation forums outlined in Section 2.4. Looking at the results from TREC (Voorhees 2004)

[1] `http://search.cpan.org/~dankogai/Encode-2.18/Encode.pm`

and CLEF (Vallin *et al.* 2005) we can see that the accuracy over factoid questions is comparable for monolingual systems in English and other European languages. While the accuracy of the top TREC system (77%) is slightly higher than than CLEF's best system, the results from TREC and CLEF are similar. Both forums had a small number of systems with accuracies in the mid-60's and other competitive systems had scores between 30 and 50%. In contrast, the Asian language systems participating in NTCIR tasks had significantly lower performance. In the Japanese monolingual task at QAC2 (Fukumoto *et al.* 2004), the top system had an accuracy of just over 50%, four other systems were above 40% and the majority of systems scored between 20 and 40%. There could be some argument given to support the view that the Japanese tasks were harder. The questions tend to be longer and more complicated in expression than those used at either TREC or CLEF, but the answers required were still basic common and proper nouns, and number and time expressions, and they didn't require any inference or real world knowledge.

Cross-lingual question answering showed a performance drop of between 10 and 20% when compared to the equivalent monolingual task at both CLEF (Vallin *et al.* 2005) and NTCIR (Sasaki *et al.* 2005). The vast majority of cross-lingual QA systems were constructed by taking an existing monolingual system and adding machine translation (MT) somewhere in the pipeline, as described in Section 2.2. Generally the question was translated by an off-the-shelf MT system but, given the quality of these MT systems, the resulting question was often ungrammatical, affecting the accuracy of any parsing, and, if the wrong sense of the word is translated, even keyword processing is affected. A few CLQA participants made an effort to construct systems that used full question analysis modules for the question language. One in particular, Synapse Développement's Qristal system (Laurent 2005), was easily the best CLQA system at CLEF achieving accuracies of 39.5% for English-French and 36.5% for Portuguese-French.

Passage retrieval was a limiting factor for many of the QA systems, with the answer often not being returned in any of the retrieved passages. However, research teams that focussed their efforts on the passage retrieval stage demonstrated that it was possible to get highly effective results by using QA specific techniques. The NTT

Communications Science Lab group showed this in NTCIR's QAC2 where they only once failed to return an answer bearing document (Isozaki 2004). In that task, they had the highest scoring system, *however* they still only had an accuracy of 50% and their evaluation showed that accurate question processing is essential for accurate question answering. For this reason, and because oracle IR data is available in the form of lists of documents guaranteed to contain the answer, this thesis focusses on the two modules where the scope for improvement is largest, and which would appear to benefit most from linguistic knowledge. Current methods used in question processing and answer extraction modules are reviewed below.

2.7 Question Processing

The primary goal of question processing is to determine one or more expected answer types (EAT) which Hovy *et al.* (2001) showed was vital for precisely and efficiently pinpointing the location of an answer within a passage. Reviewing the techniques used for question processing in recent evaluations, the defining factors of most systems are the size and shape of the named entity taxonomy used, and whether the system used a rule-based or machine learning technique to classify the EAT within this taxonomy. The discussion below describes some of the taxonomies that have been used, and reviews some of the results achieved using the different classification methods.

The EAT is taken from a named entity (NE) taxonomy such as the IREX taxonomy (Sekine and Isahara 1999). This is commonly used by systems in NTCIR tasks and consists of 8 types: ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY and PERCENT. Similar flat generic taxonomies were also used by systems at TREC and CLEF, however at all three forums the better performing systems tended to use much larger, richer taxonomies. Several groups (eg, Bouma *et al.* 2005, Nii *et al.* 2004) enriched their NE type sets by extracting apposition relations such as *John Howard, Prime-minister of Australia* from corpora, as described by Pantel and Ravichandran (2004). This method has the benefit of collecting named entities as well as their types, but produces a flat taxonomy. Hierarchical topologies allow

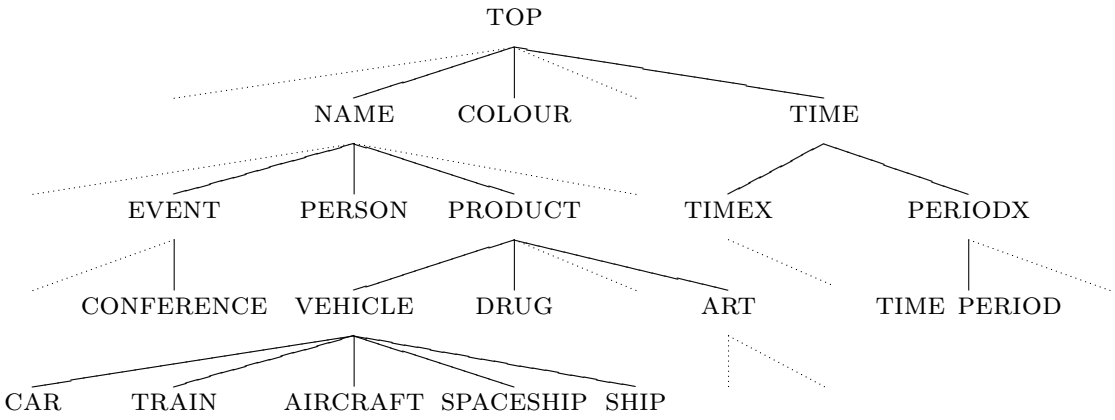


Figure 2.7: Sekine's Named Entity Hierarchy, version 4 (abridged)

more flexibility in matching since a question tagged as requiring a VEHICLE could be answered by a named entity tagged as TRAIN. Sekine's Extended Named Entity Hierarchy[2] (Sekine *et al.* 2002a) is one of the larger hierarchical taxonomies used by multiple systems at NTCIR; Figure 2.7 shows a section of this taxonomy.

The proponents of the small (8-12 type), generic class set claim that including more categories leads to less accurate classification and hence to a less accurate overall system (Kurata *et al.* 2004). The truth of even the first part of that claim is not evident, as shown by research groups like Laurent *et al.* (2005), who achieved EAT accuracy of over 90% for a NE topology of 86 types. It is also apparent from system results that the accuracy gained in using a small NE type set is lost in the lack of discriminative ability to select between, for example, CITY and COUNTRY. Ambiguity of answer types is another issue that, on the surface, might argue for generic types. For example, in *Who recently appointed Max Borg as their new director?* the answer could potentially be a HOSPITAL or a COMPANY, and perhaps ORGANISATION would be a better choice. Two methods have been suggested for dealing with this problem without shrinking the type set—assigning multiple types in case of ambiguity or using a hierarchical topology and assigning the most specific type that covers all possible options.

The methods used for finding these EATs can generally be split into rule-based

[2] `http://nlp.cs.nyu.edu/ene/`

pattern matching or machine learning techniques. Li and Roth's (2006) work on question classification using their SNoW learning architecture, trained on 21,500 questions, was able to correctly classify the required answer type for 1000 questions with 89.3% accuracy, using a hierarchical type taxonomy of 50 types. Despite this success, very few recent QA systems use machine learning in their question classification, with Tomás *et al.* (2005) who achieved an accuracy of 77% at CLEF being one exception. Lack of question training data annotated with their answer type may be one reason machine learning techniques have not been used extensively. Data sparsity for rare types would also be a problem. Given the potentially low volume of training data, selecting relevant and informative features becomes very important. One team that used an SVM for question classification between 12 NE types concluded that given the small training corpora available, features that included linguistic knowledge were very important (Kimura *et al.* 2005). This echoes the findings of Li and Roth (2006) that showed that using syntactic features such as part of speech and phrase chunks gained significant improvement over just word features for coarse grained classification, while adding semantic features made a big difference when classifying into fine grained classes.

The rule-based pattern matching methods can range from simple handcrafted regular expressions to complex rules and templates. Regular expressions based on words work reasonably well for coarse grained NE topologies (eg, *who* → PERSON) but tend to have low coverage of more complex questions, are time consuming to construct and very specific to the language and question type they are written for. Patterns and rules based on more abstract features such as phrase chunks like NP (noun phrase) and named entity tags can achieve a higher coverage with less rules.

Apart from question classification, question processing also involves finding search terms for the passage retrieval stage, which can be as simple as taking all content words from the question, but may also involve adding terms specific to the EAT found, such as adding birthplace if the EAT is LOCATION (Sakai *et al.* 2004). Question processing can also extract constraints, relations and dependencies in those systems that go beyond the standard answer extraction method.

Some form of question classification was performed by almost every question an-

swering system surveyed. While many different NE taxonomies were seen, all the better performing systems used larger taxonomies (greater than 40 types). Machine learning was shown to perform very well for classification across a large taxonomy, but was not used by many systems, possibly because of a lack of training data. Most systems that did use machine learning reported that features that encoded linguistic knowledge had a positive impact on classification accuracy. Methods for question classification when only small amounts of labelled question data are available will be explored in Chapter 3.

2.8 Answer Extraction

Answer extraction techniques for almost every system involved identifying all possible answer candidates from within the returned passages, and then scoring these candidates to return the one with the best score. The distinguishing feature of any system was the method of scoring these answer candidates.

The answer extraction method used by many QA systems is to filter out any candidates with a named entity type that did not match the EAT of the question, and then to score the remaining candidates according to their proximity to keywords from the question. As a basic heuristic this works reasonably well, but Breck *et al.* (2001) showed that it is inherently limited. In their experiments they found that, even with perfect EAT classification, sentence selection and named entity tagging, it was only possible to correctly answer 59% of the questions from TREC 9 using the above method. The problem this highlights is the frequent occurrence of multiple NEs of the same type in an answer sentence. It was exacerbated by the fact that over a quarter of the questions were given (by a human annotator) the EAT of DefaultNP because they could not be more precisely classified, despite a reasonably sized NE taxonomy (24 types). However, even if the DefaultNP questions were ignored, the accuracy was still only 69%. This argues not only for a larger NE taxonomy, but also for answer selection methods with more discriminatory powers.

More linguistically-aware scoring methods looked not at distance according to word proximity, but measured similarity in other ways, such as syntactically or se-

mantically. One scoring method used a number of times combined both semantic and syntactic similarity. This involved selecting sentences that were paraphrases of the declarative form of the question (that is semantically similar to the question), and writing rules that scored entities within those sentences according to where they matched a syntactic answer pattern. Kaisser and Becker (2004) manually created declarative rephrases for a hand created set of question patterns such as $When + did + NP + V\ INF + NP|PP$, with rules for extracting and type checking the answer from the pattern. The construction of these patterns was very time-consuming and only 157 were completed, making coverage a problem. Their evaluations found that when the patterns were triggered, the accuracy was very good but too often they had to fall back to less linguistically motivated methods. Tanev *et al.* (2004) attempted to rectify this coverage problem by rephrasing each question as it is parsed according to a small set of hand crafted rules. They then matched this transformed parse against the syntactic trees of the parsed passages, using a thesaurus to allow synonym matching. While this method found a transformation for every question, it only allowed one rephrase and so actually had less coverage than Kaisser and Becker's (2004) method. Amaral *et al.* (2005) used a similar method to Kaisser and Becker (2004), but with a more powerful flexible description language and many more patterns. They achieved the highest accuracy (64.5%) at CLEF, establishing the validity of the method, provided huge resources are available.

Dependency parsing provides similar discriminatory power to syntactic tree matching, but without the inflexibility inherent in a method based on word order. Han *et al.* (2004) experimented with adding a dependency based component to their proximity based ranking score and found they achieved an improvement of 3.5% in their factoid accuracy. They used the Conexor FDG parser to extract these dependencies, but ignored the type of dependency (eg, *subj*, *obj*, *agt*) which this parser provides. Cui *et al.* (2004), using MiniPar as their dependency parser, found a balance between strict dependency matching with its associated low coverage (eg, Fuchigami *et al.* 2004) and the discarding of dependency types. They built a statistical model of similarity between dependency relations using question-answer pairs from previous TREC tasks and then extracted dependency relation paths between entities and calculated

a similarity score based on the full relation path. Their results showed a significant accuracy improvement over their baseline proximity-based method, particularly for those questions where the EAT could not be precisely typed.

Going a step beyond dependency parsing, similarity can also be measured in terms of matching predicate-argument structure, or frame elements. While not used by any of the systems at the forums under review, there are QA systems based on these more semantic notions of similarity. Robust Minimal Recursion Semantics (RMRS) was suggested by its creator (Copestake 2003) as a useful representation for use in QA, and this was trialled in the experiment described in Leidner (2002) which used predicate-argument matching from RMRS representation. This system appears to be more a proof-of-concept, rather than a fully developed QA system, but it achieved reasonable results (MRR of 0.463 on TREC-8 data) with very little resources or tuning. The system described in Narayanan and Harabagiu (2004) uses both predicate-argument structure based on PropBank (Kingsbury *et al.* 2002) and frame elements from the FrameNet project (Baker *et al.* 1998), tagged with an SVM classifier to identify key elements from a question and match them to potential answers, and find that this sort of information is particularly useful for complex questions.

Various groups have tried to go beyond similarity based methods by creating logical forms (Quaresma and Rodrigues 2005, Greenwood *et al.* 2002) or semantic networks (Hartrumpf 2005) from questions and passages to try and use more intelligent resolution methods. While Harabagiu *et al.* (2005) found that they could increase their accuracy by 12.4% by allowing a logical prover to over-ride the originally selected answers when confident, in general methods using formal logic are constrained by to a lack of background knowledge. Tanev *et al.* (2004), who framed the question answering problem as an entailment task, tried to semi-automatically acquire this real world knowledge but discovered that it was only used in 7 out of 230 questions. In fact, Hartrumpf (2005) found, when they experimented with adding background knowledge and inference rules, that the majority of questions were semantically similar to a sentence containing the answer and thus similarity measures were sufficient in most cases.

A wide variety of scoring systems were used for answer extraction in the surveyed

systems. It was notable that, while the top NTCIR systems were still using the standard proximity-based method, all of the top performing QA systems at TREC and CLEF had focussed on more linguistically aware answer selection methods. This may be a factor in the relative performance difference between the forums, particularly given that the Japanese questions have been linguistically more complex. The methods that performed well were those that looked for sentences that were similar to the question, with similarity being measured both syntactically and semantically. Coverage was an issue for judging semantic similarity since this depended on either hand written paraphrases or linguistic resources such as thesauri, which are time consuming to create. Various models for measuring similarity are outlined in Chapter 4, and Chapter 6 details a novel method for measuring semantic similarity and then using this for the purposes of answer extraction.

2.9 Summary

Question answering is a form of information access, where a user can define their information need in the form of a natural language question. Question answering systems have been written to respond to questions of varying levels of complexity, from simple factoid questions, to more open-ended questions such as those requiring a definition or a reason as an answer. Recent research in question answering has been supported by the objective evaluation and comparison opportunities facilitated by the TREC, CLEF and NTCIR forums.

A standard 3-part pipeline architecture for question answering systems has emerged through these forums, comprising of a question processing module that attempts to classify the expected answer type of the question, a passage retrieval module that retrieves passages likely to contain an answer and an answer extraction module that pinpoints the most likely answer candidate within the returned passages. Machine translation can be added at various points in this architecture to create a cross-language question answering system. Since the aim of this research is to investigate using linguistically motivated methods to improve accuracy in question answering, this thesis focusses on the question processing and answer extraction modules which

would appear to benefit most from the linguistic knowledge available. Question answering specific passage retrieval techniques have been able to produce answer bearing passages for almost every question in some systems, and hence the use in this research of 'oracle' passage retrieval is not unrealistic.

Evaluation methods have been a focus of the various evaluation forums, and through the years some standard metrics have been publicised. Mean reciprocal rank (MRR) was one of the early metrics used, but it is considered a lenient evaluation since it gives credit to correct answers which are returned after incorrect answers. This form of evaluation is more appropriate for information retrieval, where a user can judge relevance for themselves, than for question answering, where a user can not necessarily be expected to recognise an incorrect answer. Other evaluation metrics used include accuracy, confidence-weighted score and F-measure. All of these measures first require a correctness judgement to be made on individual questions. While in some cases this is straightforward, complications have arisen from issues of answer granularity, exactness and, in the case of definitions, completeness. All of the evaluation forums provide end-to-end system scores only, but discussion amongst participants has often suggested a desire for individual module evaluations. Many research groups provide some form of modular evaluation within their system reports, but this is not yet done in a systematic manner.

The tasks evaluated at each of the forums have been getting gradually more difficult, moving towards a goal described in a research roadmap compiled in 2001 (Burger *et al.* 2001). This roadmap envisages a user asking a question and getting back a correct answer in a coherent format, at the appropriate level for the user. Many milestones defined in the roadmap, including those requiring inference, resolution of conflicting answers and user modelling have not yet been achieved, but the tasks have become more realistic. The first stage in this was requiring single, exact answers to real factoid questions, or else a definitive statement that the answer could not be found. More recently the tasks have included more complex questions requiring lists of answers, definitions and the understanding of context. Looking ahead the immediate future plans for each forum are focussed on being able to answer more difficult questions before moving on to issues of user modelling and inference.

As this thesis will be focussing on Japanese question answering, issues peculiar to Japanese text processing are particularly relevant. As a non-segmenting language, tokenistion is an important step in any text processing, and the inconsistency in the definition of a token provides problems in many areas of text processing, but particularly pattern matching. Another issue involves the Japanese characters—Japanese can be written using four different alphabets which causes problems not only for the level of orthographic variation this enables, but also because this mandates a character encoding set that can encode thousands of characters. There are currently at least three incompatible character encoding formats for Japanese, and this adds an extra layer of complexity when trying to integrate tools and resources from different sources, with may use different character encodings.

A review of the results from recent question answering tasks showed that cross-language question answering systems had lower accuracy than monolingual systems in general, but that the one system that was designed as a cross-language system, rather than a modified monolingual system, did much better than other cross-language systems. In a similar fashion, passage retrieval was a limiting factor for many question answering systems, but those that focussed on QA specific passage retrieval techniques were able to create passage retrieval modules with almost 100% recall. This did not, however result in question answering performance of anywhere near 100%. One of the most interesting observations that could be made after reviewing all the results was that while systems at TREC and CLEF achieved comparable accuracy on tasks involving English and other European languages, the systems at NTCIR which processed Japanese questions lagged well behind in terms of accuracy.

One reason of the performance difference between Japanese and other systems may have been the answer extraction techniques used. The defining factor in answer extraction modules was the method used to score potential answer candidates. While many systems, including all the top Japanese systems, just looked at candidates with a named entity type the same as the EAT, and scored it according to its proximity to keywords, the better performing systems used similarity measures other than proximity. Syntactic similarity was found to be a reasonable indication of an answer bearing sentence, while syntax combined with semantic similarity was very accurate,

but suffered from the low recall associated with many deep linguistic methods. Various people have tried using logical inference and real world knowledge, but found that semantic similarity was sufficient for finding answers to the majority of questions.

Differences in question processing modules revolved around the size and shape on the named entity (NE) taxonomies used as classification labels, and the techniques used to classify a question's expected answer type (EAT) according to this taxonomy. All the better performing QA systems used large NE taxonomies (greater than 50 types) but there was a lot of variation in classification techniques. Despite the success of a few systems that used machine learning techniques, most systems elected to put their efforts into rule-based systems with hand crafted rules and patterns. The lack of training data may be one factor contributing to this trend. Chapter 3 will investigate the question classification options available for small amounts of labelled question data, and how they perform over NE taxonomies of different sizes.

Chapter 3

Question Classification

Question classification is the primary goal of a QA system's question processing module. A review of current QA systems shows that the defining differences in question classification methods are the size and shape of the classification taxonomy and whether the classification uses pattern matching or machine learning techniques. This chapter details question classification experiments that investigate the classification accuracy achievable over named entity (NE) taxonomies of different sizes, using both pattern matching and machine learning techniques. The data set used is a set of 2000 Japanese questions described in Sekine *et al.* (2002b).[1] Each of the questions in the question set have been annotated with question type (who, what, when...), the expected answer type (EAT) taken from Sekine's Extended Named Entity (ENE) hierarchy[2] (Sekine *et al.* 2002a), and the answer to the question with the document ID of its source document. As this is quite a small data set to use as training data for machine learning, these experiments will examine particularly the validity of using machine learning when only small amounts of training data are available. The aim at this stage of the research is to explore the effects of different techniques, including different feature sets, on classification accuracy across different sized taxonomies. The effectiveness of the various taxonomies in the full question answering process will be examined in Chapter 6.

[1] available from `http://www.cs.nyu.edu/~sekine/PROJECT/CRLQA/`

[2] `http://nlp.cs.nyu.edu/ene/`

The first section of the chapter describes named entity taxonomies in general and how they can be used for question answering. Details of the six taxonomies that are used in the experiments are then presented. The next section explains the pattern matching experiment that was conducted to investigate the accuracy and coverage possible with hand-crafted patterns. This experiment highlighted the problems that can be caused by the Japanese text processing peculiarities that were mentioned in Section 2.5. The machine learning experiments described in Section 3.3 use TiMBL, a memory based learner and four different feature sets, incorporating different linguistic resources. Results of each experiment, over the 2000 question set (with the machine learning results using leave one out cross validation), and over a held out 100 question data set from QAC-1 are discussed in Section 3.4, followed by a summary of the general findings.

3.1 Named Entity Taxonomies

A named entity taxonomy is a set of named entity types, which may be organised into a hierarchy. Their main use in question answering is to narrow down the set of possible answers, though they can also serve to select the most appropriate answer extraction strategy, or allow context sensitive query expansion at the passage retrieval stage. Strictly speaking, a named entity is an entity identified by name, such as *Clinton* (of type PERSON) or *Japan* (of type COUNTRY), but the named entity taxonomies used for information extraction, summarisation and question answering have generally included other classes such as DATE and MONEY, since these describe information often requested and easily extracted.

As seen in Section 2.7, there are differences of opinion over the ideal size of the named entity taxonomy used for question classification in QA systems. Those researchers that support small type sets, of around 4 to 12 types, argue that classification accuracy is important to the rest of the QA process and that high accuracy is only achievable when the set of labels is small. The groups that use larger sets reason that the discriminative power of a richer taxonomy makes up for any loss in accuracy.

As well as size, the other variation seen in NE taxonomies is shape: a taxonomy

can have either a flat or a hierarchical structure. The advantage of a hierarchical taxonomy over a flat one is that a hierarchical taxonomy allows for the use of soft decision making strategies.

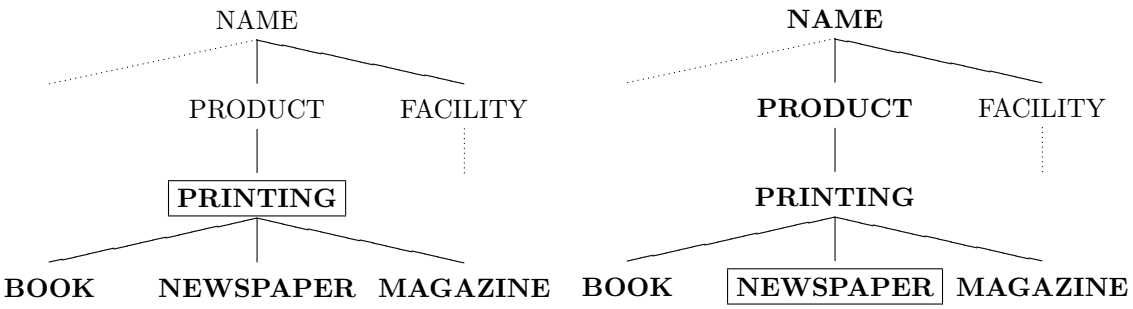
Two examples of these soft decision making strategies involve classifying the question with either its most likely EAT, or else the lowest common parent of multiple potential types. Then, during answer extraction, other NE types nearby in the hierarchy may be considered. Figure 3.1 shows the types that might be considered for each strategy for a question like *What was Sir James A. H. Murray famous for editing?*. For the lowest common parent strategy (Figure 3.1(a)), the question classifier could return PRINTING and the answer extractor might only look at candidates of that type and the types below. If, instead, the strategy is to pick the most likely candidate (Figure 3.1(b)), and if that was considered to be NEWSPAPER, the answer extractor could examine candidates within a certain distance of NEWSPAPER, up, down and across the tree.[3] The score of an answer could depend then on the distance of its type from the EAT returned by the question.

In order to investigate the effects of different classification techniques on taxonomies of different sizes and shapes, I have defined 6 named entity taxonomies that will be used in the question classification experiments. As the question data is annotated with types from Sekine's ENE hierarchy, all the taxonomies are based on that hierarchy, and are created by merging various types into set categories. All the taxonomies are different sizes and, to allow for the use of soft decision making answer extraction strategies later on, two of the larger taxonomies are hierarchical. Each of the taxonomies is described below, with the vital statistics of each summarised in Table 3.1.

The first taxonomy is the full Extended Named Entity (ENE) hierarchy, version 4. This is hierarchical and contains 148 types. A section of this taxonomy was shown in Figure 2.7 and the full version is reproduced in Appendix A.6.

Li and Roth's (2006) work on question classification by machine learning uses a coarse and a fine-grained taxonomy. In order to make some rough comparisons with

[3]Sir James A. H. Murray was the first editor of the Oxford English Dictionary.



(a) Lowest common parent strategy

(b) Most likely type strategy

Figure 3.1: Soft decision making strategies using a hierarchical Named Entity taxonomy. The boxed label is the EAT returned by the question classifier, and the bolded labels are the types of named entity the answer extractor will consider.

Label	Size	Hierarchical?	Comment
NE4	4	No	based on Li and Roth's coarse grained taxonomy
NE5	5	No	NE4 with ORGANIZATION added
IREX	9	No	standard IREX supplemented with a NUMERIC class
NE15	15	No	drawn from top two levels of ENE
NE28	28	Yes	augmented NE15, hierarchically splitting NUMBER, TIME and TIME PERIOD
ENE	148	Yes	version 4 of Sekine's ENE

Table 3.1: Summary of the characteristics of each Named Entity taxonomy used for question classification in this thesis

this work, I have tried to approximate their coarse grained set, which consisted of ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC. For the data sets to be used here, none of the gold standard EAT classifications correspond to the ABBREVIATION or DESCRIPTION classes, so I have defined two small sets, NE4 and NE5, where NE4 consists of THING, PERSON, LOCATION and NUMBER, and NE5 adds an ORGANIZATION class. In NE4, organizations are considered part of the PERSON class. Both of these sets are flat taxonomies.

The IREX named entity set (ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY and PERCENT) was originally defined for an NE extraction task in the Japanese IREX project in 1998 (Sekine and Isahara 1999). It has since been one of the standard NE type sets used in systems at the NTCIR workshops. I have defined a flat taxonomy of 9 types, based on the IREX set, but with an added NUMBER class for any numbers not related to money or percentages.

The taxonomies NE15 and NE28 are mid-range taxonomies that allow more expressive type labels, without being as fine-grained as the full ENE. NE15 is selected mostly from the top and second layer of ENE and is a flat taxonomy detailed in Appendix A.4. NE28 expands on NE15 by allowing further divisions of NUMBER, TIME and TIME PERIOD, in a hierarchical manner. This is illustrated in Appendix A.5.

These six taxonomies were selected to provide a range of sizes that could later (in Chapter 6) be used in answer extraction. The three smaller taxonomies were designed to be similar to others that have been used in other question classification experiments, for purposes of comparison. The mid-range taxonomies were defined for this research to provide moderate discriminative power during answer extraction, without requiring a very large, detailed type set. NE28 was used to explore the differences a hierarchical taxonomy can make. The Extended Named Entity hierarchy, as well as being used for the original gold standard classification, provided a representative of the large taxonomies that are supported by many groups for their discriminative powers, while being criticised by others for being too difficult to enable accurate classifications. The following sections describe the question classification experiments that used these taxonomies.

3.2 Pattern Matching

Pattern matching was the most common technique used for question classification in all three evaluations described in Section 2.4. To investigate the issues involved with this technique, and also the accuracy achievable, the first question classification experiment used a set of manually created pattern matching rules for Japanese. ChaSen (Matsumoto *et al.* 1999) was used to tokenise the question, and the first stage attempted to identify the question type. The question types were annotated in the original data with 17 different question types. In this experiment the question types なに＋漢字 *nani+kanji* "what+<kanji word>" and なに＋ひらがな *nani+hiragana* "what+<hiragana word>" were collapsed into one class なに *nani* "what" since the alphabet used for the word after "what" did not appear to give useful information. The 16 question types used for classification are shown in Figure 3.2. Most of the rules for this stage involved identifying one of the question words or its orthographic variants. Questions that include 何 *nani* "what" were further processed so that fragments such as "what place" were classified as WHERE. This question word matching was more complicated than it would be in English, because of the tokenisation issues which, for example, caused *dare/ka* ("who"/<QM>) to be instead treated as *dareka* ("somebody"). Some of the patterns used to find the いつ *itsu* "when" question type are shown in Figure 3.3. Once some rules were created to deal with these issues, question type could be detected with an accuracy of 98.4%. Approximately a third of the remaining errors could be attributed to tokenisation problems, while with another quarter, the gold standard annotation was inconsistent, for example classifying some questions as *how many* and others as *what* when the question asked "what number of pieces...". The pattern matching rules were constructed by examining this data set; this may be one reason for the very high precision. To test this, the same pattern set was used to classify question types for a smaller held-out set of 100 questions. For this set the question type was correctly classified in 93 out of 100 questions, with 4 errors due to segmentation issues. The other 3 errors could all be considered valid alternatives to the gold standard. For example, one of the questions, *What modern prefecture and city was Miyamoto Musashi born in?*, was classified as a WHERE

いくつ	*ikutsu*	"how many"
いくら	*ikura*	"how much"
いつ	*itsu*	"when"
どう	*dō*	"how"
どこ	*doko*	"where"
どちら	*dochira*	"which"
どの	*dono*	"which/what"
どれくらい	*dorekurai*	"how long/far/much"
どんな	*donna*	"what kind of"
何%	*nanpāsento*	"what percent"
何時	*nanji*	"what time"
何人	*nannin*	"how many people"
何年	*nannen*	"what date"
なに	*nani*	"what"
誰	*dare*	"who"
は	*ha*	used for sentences ending in a topical marker

Figure 3.2: Question types used for question classification

いつ	*itsu*	"when"
いつまで	*itsumade*	"until when"
いつ頃	*itsugoro*	"approximately when"
いつごろ	*itsugoro*	orthographic variant of above
いつか + ら	*itsuka* + *ra*	systematic incorrect segmentation: *itsuka* means "sometimes", while *itsukara* means "from when"

Figure 3.3: Examples of patterns used to identify WHEN questions

question, but the gold standard classification was WHAT.

Once the question type was determined, this was used to help predict an answer type. Different rules sets were used depending on the question type, but most of the rules looked for a word (called here the question focus) at a particular relative distance from the question word, and checked whether that word was a recognised named entity type. A small set of indicative words was hand-constructed, linking certain terms to their most likely category, often when a word had a very common synonym, such as 点数 *tensū* "point" for ポイント *pointo* "point". The manual construction of this set was time consuming, since terms had to be limited to those

that specifically indicated one entity type only. Often when a term was added to the list, it caused a negative impact on overall accuracy and had to be removed. With time and resources to conduct large-scale experiments, expanding this related word set should have a positive effect on accuracy, but with the small data set and limited resources available it was difficult to create a set that was general purpose and not over-fitted to the data being used.

On many occasions, the question focus did not indicate an NE type, or the question word was not found explicitly in the question. In this case, the most likely NE type was returned. This was not always the most frequent type for that question type, but more specifically, the most likely type given that none of the preceding rules could be applied.

The logographic nature of the Japanese kanji character set was in some ways an advantage, since, for example, any word that contained the character 時 was concerned with "time", regardless of its pronunciation. This did, however, lead to quite a complicated expression of a basic rule set, since the rules sometimes needed to match full words, and other times characters within words. They also had to cover instances when the phonetic representation was used instead of a key character.

The pattern matching process used was a simple two step process, classifying the question type by identifying a question word, and then using this question type to select a rule set for determining the EAT, where all the rule sets generally found the question focus and determined whether it was related to any answer type. Despite the simplicity of the rules, they were complicated and took many days to create and refine, in large part due to issues of Japanese tokenisation and orthographic variation. These issues forced the creation of rules that were specific to one character encoding and one tokenisation tool, hence limiting reuse. The results of this experiment are discussed in Section 3.4, along with those from the machine learning experiments described in the next section.

3.3 Machine Learning

Machine learning techniques are often promoted as being more robust and reusable for natural language applications, since a system can often be ported to a new language or genre just by changing the training data. Li and Roth (2006) showed in their experiments that machine learning could be an effective way to classify expected answer types for their 1000 English question data set drawn from past TREC competitions. However, their work also showed one of the problems with machine learning—they used 21,500 annotated questions as training data for their system, including 500 questions specifically constructed for rare types, and often this quantity of data is just not available. Lack of question-specific data was shown to hurt performance of statistical parsing of questions (Judge *et al.* 2005) and, particularly relevant in this case, the shortage of any resources in languages other than English can mean that statistical methods don't appear viable. The aim of the machine learning experiments reported here was to investigate whether machine learning is an effective technique for question classification when only small amounts of training data are available. In this instance, 2000 annotated Japanese questions were used as training data.

The learner used for these experiments was TiMBL (Tilburg Memory Based Learner) version 5.1, a memory based learner developed primarily for NLP tasks (Daelemans *et al.* 2004). TiMBL classifiers can be trained very quickly, and hence allow a fast turnaround when evaluating the effects of different features. The learning algorithm used was TiMBL's IB1 algorithm, with features weighted by their Gain Ratio value. While some of the features (detailed later) were binary, many were symbolic, such as words or named entity types. In order to use the information that, for example, `country` is more similar to `province` than `person`, the modified value difference metric (MVDM) was used as a similarity measure between feature values.

The classifiers were evaluated first using a 'leave out one' strategy, for each of the 2000 questions. Next, to see how well the training generalised to unseen data, 100 questions from the question set used in the QAC1 task at NTCIR-3[4] were manually

[4]available from `http://research.nii.ac.jp/ntcir/permission/perm-en.html`

いくつ	*ikutsu*	"how many"
いくら	*ikura*	"how much"
いつ	*itsu*	"when"
どう	*dō*	"how"
どこ	*doko*	"where"
どちら	*dochira*	"which"
どの	*dono*	"which/what"
どれくらい	*dorekurai*	"how long/far/much"
どんな	*donna*	"what kind of"
なに	*nani*	"what"
だれ	*dare*	"who"

Figure 3.4: List of Japanese question words used as features for question classification by machine learning

annotated with their EAT, and a separate evaluation was run on these, using the first 2000 questions as training data.

3.3.1 Features

The feature sets all had the same basic format: presence or absence of each of a list of question words, and then context information for each question word (if it appeared). The list of question words was built from the keywords used to identify question types in the pattern matching experiment. This produced 11 words, each of which could have some orthographic variation. These 11 question words are shown in Figure 3.4.

Two content words from the start and from the end of each question were also used as features. The importance to question classification of the words at the start of a question has been documented by Müller (2004), but this work only analysed English questions. In Japanese, question words are more likely to be found at the end of a question. Words from the start and end of the question were used to keep the features as non language specific as possible. In the following discussion the different sorts of features will be demonstrated using the following question.

牛窪	多喜夫	さん	は	現在	埼玉県	の
ushikubo	*takio*	*san*	*ha*	*genzai*	*saitamaken*	*no*
Ushikubo	Takio	TITLE-HON	TOP	current	Saitama	GEN

どこ	に	道場	を	開い	て	いる	か
doko	*ni*	*dōjō*	*wo*	*harai*	*te*	*iru*	*ka*
where	DAT	dojo	ACC	is operating			QM

"Where in Saitama is Takio Ushikubo currently operating his dojo?"

Feature Set 1: Word-based only

The first set of features used were based on words only (or more precisely, on morphemes according to the ChaSen definition), and hence could be considered to use similar information to that available in the pattern matching experiment. As well as the binary features indicating the presence or absence of each question word, the two words either side of each question word were added if the question word was present, as well as the two words at each end of the question (ignoring the question marker if it was present). This gave a total of 59 features, although any one question would normally only have up to 9 instantiated (if only one question word was present). The instantiated features for the example question would have been then:

first word: *ushikubo*
second word: *takio*
last word: *iru*
second last word: *te*
doko present: yes
word one before doko: *no*
word two before doko: *saitamaken*
word one after doko: *ni*
word two after doko: *dōjō*

Feature Set 2: Using SProUT Tags

Both Li and Roth (2006) and Kimura *et al.* (2005) emphasised the importance of using features that encode linguistic knowledge, especially when the amount of training data is small. The easiest linguistic information to add to the data was the

base form of inflected verbs and part of speech tags which were both are available from ChaSen. In this experiment, the part of speech was primarily used to filter out non-content words like case markers and verb endings.

In order to add more semantic knowledge to the data, SProUT (Becker *et al.* 2002) was used to tag named entities within the question. SProUT uses a fairly small tag set, predominantly tagging names as `ne-person_rel`, `ne-organization_rel` or `ne-location_rel`, and specifying location types where known. A new feature set was created, again recording the presence of question words, and their surrounding context, but if any of the surrounding words had been tagged by SProUT, the SProUT tag was used instead of the word. This has the effect of making, for example, all countries look the same to the classifier, increasing feature counts while ignoring irrelevant information (such as which country). Using SProUT tags also had the side effect of combining multi-word entities into one 'word' in this instance. The instantiated features for the example question are different to those for Feature Set 1 in two ways—firstly content words only are used, replacing verb endings at the end of the sentence and the case markers before and after *doko*, and secondly replacing named entities such as *saitamaken* with the `ne-location_rel` tag.

first word:	*ushikubo*
second word:	*takio*
last word:	*haraku*
second last word:	*dōjō*
doko present:	yes
word one before doko:	`ne-location_rel`
word two before doko:	*genzai*
words one after doko:	*dōjō*
words two after doko:	*haraku*

Feature Set 3: Using Sekine's Named Entity List

A second semantic information source was the gazetteer style list provided with the ENE hierarchy. This list had over 60,000 words and their associated NE type from the taxonomy. For every feature related to a context word or tag in Feature Set 2, another feature was added to record the type of that word if it occurred in the list.

That meant the possible number of features went from 59 to 107. For the example question above, this added the information that *saitamaken* was of type PROVINCE. Looking through the data, these features fired most often for the names of countries and also for position titles (eg, President). This feature added more specific named entity information than the SProUT tags, in a form directly related to the possible EATs.

first word:	*ushikubo*
second word:	*takio*
last word:	*haraku*
second last word:	*dōjō*
doko present:	yes
word one before doko:	`ne-location_rel`
type of word one before doko:	PROVINCE
word two before doko:	*genzai*
word one after doko:	*dōjō*
word two after doko:	*haraku*

Feature Set 4: Using an Ontology

WordNet (Miller *et al.* 1990) has often been used as a source of semantic information in English. There is no Japanese version of WordNet, but the Hinoki ontology was available (Bond *et al.* 2004a). This ontology was semi-automatically constructed from a Japanese dictionary as described by Bond *et al.* (2004b) and has a fairly limited cover of about 30,000 words, with very few proper names. This was used to look up each context word in the feature set to see if it was related to any of the named entity types in the ENE. If a related named entity type was found for any word, it was added as a feature, in the same way that NE types were added in Feature Set 3. For the running example, this adds the extra information that *genzai* "current" is a word relating to TIME.

first word:	*ushikubo*
second word:	*takio*
last word:	*haraku*
second last word:	*dōjō*
doko present:	yes
word one before doko:	`ne-location_rel`
word two before doko:	*genzai*
type of word two before doko:	TIME
words one after doko:	*dōjō*
words two after doko:	*haraku*

Feature Set Summary

Feature set 1 was designed as a baseline set, which was directly comparable to the pattern matching experiment, since it used the same information. Feature sets 2, 3 and 4 were attempts to add semantic information, as used in the Li and Roth (2006) experiment, using whatever linguistic resources were available. Li and Roth used 4 different sources of semantic information: named entity tagging, WordNet, class-specific related words and distributional similarity based categories. In the experiments here Feature sets 2 and 3 relate to named entity tagging, while Feature set 4 uses a similar information source to WordNet. The class-specific related words appear to be similar to the indicative word set used in the pattern matching experiment, but resources were insufficient to create a similar list large enough to be useful, nor was there a pre-compiled distributional similarity list for Japanese.

3.4 Results

The results for each experiment across all the NE taxonomies are shown in Figure 3.2.[5] These results show that for every NE taxonomy, machine learning techniques were more accurate, with the most accurate results significantly higher than pattern matching for all taxonomies, except NE5. Surprisingly, the first feature set, which

[5]Results for QAC-1 test set are shown to only two significant figures, since the data set was only 100 questions.

	NE4	NE5	IREX	NE15	NE28	ENE
PM	0.742	0.732	0.706	0.651	0.565	0.408
words	**0.782**	**0.755**	0.731	0.684	0.623	0.484
SProUT	0.765	0.745	0.726	0.682	0.630	0.494
NE list	0.767	0.747	0.725	0.679	0.627	0.491
ontology	0.772	0.751	**0.734**	**0.691**	**0.640**	**0.495**

(a) Leave-one-out cross-validation

	NE4	NE5	IREX	NE15	NE28	ENE
PM	0.75	0.75	0.69	0.63	0.57	0.42
words	0.73	0.74	0.71	**0.68**	**0.62**	**0.48**
SProUT	0.77	0.75	0.69	0.64	0.61	0.45
NE list	**0.78**	**0.76**	0.69	0.66	0.60	0.45
ontology	0.77	**0.76**	**0.72**	0.64	0.61	0.41

(b) Test set from QAC-1

Table 3.2: Accuracy in question classification, over the full development set using leave-one-out cross-validation and for a held out set of 100 questions from QAC-1, using the development set as training. The first row from both tables has the results from the pattern matching experiment, and the last four show the results from the four feature sets used in machine learning.

was based only on words, achieved the best results for the smaller taxonomies, despite using the same information as the pattern matching. Adding semantic information did not give the expected performance increase, though we begin to see small gains in accuracy due to this extra information as the taxonomies get larger.

Looking at the effects of taxonomy size, we see that accuracy does decrease as the taxonomy size increases. Breaking down the results by class for each taxonomy, for the NE4 taxonomy the easiest class to classify was NUMBER, correctly identified 94% of the time. THING on the other hand was often misclassified as PERSON, and only correctly labelled in 55% of cases. NE5 differed from NE4 by splitting the PERSON category into PERSON and ORGANIZATION which had the effect of increasing classification accuracy on PERSON, probably because it was now a more tightly defined class. The ORGANIZATION class, however, had very low accuracy (50%), most frequently being misclassified as THING, but also often as LOCATION. This reflects the different ways organisations such as companies can be referred to—in some cases they are places of action and in others, agents of action.

The IREX taxonomy is similar to NE5, but with NUMBER split into DATE, TIME, MONEY, PERCENT and NUMBER. While ORGANIZATION and THING are still mislabelled in almost half the instances, interestingly, the accuracy for the PERSON and LOCATION classes goes up from the NE5 results. It appears that despite these classes not changing in any way, the feature sets are becoming more discriminatory with the larger taxonomy. Looking at the new numeric classes, DATE and MONEY were the most reliably identified (89% and 94% of instances respectively), but surprisingly PERCENT was only correctly classified 60% of the time, most commonly being mis-labelled as NUMBER. Looking back at the original features, it appears that SProUT tags 何% *nanpāsento* "what percent" as `percentage_rel`, hence removing the question word 何 *nan* "what" which would otherwise be used as a feature to direct attention to the %. This is a peculiarity of using SProUT that would need to be addressed in future experiments.

Like IREX, NE15 also has a PERSON, LOCATION and ORGANIZATION class. The differences are that the number classes have been split differently, into NUMBER, TIME and TIME PERIOD, and that THING has been replaced by a number of more specific

classes. Classification accuracy for PERSON, LOCATION and ORGANIZATION is almost exactly the same for IREX and NE15. For the number related classes, TIME and NUMBER are detected with accuracies of 86% and 88% respectively. TIME PERIOD is much harder to identify (53% of instances). It is often mis-classified as either TIME or NUMBER. None of the other classes are correctly identified with any great accuracy. Given that THING was only classified with an accuracy of 54%, it is not surprising that sub-divisions of THING are not easily identified. Many of the classes have only a few instances in the data set (only EVENT and PRODUCT having over 20) so there are insufficient training examples to allow reliable classification.

NE28 is the hierarchical version of NE15, allowing more specific classes of TIME, NUMBER and TIME PERIOD while still using the more general classes when appropriate. The other differences involve splitting POSITION TITLE out from PERSON, and FACILITY from ORGANIZATION. FACILITY (which is used for things like *school* and *library*) was shown to be very difficult to identify because, like ORGANIZATION, things of type FACILITY can refer to places or entities, and so they were often mis-classified as LOCATION or ORGANIZATION. Most of the number classes were detected with a high accuracy, suggesting that adding the extra specificity for sub-types of NUMBER is a worthwhile modification. The classification accuracy of PERSON was higher again than NE15, possibly because removing the POSITION TITLE type entities tightened the class definition even further.

When the ENE taxonomy was used for classification, 116 out of 148 types were represented, with only 37 types appearing in the gold standard more than 10 times. Not surprisingly, the classification accuracy over the rare types was generally very low, although many of the numeric types were reliably classified despite very little representation in the training data. In general, numbers were used in very specific patterns and this provides good evidence for classification. Examining the types that did occur frequently, it was interesting to note that PERSON was classified more accurately with this taxonomy than any other. While the class was more tightly defined than in some of the smaller taxonomies, even compared to NE28, which labelled exactly the same entities PERSON, the classification accuracy was 1.5% higher (92.5%). The other classes that were reliably classified were MONEY, many of the time classes,

	NE28		ENE	
	Exact	Lenient	Exact	Lenient
PM	0.565	0.861	0.408	0.618
words	0.623	0.879	0.484	0.595
SProUT	0.630	0.862	0.494	0.602
NE list	0.627	0.861	0.491	0.594
ontology	0.640	0.880	0.495	0.603

(a) Leave-one-out cross-validation

	NE28		ENE	
	Exact	Lenient	Exact	Lenient
PM	0.57	0.82	0.42	0.67
words	0.62	0.82	0.48	0.70
SProUT	0.61	0.83	0.45	0.64
NE list	0.60	0.86	0.45	0.65
ontology	0.61	0.82	0.41	0.60

(b) Test set from QAC-1

Table 3.3: Comparing exact and lenient evaluations for question classification over the hierarchical taxonomies, for the full development set using leave-one-out cross-validation and for a held out set of 100 questions from QAC-1, using the development set as training. The first row from both tables has the results from the pattern matching experiment, and the last four show the results from the four feature sets used in machine learning.

and COUNTRY. The classes that appeared frequently but were often misclassified were generally sub-classes of ORGANIZATION such as COMPANY and GROUP.

To look at the effects of the hierarchical taxonomies, a lenient evaluation was performed on the classifications over NE28 and ENE, using the soft decision making strategy illustrated in Figure 3.1(b). This demonstrates the effective EAT classification accuracy when soft decision making is used in answer extraction. The results, presented in Table 3.3, show that under lenient evaluation, while ENE accuracy is still quite low, NE28 is very good, better than the accuracy achieved for even the smallest taxonomy.

A direct comparison with other results is not possible, since I could not find pub-

lished question classification results from this data set. The Li and Roth (2006) work in question classification by machine learning is the most similar experiment I have seen, using similar methods but differing in using a different (English) data set, and different taxonomies. Comparing their coarse grained classification results with the accuracy of NE4 and NE5 classifications achieved here, they demonstrated significantly higher results with 92.5%, compared to 78.2% and 75.5% respectively. Taking the NE28 taxonomy as the closest to their fine-grained taxonomy, the performance difference was larger with NE28 classification accuracy at 64.0% being significantly lower than their best result of 89.3%. While the level of difficulty of the two tasks may differ (since the Japanese questions were much longer on average than the English questions), this would not explain such a large performance difference.

I hypothesise that there were two main factors that differentiated the experiments. The first obvious difference is the amount of training data. The Li and Roth results reported here were achieved using 21,500 questions for training. In their experiments exploring the effects of training data set size, they found that by reducing the size of the training set to 2000 questions (the amount used here), they lost between 14 and 18% accuracy. While this partially explains the lower results in my experiments, one of the ideas being explored here is the validity of machine learning with small amounts of training data, and hence we are also interested in any other factor that may have led to higher accuracy.

Looking at Li and Roth's detailed evaluation, we see that the types of semantic information they received maximum benefit from were the class-specific related word list and the distributional similarity lists, the two forms of information not used here. Their best combination of semantic features (which was the set without WordNet) achieved a 5% accuracy increase over the combination of WordNet and named entity features. It is not clear how the class-specific related word lists were created, except that they were constructed after the manual analysis of 5000 questions. Experiences during the pattern matching experiment suggest that this sort of resource is difficult to build in a general way without over-fitting to the questions being analysed. It would be interesting to see whether a translation of Li and Roth's lists would be beneficial to classification on this Japanese question set, or whether they are too

closely tied to the set they were created from. Distributional similarity lists however could be built automatically from, for example, the Mainichi newspaper corpus, and could be expected, from Li and Roth's results, to deliver more accurate comparisons.

3.5 Conclusion

The majority of question processing modules at recent evaluation forums use pattern matching techniques for question classification. However manual pattern construction is very time-consuming and language specific. In the pattern matching experiment reported here there were issues particular to Japanese that added to the complexity, and also made it more difficult to re-use any of the patterns in a different application. For the machine learning experiments, the only hand-crafted data used was a list of question words. Other than that, an off-the-shelf tokeniser, named entity recogniser and ontology were the only tools used to produce the data for the features. The results showed that machine learning could achieve better accuracy than pattern matching techniques, with only a small amount of data, and much less manual effort. The machine learning techniques can also be easily applied to other languages, by providing the appropriate training data and a list of question words, so much less manual effort is required to switch languages.

Looking at the results across the different sized taxonomies, predictably the accuracy decreases as the taxonomy size increases. However, looking at the accuracy for specific classes, the larger taxonomies actually led to greater accuracy on some of the more common classes, such as PERSON and COUNTRY, and also on most of the numeric classes, even the less common ones. The greatest contribution to inaccuracies in the larger taxonomies was the vast number of infrequent categories that in smaller taxonomies were all classed together as THING. These classes in the ENE appear to be somewhat arbitrary, a criticism that can also be directed toward Li and Roth's (2006) fine-grained taxonomy. There are no apparent reasons for NATIONALITY being a sub-class of ORGANIZATION but RELIGION a top level class of its own, for example, or, in the Li and Roth taxonomy, why MOUNTAIN is a class, but RIVER is not. Often, the classes in fine-grained NE taxonomies appear to be selected for the purpose of

classifying questions in the question set they are being used with. The conclusion to take from this is that large taxonomies can be used for question classification, but more work needs to be put in to determine which type distinctions will actually be useful at the answer extraction end, and how they can be arranged in a hierarchy. If different types of numeric data can be accurately classified and detected, for example, then expanding the number of numeric types in a taxonomy is beneficial. This is a topic not covered in this thesis, but one which needs to be examined in future research.

This thesis so far has reviewed the question answering field, and then looked at the question classification techniques that can be applied to a small Japanese question set. The second half of the thesis will look at different measures of similarity and how similarity can be used in answer extraction. The results from the question classification experiments outlined here will be used in Chapter 6 as part of the answer extraction experiments, where the effects of the different taxonomies will be explored.

Chapter 4

Similarity

Similarity, as shown in Section 2.8, is the primary means of identifying answer bearing sentences in question answering. Sentence similarity measures are also widely used in natural language processing (NLP) applications other than QA. Since the lack of systematic modular evaluation of QA makes examining answer selection strategies difficult, it is worth looking at the methods used in a recently established task that mirrors the requirements of answer selection quite closely. That task is the PASCAL Recognising Textual Entailment Challenge (RTE) (Dagan *et al.* 2005, Bar-Haim *et al.* 2006). The RTE Challenge is described in Section 4.1 below, and the similarity measuring techniques described later in this chapter are taken from systems that participated in either the RTE Challenges, or in QA tasks.

Measuring semantic similarity between words is a common research area in the field of lexical semantics. The most common methods involve using a thesaurus or a semantic ontology such as WordNet (Miller *et al.* 1990). These resources describe relationships between word senses, most often synonymy, but hyponomy, metronymy and antonymy relationships are also common. The most basic method for using these resources is just to classify two words as related if a positive (not antonymic) relationship is found. Extensions to this basic method may issue a real-valued similarity or distance score between two words, depending on the type of the relationship found between them, or perhaps the length of the chain of relationships between them. Relationships in WordNet are represented by first clustering word senses into concept

groups (or synsets), each with a gloss describing the concept. Concepts are then arranged in an *is-a* hierarchy. `WordNet::Similarity` (Pedersen *et al.* 2004) is a set of Perl modules which implement many different lexical similarity metrics based on WordNet. These metrics involve a variety of techniques, some of which use the lowest common parent of two terms to measure the similarity between those terms, using either the information content of that lowest common parent, or the depth from the top of the hierarchy. Another technique measures the length of the connecting path between terms. A separate set of methods in the package measure relatedness rather than similarity, by measuring the word overlap between the glosses of two concepts.

A technique for measuring lexical similarity which does not use a structured resource like WordNet is distributional similarity, which is based on the idea expressed by Dr J. R. Firth - "You shall know a word by the company it keeps" (Firth 1957). This technique involves comparing words by the set of words they often co-occur with, rather than directly. Distributional similarity, like the relatedness measures from the `WordNet::Similarity` package, will find relationships between words that are not necessarily synonyms or hyponyms because it finds words that are used in the same way or with the same context. For example, a list of words related to *rain*, as returned by Lin's (1998) distributional similarity measure, are *sunshine*, *park*, *mess* and *garden*.

Relational similarity is used here to refer to the similarity of relationships between words, often encoded as syntax. The metrics of relational similarity range along a spectrum of complexity, which aligns with the depth spectrum of linguistic processing tools. Hence word based metrics only require the use of shallow processing tools such as tokenisers and part of speech taggers, and can only approximate inter-word relationships by word collocation or word order. Syntactic based metrics can use more complex tools such as phrase chunkers, syntactic parsers and dependency parsers to abstract away from word order and surface form and compare syntactic 'shape' of sentences. While syntax can provide a lot of information about a sentence, the grammatical relations syntax describes, such as subject and object, are still an approximation to semantic relations. Formal semantics-based similarity metrics are at the far end of the spectrum. They use deep parsers to extract and compare these

semantic relations, attempting to compare the actual meaning of two texts, rather than the expression of that meaning. At each point in this complexity spectrum, it is possible to incorporate ideas of lexical similarity—that is semantic similarity between two words.

In the following sections I explain the PASCAL Recognising Textual Entailment task, showing that it has similar requirements, and therefore should use similar techniques, to those in the answer selection stage of question answering. I then describe some similarity metrics and associated tools that fit at various points on the complexity spectrum, and discuss how lexical similarity can be used at each level.

4.1 Recognising Textual Entailment

The PASCAL Recognising Textual Entailment (RTE) Challenge requires recognising whether a text *T entails* a hypothesis *H*. The definition of entails, as used at the RTE Challenges, requires that a typical person could infer the hypothesis from reading the text. Hence, the text:

<T> The Alameda Central, west of the Zocalo, was created in 1592.

does entail the hypothesis:

<H> The Alameda Central is west of the Zocalo.

but does not entail:

<H> The Zocalo was created in 1592.

The RTE Challenges isolate the entailment task, which is used in applications such as question answering, paraphrase acquisition, information retrieval and information extraction, allowing the methods used in that task to be evaluated directly, rather than within a full system where there are many interplaying factors. To maintain the relevance of entailment to these applications, RTE organisers formed their text-hypothesis pairs from data used by previous evaluation forums for QA, IR, etc. For example, from the question *When did Cardinal Juan Jesus Posadas Ocampo die?* they used a possible answer-bearing sentence as the text:

> <T> The two suspects belong to the 30th Street gang, which became embroiled in one of the most notorious recent crimes in Mexico: a shootout at the Guadalajara airport in May, 1993, that killed Cardinal Juan Jesus Posadas Ocampo and six others.

and formed the hypothesis from the potential answer and the question:

> <H> Cardinal Juan Jesus Posadas Ocampo died in 1993.

While the concept of entailment appears to suggest a basic need for logical inference and reasoning to recognise relationships such as *X was killed* entails *X is dead*, in the first RTE Challenge in 2005 the top 5 systems used combinations of statistical and word overlap methods only. Vanderwende *et al.* (2005) carried out a manual evaluation of all the test data from the first RTE Challenge and discovered that about half the entailment decisions could be made using only a thesaurus and syntactic information. In other words, entailment could often be detected by measuring similarity, which was encoded in both the words and the relationship between the words, as expressed in syntax, referred to in this chapter as relational similarity. The complexity spectrum of similarity metrics used in the RTE Challenge, and also in QA are reviewed in the following sections.

4.2 Word-based Metrics

Word based similarity metrics require very little pre-processing of text. At a basic level, this might just be tokenisation — breaking the text up into words. A slightly more intelligent metric might make use of part of speech tags (tagging words as nouns, articles etc.) and perhaps lemmatisation where words are converted to their base, uninflected form before matching. In the case of example (12) below, lemmatisation would convert *walked* to *walk* and *strolled* to *stroll*, and both would be tagged as verbs.

(12) a. John walked in to town

b. John strolled in to the city

Most linguistic processing tools at this level have a high coverage as well as high accuracy, so while the text models used for comparison do not have a lot of information, the information they do have is generally complete and correct.

The bag-of-words model describes any representation of text that depicts only the presence or absence of words and not the order. This usually takes the form of a n-dimensional vector for a vocabulary of n words. In the case where the vector terms can only be one or zero, the dot product of two vectors is the equivalent of the basic word overlap between the two sentences they represent. Often the vector terms can take on real values that might indicate the frequency of a term within the text, or else the weight given to that term. For example, nouns may be given higher weights than other terms in the question, giving a weighted word overlap. When this comparison is controlled for sentence length, in order that long sentences are not given undue weight, we have the *cosine similarity* metric (Witten *et al.* 1999), common in information retrieval.

Similarity metrics based on bag-of-word models are extensively used because they are quick, simple to implement and work well as an estimate of similarity. However they suffer from being at the same time too strict and too lenient: too strict because they require an exact word match and so synonyms like *street* and *road* in example 13 will not match, even though they may represent the same physical entity; too lenient because there will be an exact match between the two sentences in 14 below, despite the obvious difference in meaning.

(13) a. Jess walked down the street

b. Jess walked down the road

(14) a. I see what I eat

b. I eat what I see

One method for dealing with the first issue, detailed in Corley and Mihalcea (2005), uses the various word similarity metrics based on WordNet (Miller *et al.* 1990) that were mentioned earlier and then combines individual word similarity scores to obtain an overall text similarity. If synonyms such as *road* and *street*, and *walk* and

stroll are recognised as being similar, then a bag-of-words based similarity model would correctly give a high score to examples 13 and 12. The second issue is more complicated—methods involving longest substring (Papineni *et al.* 2002, Lin and Och 2004) or edit-distance (Akiba *et al.* 2001) are used for the evaluation of machine translation or text summarisation, but these are only appropriate when the goal is lexical similarity to reference texts, rather than semantic similarity. Even in this case, some sentences will be problematic. Many edit-distance algorithms would give the same score to example (14) and example (15) below, but while swapping two words in (15) has little effect on the meaning, doing the same to (14) does change the meaning significantly. Details like this are difficult to evaluate without some idea of relational similarity.

(15) a. I ran quickly to school

b. I quickly ran to school

4.3 Syntax-based Metrics

Syntax based similarity metrics use tools of a deeper level of linguistic knowledge such as phrase chunking or full syntactic parsing to detect and compare the syntactic structure of sentences. These methods compare the 'shape' similarity between sentences by abstracting away from the original surface words and pattern matching more abstract objects, such as noun or verb phrases. More informative abstractions often used in these patterns are named entity tags, assigned by a named entity recogniser. Figure 4.1 shows some possible patterns where words have been replaced by their part of speech or named entity type, and full phrases replaced with their phrase type, such as *noun phrase* (*NP*) and *prepositional phrase* (*PP*). This sort of pattern has been used in answer extraction modules that transform the question to its declarative form such as that described by Kaisser and Becker (2004). Matching at a syntax level can also be done using various tree matching algorithms to match syntactic trees. Emms (2006) used this method to select answer bearing sentences in a question answering task and found that the tree matching method worked better than word sequence

PERSON_NAME *was born in* DATE
PERSON_NAME *was born in* LOCATION
When did NP V_INF PP

Figure 4.1: Examples of sentence patterns using named entity, phrase chunk and part-of-speech abstractions

similarity measures when comparing sentences like example (16) below, where the question word sub-tree maps to a multi-word noun phrase which contains the answer to the question.

(16) a. What do child processes inherit from their parent processes?

b. A child process inherits the owner and permissions from the ancestor process

Deeper processing tools such as dependency parsers allow a more abstract comparison which moves away from the reliance on word order to provide meaning. Dependency relations, which can be returned by dependency parsers, are more complex comparison units which represent two words from the sentence and the syntactic relationship between them. These relationships include, for example, those between a noun and its modifier or a verb and its subject. Basic Elements (BE) were designed by Hovy *et al.* (2005) to represent these relationships and use the format *head*|*modifier*|*relation*. An example of a sentence and its associated BEs, taken from Hovy *et al.* (2005) is shown in Figure 4.2. A similarity metric that matched BEs would then be able to match the sentences in example (15) because in both cases the BE *ran*|*quickly*|*mod* should be formed. The original goal of BEs was to have a unit that could be automatically generated and which was useful for evaluating summaries created by document summarisation systems, by comparing them to reference summaries written by humans. As well as summarisation, BEs have since been used by Nicholson *et al.* (2006) for measuring entailment at the second Recognizing Textual Entailment Challenge (Bar-Haim *et al.* 2006). Their official results were disappointing, partially due to a mismatch between training and test data for their classifier, but also because strict BE matching was not able to handle multi-word paraphrases

"two Libyans were indicted for the Lockerbie bombing in 1991"

libyans|two|nn
indicted|libyans|obj
bombing|lockerbie|nn
indicted|bombing|for
bombing|1991|in

Figure 4.2: An example of a Basic Elements (BE) sentence representation

such as *town of Jenin* for *Jenin*. To allow these terms to align under a BE model, the comparison method would need to look for one-to-many BE matches. Cui *et al.* (2004) does this by comparing dependency relation paths, using a statistical model of path similarity. While this will now correctly identify example (17) as matching and example (14) as non-matching, dependency relations still fail for comparisons such as example (18) below.

(17) a. What American revolutionary general turned over West Point to the British?

b. ... Benedict Arnold's plot to surrender West Point to the British

(18) a. Who was the ball given to?

b. I gave Martin the ball

4.4 Semantics-based Metrics

The interword relationships described by syntax, including the dependency relations, are still approximations to the 'real' relationships between entities in text. In formal semantics, semantic roles (also known as thematic or theta roles) describe the role of an entity with respect to a verb according to the semantics rather than expression of the text (Dowty 1989). These roles can include, for example, AGENT, the performer of an action, INSTRUMENT, an object used to perform the action and THEME, the object acted on. The benefit of using these roles in analysis is that they

are stable over any expression of the same truth conditions. For example, (19) and (20) below have different syntactic analyses despite describing the same situation, with the subject being *John* in (19) and *the door* in (20).

(19) John opened the door with that key.

(20) The door was opened by John with that key.

In contrast, for both of the examples a semantic analysis says that *John* is the AGENT, *the door* is the THEME and *that key* is the INSTRUMENT. This abstraction away from expression is particularly useful in question answering which is generally focussed on the factual meaning of text.

Grammar formalisms such as HPSG (Head-driven Phrase Structure Grammar: Pollard and Sag 1994) and LFG (Lexical Functional Grammar: Dalrymple 2001) combine syntactic and semantic information in their lexica, including information about the arguments (or roles) that each verb normally has. In order to make use of this extra information, semantic formalisms have been developed that describe the manner of semantic composition and hence give a formal meaning description of sentences derived from the semantics of the individual words. Two such formalisms are glue semantics (Dalrymple *et al.* 1993) and Minimal Recursion Semantics (MRS: Copestake *et al.* 2005). Although glue semantics is usually associated with LFG and MRS with HPSG, both can be used with any sufficiently descriptive grammar.

Glue semantics uses as its main element a bag of *glue premises* and associated predicates where a *glue premise* describe the way its associated predicate is related to other predicates (Dalrymple *et al.* 1993). The meaning of a phrase is then deduced by applying linear logic.

Minimal Recursion Semantics (MRS) in contrast uses a bag of *elementary predicates* which have features for each of their argument slots. These *elementary predicates* are combined into phrases by using composition rather than deduction. Examples of the lexical predicates of `kiss` from glue semantics and MRS are shown in Figure 4.3. The glue semantics representation specifies that `kiss` requires two arguments, an AGENT X and a THEME Y and when these can be filled (according to the mapping principles detailed in the grammar), a sentence with logical meaning kiss(X, Y) will

$$\textbf{kiss}{:}(\forall X, Y.agent(f_{1\sigma}, X) \otimes theme(f_{1\sigma}, Y) \multimap f_{4\sigma} = kiss(X, Y))$$

(a) Glue Semantics

$$\begin{bmatrix} \texttt{\_kiss\_v} \\ \text{LBL} \quad h1 \\ \text{ARG0} \quad e2 \\ \text{ARG1} \quad h3 \\ \text{ARG2} \quad h4 \end{bmatrix}$$

(b) Minimal Recursion Semantics

Figure 4.3: Representation of `kiss` in glue semantics and MRS

be formed. The MRS representation does not name the arguments, but the composition rules within the associated grammar specify that the KISSER is the predicate indexed by the ARG1 value (*h3*) and the KISSEE predicate is indexed by ARG2 (*h4*). ARG0 is the referential index that the `kiss` predicate is indexed by.

Given the aim of this research is to use deep linguistic knowledge wherever possible, obviously one of the deep semantic representations should be used for a similarity based answer selection module. Because of the additional robustness requirement, I have elected to use Robust Minimal Recursion Semantics (RMRS) which is a modification of MRS designed to provide a representation that can describe the output of deep and shallow processing in a compatible fashion (Copestake 2003). This allows applications to use the full strength of deep processing tools such as deep parsers where available, but also shallower tools such as part of speech taggers and dependency parsers to increase the robustness of the application. Similarity comparisons using RMRS allow for the most informative comparison possible with the available data. Chapter 5 provides a full explanation of RMRS, tools that can output RMRS and how RMRS can be used to merge the outputs of different linguistic tools. Chapter 6 will show how I use RMRS as a representation format for the similarity based answer selection task.

4.5 Summary

The task of the answer extraction part of a question answering system could be considered the same as detecting entailment, a task evaluated at recent PASCAL Recognizing Textual Entailment Challenges. In these challenges, as well as in QA evaluations, measuring similarity has been shown to be a productive strategy. Lexical similarity (similarity between two words) is a much researched area of lexical semantics, but measuring sentence similarity requires not only looking at individual words, but also the interword relationships. Similarity metrics can be classified according to how these interword relationships are modelled.

Lexical-based metrics are quick and easy to implement, but are not adequate for distinguishing between sentences like *I eat what I see* and *I see what I eat*. The basic bag-of-words model measures lexical similarity only, but it can be extended to match semantically similar words.

Syntax-based metrics provide more abstraction from word level comparisons. At a basic level this can just mean replacing words with a named entity tag, or a phrase with its phrase type (such as noun phrase), and then matching the abstract sentence patterns by 'shape'. A more informative syntactic comparison uses the dependencies between words, irrespective of word order, to compare between sentences.

More informative comparisons can be made if a parser can use semantic as well as syntactic information from the grammar. This can enable matching on semantic arguments such as THEME or AGENT, which relate to the truth conditions rather than the expression of the sentence.

Robust Minimal Recursion Semantics (RMRS) uses a representation that can combine information from deep and shallow processing tools, allowing the comparison of semantic information when it is available, but providing the robustness that is a feature of shallow processing methods. This representation forms the basis for the comparison algorithm that will be used for answer extraction in Chapter 6, and is introduced in detail in the following chapter along with some of the linguistic tools that can produce and manipulate RMRS.

Chapter 5

Robust Minimal Recursion Semantics

This research in question answering has two main goals—to maintain language independence wherever possible, and to make use of any available linguistic knowledge in a robust fashion. I chose to use Robust Minimal Recursion Semantics (RMRS) as a representation format because it contributes to both of these goals. In terms of language independence, RMRS provides an abstract representation of text, including semantic relations, which enables applications to work at this abstract level, in many cases ignoring language specific differences, such as word order. Also contributing to the language independence goal, there are now tools available for many languages which can output RMRS format, so any algorithm or application developed primarily in one language can be easily migrated to use data in any of the other languages with RMRS-producing resources. Some of the tools available for different languages are described in Section 5.2.

RMRS is a good option for the robust use of linguistic knowledge because of both its descriptive power, but also its capacity for principled underspecification. It is derived from Minimal Recursion Semantics (MRS: Copestake *et al.* 2005), and maintains the flat structure of MRS but takes atomisation a step further making not only predicates, but each predicate argument a separate atomic unit. A full description of the RMRS formalism is out of scope for this thesis (see Copestake 2004

for full details), but details relevant to this research will be explained in Section 5.1. RMRS can describe not only words, with their part of speech and sense, but also the syntactic and semantic relationships between the words and the intra-sentence scope of quantifiers such as *all* or *every*. However it is still possible to use the RMRS format if all this information isn't available. This means that any application designed to work with the RMRS format can operate over data from any depth of linguistic processing, delivering the robustness of shallow processing when the more complex and informative tools fail. While this feature of RMRS is generally employed by falling back to the most informative level of information available, in Section 5.3.1 I describe an algorithm I used to merge RMRS outputs from different tools.

5.1 Introduction to RMRS

Robust Minimal Recursion Semantics (RMRS) is a form of flat semantics which is designed to allow deep and shallow processing to use a compatible semantic representation, while being rich enough to support generalised quantifiers (Copestake 2003). A full RMRS representation consists of a bag of elementary predicates and their arguments (RELS), and a list of scoping constraints (HCONS).

Elementary predicates require a relation name, a relation type, a unique label and an ARG0 feature, which is known as the referential index. Predicates that share an ARG0 feature are said to be co-referentially indexed and hence have the same referent in the text. An example of an elementary predicate representing the word 発足 *hossoku* "inauguration" is shown in Figure 5.1(a). In this example the relation name is `_hossoku_s_1`, the label *h8* uniquely identifies the predicate and the referential index *e9* refers to the "inauguration" event. Predicates can have one of two types: REALPRED, which is used for predicates related directly to words in the text; and GPRED, which is used for predicates representing grammatical relations or unknown words. The type in Figure 5.1(a), indicated by the underscore (_) at the start of the relation name, is REALPRED. The relation name for a predicate of REALPRED type is derived from the related word in the text, while relation names for GPRED predicates come from a closed set of grammatical relations. Named entities are represented as a

(a) Fully specified predicate

(b) Underspecified predicate

Figure 5.1: Fully specified (left) and underspecified (right) elementary predicate for 発足 *hossoku* "inauguration"

predicate of type GPRED, with a relation name of `named_rel` and a CARG (Constant ARGument) feature that holds the actual name. The predicate in Figure 5.1(a) also has an ARG1 feature, which indexes another predicate that fills this argument slot.

The ARG features of a predicate are similar to the PropBank argument annotations (Kingsbury *et al.* 2002), denoting different argument slots, but without having a globally defined semantic role for each slot. Semantic roles are defined on a predicate by predicate basis, however in practice, ARG1 is often the AGENT and ARG2 the THEME, while ARG3 and ARG4 are less common, but are used for ditransitive verbs, such as for the GOAL of *give* or *put*.

A fully specified RMRS representation of the sentence

JR	が	発足	した	の	は	いつ	です	か
JR	*ga*	*hossoku*	*shita*	*no*	*wa*	*itsu*	*desu*	*ka*
JR	NOM	inauguration	did	NML	TOP	when	is	QM

"when was JR inaugurated?"

is given in Figure 5.2. This shows the bag of predicates as the RELS feature and scoping constraints, which describe the possible scope of each quantifier, in the HCONS feature.

Of the ten elementary predicates shown in Figure 5.2, only three are of type REALPRED, `_hossoku_s`, `_itsuka_n` and `_no_n`. These relate to the words 発足 *hossoku* "inauguration", いつ *itsu* "when" and の *no* "NML". The only other content word from the question is *JR*, which, as a named entity, is represented by the first predicate, which has a type of GPRED and a CARG feature holding the actual surface form.

As an example of the argument indexing within this figure, the ARG1 feature of

the `_hossoku_s` "inauguration" predicate, *x4*, is the referential index of the `named_rel` predicate representing JR, describing the fact that JR was the entity that inaugurated. The predicate `_no_n` also has an ARG1 feature because it is a nominalising noun that requires a verb phrase to nominalise. Since it requires a phrase, rather than an entity, its argument value links to the handle of a GPRED predicate, rather than a referential index.

The other six predicates are not related directly to words in the sentence and are all of type GPRED. The two `udef_rel` predicates are a feature specific to the Japanese grammar. Japanese does not, in general, use determiners and so, for example, 発足 *hossoku* "inauguration" is ambiguous between "an inauguration" and "the inauguration". In order to maintain compatibility with other languages that require determiners, a underspecified quantifier `udef_rel` is added for each noun, which could be specified to an definite or indefinite quantifier in translation. In this case they scope over the nouns `named_rel` and `_no_n`. The `preposition_m_rel` and `question_m_rel` are both message relations which are used to indicate a clause or phrase. In this case, `preposition_m_rel` scopes over the `_hossoku_s` predicate, which heads the phrase JRが発足した *JR ga hossoku shita* "JR was inaugurated", and is the phrase nominalised by the `_no_n` predicate. The `question_m_rel` shares a handle with the TOP feature of the RMRS, indicating it is the highest scoping predicate.

The last two GPRED predicates are indirectly related to words in the text. The `wh_rel` is related to the way the grammar handles *wh*-questions—in this case, rather than form a REALPRED predicate `_itsu_n` from the word いつ *itsu* "when", the grammar produces two predicates: `_itsuka_n` which relates to the word "sometimes" and the `wh_rel` to indicate a question word. Both of these predicates share the same referential index, *x18*, indicating that they refer to the same entity. The last predicate `cop_id_rel` relates to the word です *desu* "is" which is the copula verb relating the nominalised phrase JRが発足したの *JR ga hossoku shita no* "the inauguration of JR" with its complement, the "when" entity.

The HCONS values enumerate the scoping relations already described—the `udef_rel` predicates scoping over the nouns, the `preposition_m_rel` scoping over the `_hossoku_s` predicate, the `wh_rel` over the `_itsuka_n` and the `question_m_rel` over the `cop_id_rel`.

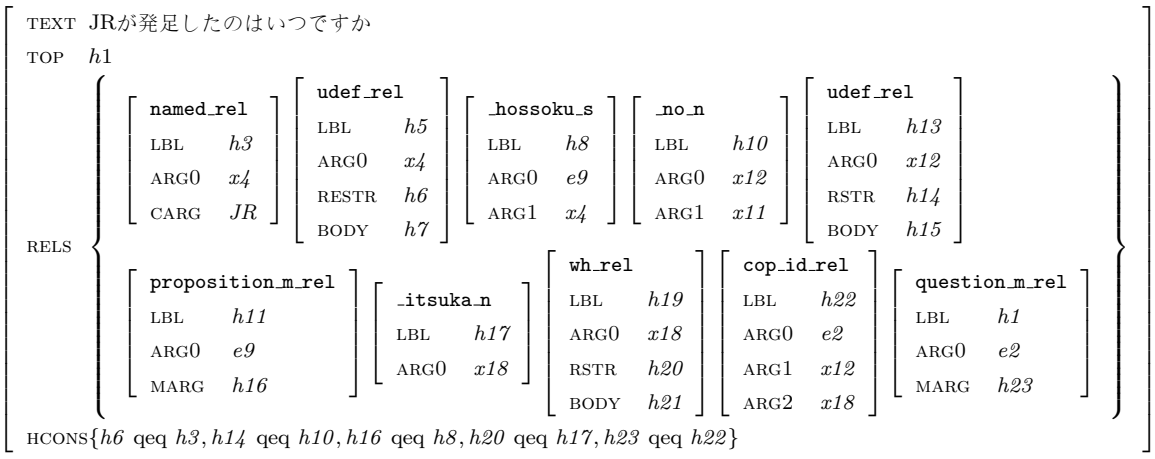


Figure 5.2: Full RMRS representation for JRが発足したのはいつですか *JR ga hossoku shita no wa itsu desu ka* "when was JR inaugurated?"

The suitability of RMRS as a framework for integrating results from deep and shallow parsing comes from the ability to underspecify a representation in a consistent and coherent fashion. The overall RMRS representation can be underspecified by omitting scoping constraints on quantifiers as well as omitting grammatical predicates such as message types and quantifiers. At a lower level, the individual predicates can also be underspecified in a principled way that maintains compatibility with a fully specified version. The structure of the relation names is one mechanism that enables this—the standard format is `lemma_pos_sense` where `pos` is drawn from a small set of parts of speech, and `sense` is a number or character string that indicates the sense defined within the relevant lexicon. Hence a predicate with name `_hossoku_s_1` relates to a word has been identified as the first sense of the **s**ahen (verbal noun) form of *hossoku*, and is compatible with, but more specific than, a predicate with a relation name of `_hossoku`. Argument features are the other way in which predicates can be underspecified. Omission of argument features is demonstrated in Figure 5.1, but it is also possible to use a less specific argument type, rather than omit it all together. For example, the ARGN feature can be used when the existence of a dependency relation is recognised by the parser, but the exact argument slot is not known. A possible

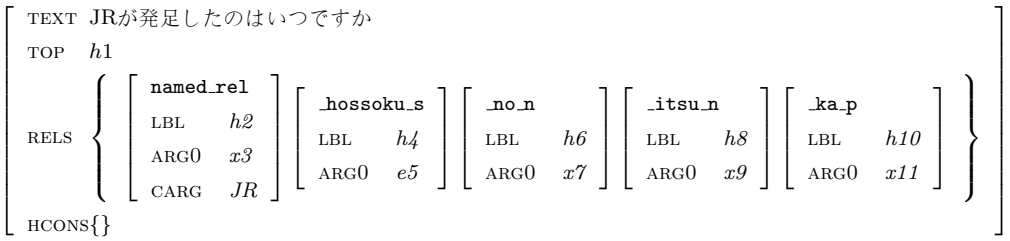


Figure 5.3: Underspecified RMRS representation of JRが発足したのはいつですか *JR ga hossoku shita no wa itsu desu ka* "when was JR inaugurated?"

underspecification of Figure 5.2 is shown in Figure 5.3. Compared to Figure 5.2, we see that there are no predicates of type GPRED except for the `named_rel`. The `_hossoku_s` and `_no_n` predicates are similar to those in the fully specified case, except that they are both missing the ARG1 feature. The `_itsu_n` and `_ka_p` predicates come directly from the words in the text, without the extra *wh*-question processing that produced the fully-specified version. This underspecified representation is what you might get from the output of a tokeniser and POS tagger.

5.2 Linguistic Processing Tools

The power of RMRS as a formalism for robust natural language processing comes from its descriptive power, as well its ability to be underspecified in a principled manner. That makes it important that language processing tools which produce RMRS output are available at all levels of linguistic processing. DELPH-IN[1] is a collective of research groups with an aim to develop systems that get at the meaning of text in a robust and efficient manner. They have elected to use (Robust) Minimal Recursion Semantics as a framework for developing their systems and through their research, grammars and tools have been created or extended to produce RMRS output at varying levels of processing. This includes HPSG grammars, statistical parsers, chunkers, part-of-speech taggers and named entity recognisers in a variety of

[1] `http://www.delph-in.net`

languages such as English, German, Japanese, Norwegian and Modern Greek. The PET system (Callmeier 2001) provides an efficient parser for use with HPSG grammars, and includes handling for unknown words, based on part of speech tags. The Deep Thought project which is part of DELPH-IN, led to a system called The Heart of Gold[2] (Schäfer 2006) which is middleware that integrates the output of tokenisers, named entity recognisers and parsers such as PET using RMRS and XML as a framework.

5.2.1 Japanese Tools

My research has been predominantly focussed on Japanese and I have been using tools written and extended by DELPH-IN members at all levels of processing. At the deepest level that is JACY, a broad-coverage linguistically precise grammar of Japanese based on the HPSG formalism (Siegel and Bender 2002). Figure 5.2 was output from PET, using the JACY grammar.

As an intermediate level tool when JACY failed to find a parse, I have used CaboCha (Kudo and Matsumoto 2002), a dependency parser of Japanese which was extended by Spreyer to produce RMRS output, in a similar manner to Spreyer and Frank (2005). The default output of CaboCha consists of unnamed dependency relations between words or chunks. Examples of CaboCha output are in Figure 5.4, showing the unlabelled dependencies between chunks. The CaboCha RMRS output can be configured to use only the CaboCha output, or enhance this with heuristics based on part of speech, or on verb valency information where this is available. For most of my experiments I used the output level enhanced by part of speech based heuristics, but not the verb valency information. The CaboCha version of Figure 5.2 is shown in Figure 5.5. In this version, the `udef_rel` predicates have been added, with the correct scoping, and the `_hossoku_s` predicate has its ARG1 pointing to the JR entity. However the text hasn't been recognised as a question, so the top scoping predicate is the `proposition_m_rel` which now scopes over the whole sentence, and there is no special processing of the *wh*-question word. This level of processing is be-

[2] `http://heartofgold.dfki.de`

JRが————D
発足したのは —D
いつですか

知床国立公園は————D
年間何人余りは —D |
訪れる —D
観光地であるか

Figure 5.4: Examples of the default output of the CaboCha dependency parser

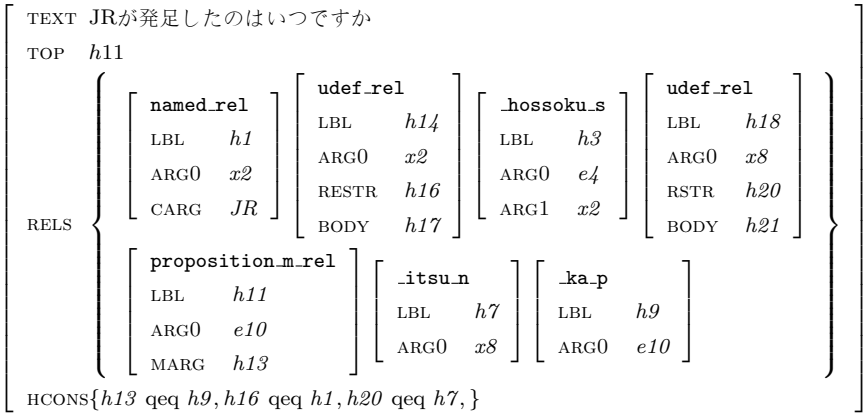
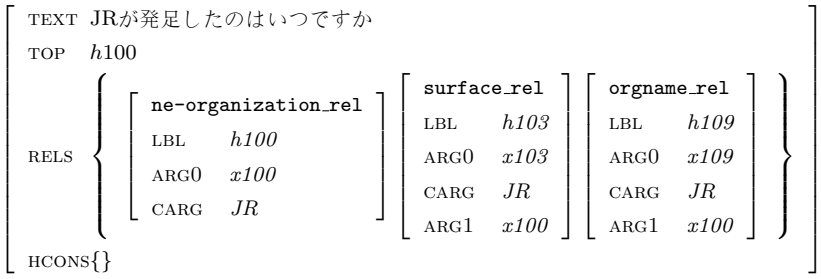


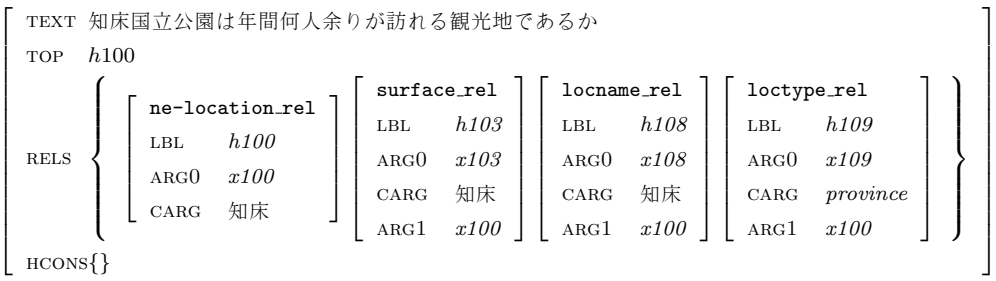
Figure 5.5: RMRS representation for JRが発足したのはいつですか *JR ga hossoku shita no wa itsu desu ka* "when was JR inaugurated?" from CaboCha

tween the fully specified JACY output and the underspecified version in Figure 5.3, containing more semantic information, but missing much of the syntactic detail that JACY supplies.

In addition to parse information, named entity information can also be output using RMRS format. I have used SProUT to detect named entities (Becker *et al.* 2002). SProUT produces not only a predicate with a named entity type relation name, but it can also produce other, related, predicates which provide more information about the named entity. The SProUT output for the sentence used in Figure 5.2 is shown in Figure 5.6(a), while the output from SProUT for a sentence containing the term 知床 *shiretoko* "Shiretoko", an area of Japan, is shown in Figure 5.6(b). They each have a predicate describing their named entity type (`ne-organization_rel` and `ne-location_rel`) and a `surface_rel`. In addition, there is a predicate for the entity



(a) Organization Entity



(b) Location Entity

Figure 5.6: RMRS output from SProUT

name, and in the case of the location, another predicate giving the location type. In these instances, the CARG value is the same for the first three predicates in each RMRS, but that does not have to be the case. It would be possible, depending on gazetteer used, for an abbreviation to be recognised in the surface form, and the official name given in the `orgname_rel` for example.

The part of speech information for both CaboCha and JACY comes from ChaSen (Matsumoto *et al.* 1999), a Japanese tokeniser and part of speech tagger. An extension to ChaSen for producing RMRS output was available, but was eventually unnecessary, since CaboCha had 100% coverage for the data I was using and the ChaSen information was always less specific than that from CaboCha.

5.3 RMRS Manipulation

In order to use RMRS data from a variety of sources, I have written some Perl modules[3] for manipulating RMRS structures. These modules allow me to read in RMRS data from a database or XML file, produce human readable representations and manipulate the RMRS structures in various ways. Some of these modules implement the RMRS comparison algorithm outlined in Chapter 6, as well as the various matching functions it uses.

5.3.1 Merging

The integration of RMRS output from different sources is one of the factors that make RMRS a useful framework for NLP applications. Often this integration really only consists of using the most informative level that produced a parse. This is still a useful model, since the application receives the same input format, regardless of source and can use RMRS structures from different sources in the same operation (for example comparison). However, it is possible to go further by merging RMRS structures from multiple sources into one RMRS structure. The Heart of Gold does this by adding all predicates from the different sources into one large RMRS. I have taken this a step further by matching elementary predicates according to their referent in the text (based on character offsets of the word within the text), and then selecting the most specific predicate in each class. That means that a predicate that has sense or argument information will replace one that doesn't, and a predicate with a more specific named entity relation type can replace a `named_rel` predicate. This differs from the unification process where the unified predicate would be the least specific of the two, or indeed the most specific predicate that subsumed both predicates. For my purposes, merging is more applicable than unification, since we assume that the more specific information is the more accurate and hence there is no need to allow for conflicting predicates. Extra predicates such as message types and those that add information about a named entity can be added during merging. The merge

[3]available on request: rdrid@dridan.com

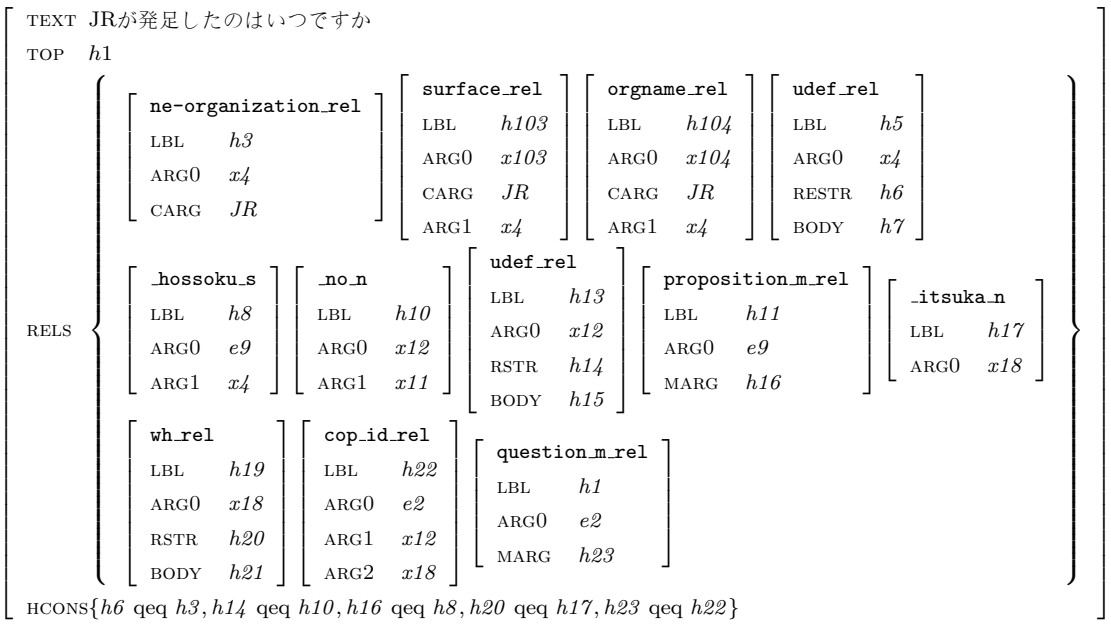


Figure 5.7: Output of merging RMRS from JACY and from SProUT

recreates all the argument index links to maintain consistency, and aims to produce a more informative RMRS structure than either of the component RMRS structures. The most useful application of this is merging two complementary sources, such as from JACY and SProUT, to take advantage of the more descriptive named entity predicates produced by SProUT. Figure 5.7 shows the results of merging the RMRS structures from Figure 5.2 and Figure 5.6(a). At this point, there is no benefit to merging output from CaboCha and JACY, since the JACY output should always be more specific, and hence taken as the preferred output. It might be possible to merge partial JACY parses with full CaboCha parses when JACY can produce parses for sentence fragments, but not the whole text. This would be another method of using the most specific linguistic information available, while still maintaining robustness, however this has not been implemented at this time.

5.4 Summary

Robust Minimal Recursion Semantics is a useful framework for linguistic processing applications because it allows language independent processing at the most informative linguistic level available. The main components of the RMRS structure are elementary predicates which each encapsulate information about a single word or term, and the relationships that word has to other elementary predicates in the text. Information about the word form is contained in the relation name of each predicate, or in the case of unknown words, in the CARG feature. Relationship information is encoded in the ARG features which relate to specific semantic roles, defined in the lexicon. An RMRS structure can be underspecified by leaving out unavailable information about word senses and predicate arguments.

Linguistic processing tools that produce RMRS are available in many languages. DELPH-IN researchers have created or extended RMRS producing tools of all levels of processing, from tokenisers to full semantic parsers. In Japanese there is ChaSen, a tokeniser and part of speech tagger; CaboCha, a dependency parser; SProUT, a named entity recogniser; and JACY, a Japanese HPSG grammar that is compatible with the PET parser.

I have created Perl modules for manipulating RMRS structures, providing a consistent structure for reading, creating and presenting RMRS structures. In addition, I have implemented a merging algorithm which can merge RMRS structures from different sources, to produce the most specific structure possible in a coherent fashion. I have also designed and implemented an RMRS comparison algorithm which is detailed in Chapter 6.

Chapter 6

Answer Extraction using Similarity

The aim of this phase of the research is to design a robust method of extracting the answer to a question from a relevant document. Robust Minimal Recursion Semantics (RMRS) is used as the sentence representation since it provides both robustness and an abstraction away from language specific surface forms, contributing towards the thesis goals of language independence and robust use of linguistic knowledge. Hartrumpf (2005) showed that the majority of questions at CLEF 2004 and 2005 could be answered by finding a sentence with a similar semantic representation to the question. To investigate the applicability of this claim to Japanese questions, the first step was to design and test a generic comparison algorithm to compare the similarity of two sentences represented by RMRS structures. This algorithm was then used first to select sentences that were semantically similar to questions in a Japanese question set, and then as part of a scoring system for ranking answer candidates.

The comparison algorithm is detailed in Section 6.1. Section 6.2 then describes how this algorithm was used in a paraphrase detection task, designed to evaluate how well the algorithm could detect semantic similarity. In Section 6.3 the comparison algorithm was used to select sentences that were semantically similar to questions in a QA task. Finally, in Section 6.4, the question–sentence similarity scores were used in calculating the scores of potential answer candidates in a QA answer extraction task. The next section describes the comparison algorithm in detail.

6.1 Comparison Algorithm

The comparison method is based on code included in the LKB (Copestake 2002) as used by Ritchie (2004). This LKB code was intended for comparing two RMRS representations of the same sentence from different sources. While this is still possible using the algorithm described here, the focus is on semantic similarity, allowing lexical substitution, and so more flexible predicate matching has been added. The comparison algorithm is language independent and can be used for any RMRS structure, but the flexibility of the matching depends on the quality and range of the lexical resources available for the language of the text to be compared. The system is designed to make it easy to add additional resources where they are available.

Briefly, the algorithm attempts to match predicates individually, examine their related grammatical predicates and then use any semantic role information, represented in ARGn features, to evaluate the quality of the matches. It then searches for the set of matches which generates the best average score and processes any unmatching predicates. Finally it produces a representation of all matched and unmatched predicates, as well as a numeric score. This score represents the calculated distance between the RMRS structures. The use of this score is application dependent—it may be used to rank sentences, or a threshold score indicating similarity could be learnt for a particular purpose.

The stages of the algorithm are explained in detail below, using Japanese sentences (21) and (22) as an example. These sentences are both definitions for the Japanese word ドライバー *doraibä* "driver", taken from two different Japanese dictionaries. The full RMRS structures for these sentences are shown in Figure 6.1.

(21)

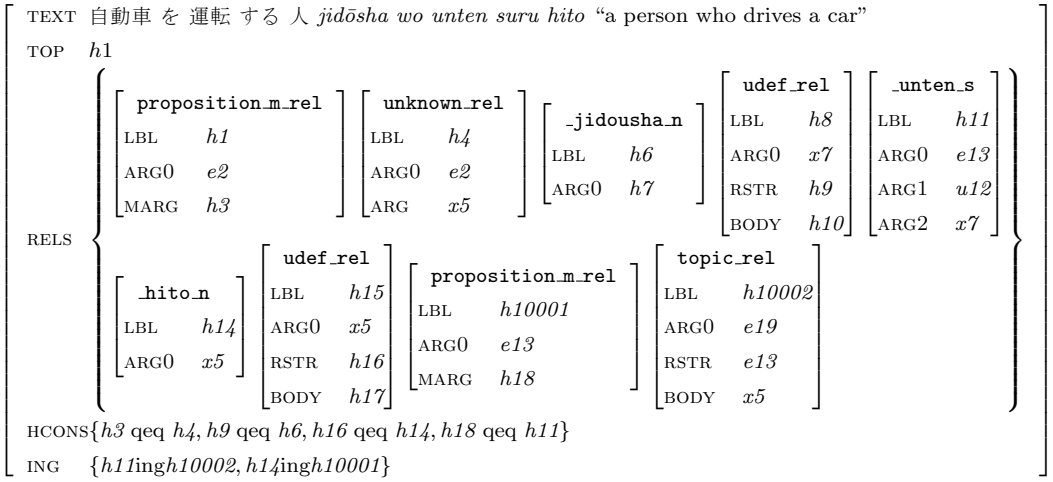
自動車	を	運転	する	人
jidōsha	*wo*	*unten*	*suru*	*hito*
car	ACC	drive	do	person

"a person who drives a car"

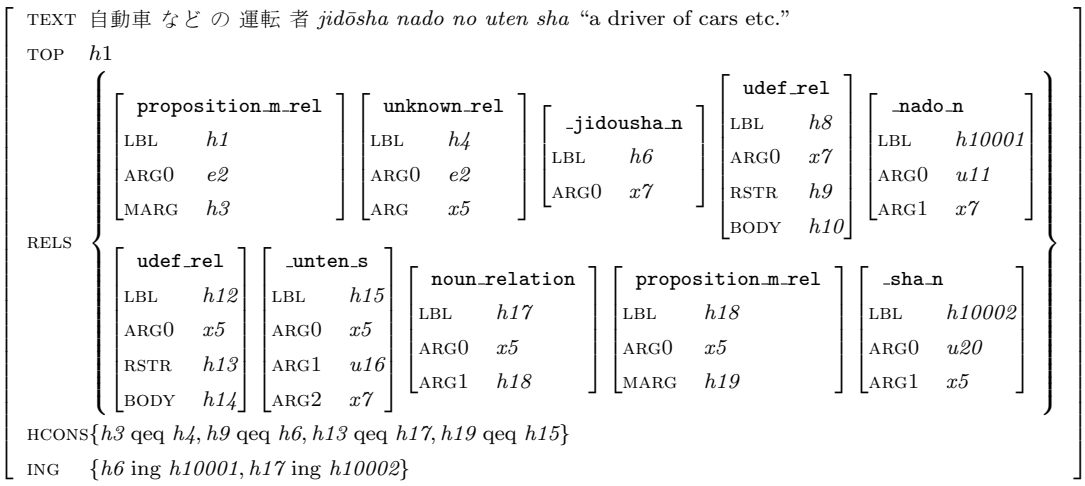
(22)

自動車	など	の	運転	者
jidōsha	*nado*	*no*	*unten*	*sha*
car	etc	GEN	drive	-er

"a driver of cars etc."



(a) 自動車を運転する人 *jidōsha wo unten suru hito* "a person who drives a car"



(b) 自動車 など の 運転 者 *jidōsha nado no unten sha* "a driver of cars etc."

Figure 6.1: RMRS structures for the example sentences

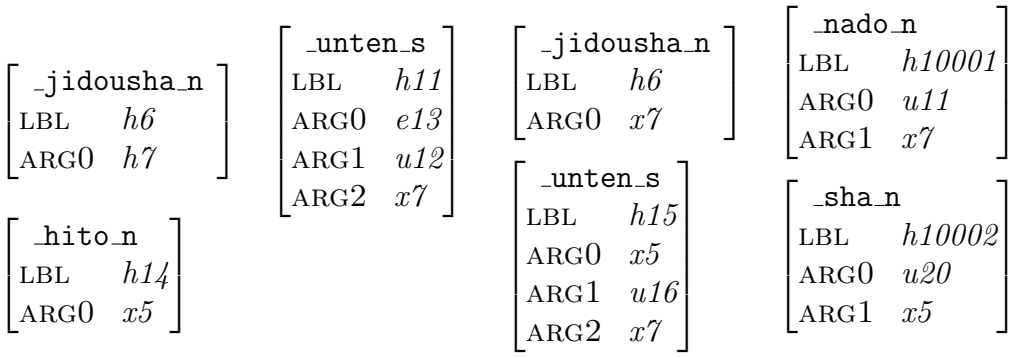
score for an EXACT match	0
score for a SYNONYM match	1
score for a HYPERNYM/HYPONYM match	2
score for a NO MATCH	3
weight for a content EP	1
weight for a non-content EP	0.1

Table 6.1: Default parameters for RMRS comparison

6.1.1 Finding predicate matches

The first stage examines all *content* elementary predicates (EPs). A content EP is defined here as an elementary predicate that directly relates to a word in the text. As mentioned in Section 5.1, this includes all EPs of type REALPRED, as well as the GPRED-type EPs with a relation name of `named_rel`. Relation name that starts with `generic_` (such as `generic_noun`, `generic_verb`) are also considered content predicates. Each content EP from the first sentence is compared with all the content EPs in the second sentence, and each EP pair will have a match type and score assigned. The match types can be either EXACT, SYNONYM, HYPERNYM, HYPONYM or NO MATCH. The scores represent the semantic distance between the EPs and, with the default parameters, can range from 0 to 3, with 0 indicating an EXACT match. Table 6.1 summarises the default scoring weighting parameters that will be used in the comparison experiments. While these parameters were selected somewhat arbitrarily, they appear to generalise well over different data sets. With a large enough amount of data, it would be possible to learn optimal values for these parameters for a particular application, but this was not done for these experiments.

The matching process is set up as a pipeline of match functions that operate on the EP pair and their associated surface forms. The comparison progresses down the pipeline until a match type other than NO MATCH has been found, or the end of the pipeline is reached. The default first match function compares relation names, and in the case of a `named_rel`, the CARG feature which holds the surface form of the unknown word. This function can only return EXACT or NO MATCH. An EXACT match will generally have a score of 0, however a slight penalty can be added for



(a) Content EPs from example 21

(b) Content EPs from example 22

Figure 6.2: An example of content EPs

orthographic variation in the surface form, or mismatch in the part of speech or sense of the relation.

Match functions that use additional lexical resources are added to the pipeline to enable a more informative comparison. A match function that uses a thesaurus would be able to return a match type of SYNONYM, while an ontology such as WordNet (Miller *et al.* 1990) or the Hinoki ontology (Bond *et al.* 2004a) could distinguish a HYPERNYM or HYPONYM as well. In the initial experiments, the only additional resource used was the Hinoki ontology, which was described in Chapter 3. Each match function has a precondition so that it is only applied in appropriate conditions. For example, some match functions may not be appropriate for two EPs with different part of speech.

Examining the two RMRS structures shown in Figure 6.1, the content EPs are those shown in Figure 6.2. Out of the 12 possible EP pairs, EXACT matches are found between the EPs with relation names of `_jidousha_n` and `_unten_s`, and a HYPERNYM relation is found, using the ontology, between the EPs with relation names of `_hito_n` ("person") and `_sha_n` ("-er"). All other pairs are of type NO MATCH. The match list at the end of this stage is shown in Figure 6.3, with matches marked with numbers 1-3 for later reference.

	RMRS 1	RMRS 2	Match Type	Score
1	[_jidousha_n LBL *h6* ARG0 *h7*]	[_jidousha_n LBL *h6* ARG0 *x7*]	EXACT	0
2	[_unten_s LBL *h11* ARG0 *e13* ARG1 *u12* ARG2 *x7*]	[_unten_s LBL *h15* ARG0 *x5* ARG1 *u16* ARG2 *x7*]	EXACT	0
3	[_hito_n LBL *h14* ARG0 *x5*]	[_sha_n LBL *h10002* ARG0 *u20* ARG1 *x5*]	HYPERNYM	2

Figure 6.3: Match list after the predicate matching stage

6.1.2 Co-referentially indexed predicates

The ARG0 processing finds all the EPs that share an ARG0 value with, and are hence co-referentially indexed with, the match EPs. These extra predicates are not content EPs, but can provide grammatical information about the way the content EPs are being used. If both EPs of a match have the same number and type of co-referentially indexed EPs, both EPs are being used in the same manner and no change is made to the score. However, if all these extra EPs do not match, indicating that the content EPs may differ in their grammatical form, a penalty is added to the match score. As the effect of these non-matching grammatical EPs was considered to be much less than the effect of non-matching content EPs, the maximum penalty they could attract was set to 0.1. If some of the grammatical EPs matched, the penalty added was proportional to the percentage that were unmatched.

In the case of match 2 (`_unten_s`, "drive") from Figure 6.3, the co-referentially indexed EPs are shown in Figure 6.4. As can be seen, the EP from RMRS 1 has only one of these non-content EPs (`proposition_m_rel`), which can be matched to one of the three EPs that are co-referentially indexed with the matching EP from RMRS 2. This signifies that in both examples, `_unten_s` is the head of a clause. The two non-content EPs from RMRS 2 which have no match in RMRS 1 specify that in

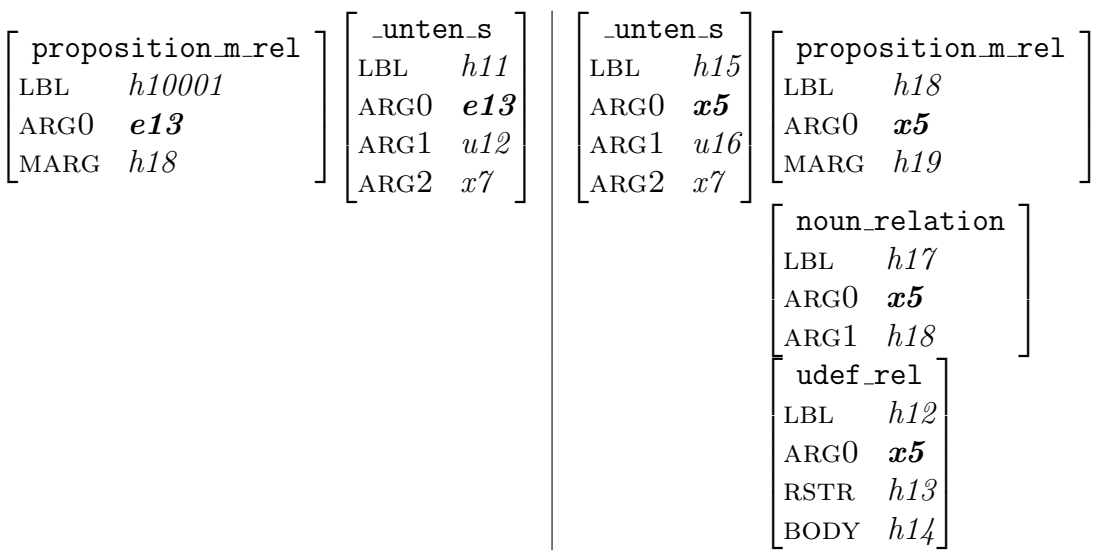


Figure 6.4: Co-referentially indexed EPs of match 2 EPs

RMRS 2, _unten_s is being used as part of an indefinite noun, rather than directly as a verb. The penalty added to the match score for these two unmatched EPs is $\frac{2}{1+3} \times 0.1 = 0.05$.

6.1.3 Semantic relations through argument features

The processing for the other ARGn features aims to make use of the semantic dependency relations that can be encoded in the RMRS structure. Each argument feature of the EPs in a match is examined and the indexed predicates are compared to see if a match exists between them. If both EPs of a match have an instantiated ARGn feature and there is no match between the EPs indexed by that feature, then a penalty score is added to the match. Pseudocode from this stage is shown in Algorithm 1. In the case of match 2 (_unten_s), both EPs have an instantiated ARG2 feature, as shown in Fig 6.5. In this case the indexed EPs are actually related by a match (ie match 1) and hence there is no penalty added to the match score.

A simple example showing the necessity of this processing step compares the following two sentences.

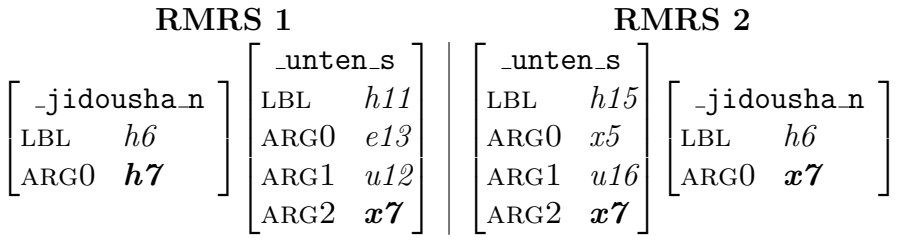


Figure 6.5: Indexed ARG2 EPs of match 2

Algorithm 1 Pseudocode for processing ARGn features

for all $m \in Matches$ **do**
  $penalty \leftarrow 0$
  for all $a \in \{1, 2, 3, 4\}$ **do**
    if $m.ep1.arg[a] \neq NULL$ **and** $m.ep2.arg[a] \neq NULL$ **then**
      if $isMatch(m.ep1.arg[a], m.ep2.arg[a]) = False$ **then**
        $penalty \leftarrow penalty + 1.5$
      end if
    end if
  end for
  $m.score \leftarrow m.score + penalty$
end for

RMRS 1	RMRS 2	Match Type	Score
_people_n LBL *h10* ARG0 *x7*	_people_n LBL *h16* ARG0 *x12*	EXACT	0
_eat_v LBL *h12* ARG0 *e2* ARG1 *x7* ARG2 *x13*	_eat_v LBL *h11* ARG0 *e2* ARG1 *x7* ARG2 *x12*	EXACT	0
_fish_n LBL *h17* ARG0 *x13*	_shark_n LBL *h10* ARG0 *x7*	HYPERNYM	2

Figure 6.6: Match list after predicate matching stage for examples (23) and (24)

(23) People eat fish.

(24) Sharks eat people.

After the predicate matching stage, matches between 23 and 24 are shown in Figure 6.6. While there is an EXACT match between the _eat_v EPs at this stage, penalties of 1.5 are added to the match score for mismatches in ARG1 indexed predicates (*x7*:_people_n and *x7*:_shark_n) and in ARG2 predicates (*x13*:_fish_n and *x12*:_people_n), bringing the match score of the _eat_v predicates to 3.

In the case that only one of the match EPs has a particular ARGn feature, no change is made to the match score. This allows for the comparison between a fully specified RMRS and an underspecified RMRS, without regarding the lack of argument information on one side as adding to the distance between two sentences.

After all the argument features have been processed, the number and type of matches in the match list will not have changed, but the scores given to each match may have been increased. The match list from our example is shown in Fig 6.7, where small increases to scores in match 2 and 3 are due to non-matching co-referentially indexed EPs.

	RMRS 1	RMRS 2	Match Type	Score
1	_jidousha_n LBL *h6* ARG0 *h7*	_jidousha_n LBL *h6* ARG0 *x7*	EXACT	0
2	_unten_s LBL *h11* ARG0 *e13* ARG1 *u12* ARG2 *x7*	_unten_s LBL *h15* ARG0 *x5* ARG1 *u16* ARG2 *x7*	EXACT	0.05
3	_hito_n LBL *h14* ARG0 *x5*	_sha_n LBL *h10002* ARG0 *u20* ARG1 *x5*	HYPERNYM	2.1

Figure 6.7: Match list after all argument processing is finished

6.1.4 Deriving the best match set

After the argument processing stage, the match list is complete and the algorithm attempts to determine the best set of matches, where the best set is the lowest scoring set, ie least distance, with each predicate included no more than once. In our example, no EP features in more than one match and so this stage is simple. In a more complex example, one EP may be matched with multiple EPs in the alternate sentence and enumerating all possible match combinations can be computationally intensive. A variation of the branch-and-bound approach is used to speed up set computation. A tree is constructed with each level representing one match from the match list. At each level, the branches correspond to adding or not adding the match to the match set. An "add" branch is pruned when adding a match is not possible, because the EPs featured in the match have already been included via an earlier match. A "don't add" branch is pruned when the current cost of the path means that regardless of future decisions, this branch could never lead to the lowest cost path. The cost of a path is calculated by starting with the maximum cost possible if no matches are added, and then, with each match added, subtracting the cost of those unmatched predicates and adding the cost of the match. Match scores can range from 0 to 3, and so for example, if the cost of a set is 17.1 and there are only two more possible

content EP matches left to add, the minimum cost this set could reach would be $17.1 - (2 \times 3) = 11.1$. (If the remaining matches are all non-content EPs, they have a maximum score of 0.3, rather than 3.) If another set at the same level already has a score lower than 11.1, the branch containing the first set will be pruned. It is possible for more than one path (or set) to have the same cost at the end of this process. In this case, the first set found is selected as the best.

6.1.5 Processing unmatched predicates

Once an optimal match set has been determined, the unmatched non-content EPs are processed to see if matches can be also found between them. Groups of non-content co-referentially indexed EPs at this stage may also be matched, if there is a corresponding group of EPs in the second RMRS. In our example, both RMRS structures have such a group consisting of EPs of type `proposition_m_rel` and `unknown_rel`, and so this EXACT match is added to the match list. The final match list for this example is shown in Figure 6.8, with each match represented only by representative EPs (that is, not all non-content predicates for each match are shown).

6.1.6 Output

The final distance score between the RMRS structures is calculated by normalising the total match score using the 'length', defined as the weighted sum of EPs involved in the two structures, where non-content EPs have 10% of the weight of content EPs. In our example, this weighted sum, for 7 content EPs and 12 non-content EPs, is $7 \times 1 + 12 \times 0.1 = 8.2$. If there were no matches, every predicate would contribute 3 times its weight to the match score, and then the total distance would be $\frac{3\times7+3\times1.2}{8.2} = 3$. In this case, where all but one content EP and one non-content EP were matched, the total match score is the sum of the matches $(0 + 0.05 + 2.1 + 0)$ and the unmatched EPs $(3 + 0.3)$, and so the distance between sentences is calculated to be $\frac{2.15+3.3}{8.2} = 0.66$.

The output returned by this comparison is this numeric distance score, as well as

Algorithm 2 Pseudocode for deriving the best match set

```
for all m ∈ Matches do
    if Sets = φ then
        newset ← 0
        concat(Sets, newset)
        newset ← 1
        concat(Sets, newset)
        min_cost ← cost(newset)
    else
        for all s ∈ Sets do
            if m.ep1 ∉ s and m.ep2 ∉ s then
                newset ← s
                concat(newset, 1)
                concat(Sets, newset)
                if cost(newset) < min_cost then
                    min_cost ← cost(s)
                else
                    delete(s)
                end if
            end if
            if cost(s) − cost_left(m) < min_cost then
                newset ← s
                concat(newset, 0)
                concat(Sets, newset)
            else
                delete(s)
            end if
        end for
    end if
end for
```

	RMRS 1	RMRS 2	Match Type	Score
1	[_jidousha_n LBL *h6* ARG0 *h7*]	[_jidousha_n LBL *h6* ARG0 *x7*]	EXACT	0
2	[_unten_s LBL *h11* ARG0 *e13* ARG1 *u12* ARG2 *x7*]	[_unten_s LBL *h15* ARG0 *x5* ARG1 *u16* ARG2 *x7*]	EXACT	0.05
3	[_hito_n LBL *h14* ARG0 *x5*]	[_sha_n LBL *h10002* ARG0 *u20* ARG1 *x5*]	HYPERNYM	2.1
4	[proposition_m_rel LBL *h10001* ARG0 *e13* MARG *h18*]	[proposition_m_rel LBL *h18* ARG0 *x5* MARG *h19*]	EXACT	0

Figure 6.8: Final match list

a representation of the final match list and the unmatched predicates. While in many cases only the distance will be used, the extra information about how the sentences were matched is useful for those applications where further processing of the results is required, such as judging entailment, or extracting an answer to a question. The final output for our example comparison is shown in Figure 6.9.

This section described a comparison algorithm which is quite generic, and can be easily tuned to a specific language or application by adding appropriate match functions to the match pipeline. Every other aspect of the algorithm is language independent. The following section describes how this algorithm was used to detect paraphrases.

6.2 Paraphrase detection

Paraphrase detection was selected as the task to evaluate the comparison method as it is a task that could directly test the validity of the semantic comparison. Paraphrase detection has been used to increase the recall of various information access

Distance Score: 0.66

Matches

RMRS 1	Match Type	RMRS 2
udef_rel: LBL *h8*, ARG0 *x7*, RSTR *h9*, BODY *h10*	EXACT 0	udef_rel: LBL *h8*, ARG0 *x7*, RSTR *h9*, BODY *h10*
_jidousha_n: LBL *h6*, ARG0 *h7*		_jidousha_n: LBL *h6*, ARG0 *x7*
proposition_m_rel: LBL *h10001*, ARG0 ***e13***, MARG *h18*	EXACT 0.05	noun_relation: LBL *h17*, ARG0 ***x5***, ARG1 *h18*
_unten_s: LBL *h11*, ARG0 *e13*, ARG1 *u12*, ARG2 *x7*		_unten_s: LBL *h15*, ARG0 *x5*, ARG1 *u16*, ARG2 *x7*
		proposition_m_rel: LBL *h18*, ARG0 ***x5***, MARG *h19*
		udef_rel: LBL *h12*, ARG0 ***x5***, RSTR *h13*, BODY *h14*
_hito_n: LBL *h14*, ARG0 *x5*	HYPERNYM 2.1	_sha_n: LBL *h10002*, ARG0 *u20*, ARG1 *x5*
		udef_rel: LBL *h15*, ARG0 *x5*, RSTR *h16*, BODY *h17*
proposition_m_rel: LBL *h10001*, ARG0 ***e13***, MARG *h18*	EXACT 0	unknown_rel: LBL *h4*, ARG0 *e2*, ARG *x5*
unknown_rel: LBL *h4*, ARG0 *e2*, ARG *x5*		proposition_m_rel: LBL *h18*, ARG0 ***x5***, MARG *h19*

Unmatched Elementary Predicates

RMRS 1	RMRS 2
topic_rel: LBL *h10002*, ARG0 *e19*, RSTR *e13*, BODY *x5*	_nado_n: LBL *h10001*, ARG0 *u11*, ARG1 *x7*
0.3	3

Figure 6.9: Final match result

applications such as question answering, information retrieval, information extraction and machine translation (Rinaldi *et al.* 2003, Shinyama and Sekine 2003, Callison-Burch *et al.* 2006). A series of workshops based on paraphrasing explored paraphrase detection, acquisition and generation (Sato and Nakagawa 2001, Inui and Hermjakob 2003, Dras and Yamamoto 2005) and the Recognising Textual Entailment Challenges (Dagan *et al.* 2005, Bar-Haim *et al.* 2006) used paraphrase data as a source of text/hypothesis pairs. Much of the work on paraphrasing attempts to acquire paraphrase patterns by comparing texts describing the same event (Barzilay and Lee 2003, Shinyama *et al.* 2002), but for this research, the goal is to determine whether two texts do represent the same event or idea, irrespective of the expression.

The Japanese annotated paraphrase data used for this task consisted of two separate Japanese dictionaries that had been aligned at a sense level as part of NTT's Hinoki project (Bond *et al.* 2004a). The hypothesis in this case was that the definitions of senses that had been judged to be equivalent could be considered paraphrases of each other. The dictionary data provided a non-trivial task, since different senses of the same word can often be defined using many similar words.

6.2.1 Paraphrase Data

The first dictionary used was the Lexeed Semantic Database of Japanese, a machine readable dictionary that covers the most familiar words in Japanese based on a series of psycholinguistic tests (Kasahara *et al.* 2004). Lexeed has 28,000 words divided into 46,000 senses, with each definition written using only the words defined in the dictionary. The second dictionary was the Iwanami Kokugo Jiten (Iwanami: Nishio *et al.* 1994), the Japanese dictionary used in the SENSEVAL-2 Japanese lexical task (Shirai 2002). The senses were aligned by showing annotators pairs of sense definitions, one from each dictionary. Each sense of one head word was paired with every sense of that same headword in the other dictionary. Three annotators judged each definition pair to be non-matching, equal, equivalent or subsuming, where subsuming was used if the definitions were equivalent in part, but one definition was broader. For example, the following two examples are translations of the definitions

of アンコール *ankōru* "encore" from Lexeed (25) and Iwanami (26).

(25) The demand for an additional performance or appearance, expressed by the customers yelling and clapping etc. for the performer after the scheduled performance at a concert etc.

(26) The desire for a repeat performance, expressed by yelling and clapping for the performer. Also the answering performance.

In this instance, all three annotators judged the Iwanami definition to subsume the Lexeed definition because it covered not only the call for performance, but also the actual repeat performance. Without *Also the answering performance.*, these definitions would have be judged as equivalent, since they use slightly different wording to express the same idea (and are hence paraphrases of each other).

6.2.2 Method

Ten sets of 5,845 definition pairs were extracted, using only those pairs that had been judged as equal, equivalent or non-matching, and excluding subsuming pairs. Sample sets were selected, rather than using the entire data set, as non-matching pairs made up a very large percentage of the full set, leading initial experiments to perform best when every sentence pair was classified as non-matching. Hence every equal and equivalent pair was selected, and an equal number of non-matching pairs were taken in the order they had been annotated in. Subsuming pairs were excluded to focus the task towards semantic similarity, rather than potential meaning overlap. I ran a binary classification task that classified each pair as either similar or non matching. Half of each set were non-matching pairs, so a majority class classifier baseline should achieve an average accuracy of 50%. The RMRS comparison method, as well as a bag-of-words comparison, was run on the first sentence of each definition in the pair, and the distance score recorded. A threshold score for deciding similarity was determined using 10-fold cross validation over each of the ten sets, which was between 2.5 and 2.7 for the RMRS comparison and 0.3-0.4 for the bag-of-words experiments.

6.2.3 Results

Table 6.2 shows the results of this evaluation. For the first experiment, a simple bag-of-words model was used to calculate word overlap. Three different RMRS comparison experiments were then run. The first used only JACY output and failed if a JACY parse was not found for both definition sentences. The second RMRS comparison used only CaboCha output, providing a lower level of linguistic information, in return for 100% recall. The last RMRS comparison used the most informative level of parsing available for each definition sentence, starting with JACY and falling back to shallow parsing methods (CaboCha) when JACY failed to find a parse. This meant that quite often a JACY analysis of one sentence was compared with a CaboCha analysis of the other. One of the benefits of using RMRS is that this comparison of outputs from different sources is completely transparent, and so there is no need to use a lower level parse for both sentences in the case where the highest level is only available on one side.

Results are given in terms of precision, recall and F-measure (with a β value of 1). Examining recall shows the major disadvantage of using deep linguistic methods on their own: it can be seen here that the RMRS from JACY allowed a significantly more accurate comparison than bag-of-words, but at the expense of only returning a score for around half of the definition pairs. What is interesting to note is that CaboCha, despite providing less information than JACY, still had significantly higher precision than bag-of-words and could give 100% recall. The experiment allowing fallback showed both that different levels of processing can be combined, and that doing so can combine high precision and high recall, hence leading to a higher F-measure.

Examining the breakdown of the results (shown in Table 6.3), both bag-of-words and RMRS had a higher accuracy when looking at only the non-matching definition pairs. Intuitively this makes sense, since it is the easier decision to make and, unlike some entailment tasks, the data wasn't deliberately selected to contain negative examples with high lexical overlap. Comparing the precision over matching and non-matching definitions for the bag-of-words and JACY only comparisons, JACY achieves a larger improvement over the matching definitions. One possible reason for

	Precision	**Recall**	**F-measure**
bag-of-words	0.740	1.000	0.850
JACY only	**0.804**	0.523	0.634
CaboCha only	0.799	1.000	0.888
JACY with fallback	0.801	1.000	**0.890**

Table 6.2: Results from paraphrase detection, using the bag-of-words comparison and the RMRS comparisons. The three RMRS results are for RMRS representations from JACY only, RMRS from CaboCha only, and then comparing RMRS from the most informative level available (JACY with feedback). The best precision and F-measure are highlighted in bold.

	Precision over Matches	**Precision over Non-Matches**
bag-of-words	0.699	0.780
JACY only	0.785	0.822
CaboCha only	0.770	0.827
JACY with fallback	0.755	0.830

Table 6.3: Paraphrase results by class

this is that JACY can detect matches that involve synonym substitution because the ontology is used.

To test the effect of the ontology lookup on the RMRS comparison, the three RMRS experiments were run again, but without the ontology matching step. The results (shown in Table 6.4) show a small performance drop indicating that the ontology is providing useful information. However the precision for all three experiments is still significantly higher than that achieved using the bag-of-words model (using $\chi^2, p \leq 0.05$), showing that the abstraction away from surface level, and the semantic relation information allow for a more informative comparison. Looking at the results split into matching and non-matching classes, we see that the performance drop caused by not using the ontology comes from a lower accuracy over the matching pairs, while the accuracy over the non-matching pairs actually gets higher. This shows one of the tradeoffs that is made by allowing synonym and hypernym matches—while this helps to find equivalences that would not otherwise be detected, it also licenses matches that should not be made.

	Precision	**Recall**	**F-measure**
JACY only	**0.797**	0.418	0.548
CaboCha only	0.794	1.0000	0.885
JACY with fallback	0.795	1.0000	**0.886**

Table 6.4: Results from paraphrase detection without an ontology. The three results are for RMRS representations from JACY only, RMRS from CaboCha only, and then comparing RMRS from the most informative level available (JACY with feedback). The best precision and F-measure are highlighted in bold.

	Precision over Matches	**Precision over Non-Matches**
JACY only	0.731	0.862
CaboCha only	0.690	0.904
JACY with fallback	0.682	0.868

Table 6.5: Paraphrase results by class, without ontology

The paraphrase experiments proved that the RMRS comparison algorithm was a valid alternative to a bag-of-words based comparison, and could transparently compare sentence representations from different levels of linguistic processing. Using an ontology did provide some benefit, but even without the ontology, the abstraction provided by RMRS allowed a more accurate comparison. The next section describes an experiment using this comparison algorithm for finding sentences which are semantically similar to questions in a Japanese question set.

6.3 Answer Sentence Selection

After confirming that the RMRS comparison gave an improvement over bag-of-words in determining semantic similarity, the next step was to investigate the findings of Hartrumpf (2005) that there are often minimal differences in the semantic representations of a question and a sentence that contains the answer to that question, by using the similarity measure in answer sentence selection for question answering.

6.3.1 Question Answering Data

I used the data set of 2000 Japanese questions from Sekine as described in Chapter 3, where each question was annotated with its answer and a document ID for the document the answer was found in. These document IDs refer to documents from the Mainichi Newspaper corpus from 1995,[1] which has been used as part of the document collection for NTCIR's Question Answering Challenge. Documents range from 3 to 83 sentences in length.

A second smaller question set was also used to enable later comparisons with other results. This is the 100 question set from QAC-1 that was used as held-out data in Chapter 3. This set is also annotated with answers and document IDs, although in this case the document IDs refer to the 1998 and 1999 Mainichi Newspaper corpora.

6.3.2 Method

Each question was compared, using both bag-of-words and the RMRS comparison methods, with every sentence in the referenced answer document. The sentences were ranked by the similarity score calculated, and the most similar sentence for each question was returned. The sentence selection was judged correct if the appropriate answer could be found in the returned sentence. This method of evaluation means that there can be more than one correct sentence for each question, if the answer is mentioned more than once in the document.

6.3.3 Results

The results for the Sekine data set are shown in the first column of Table 6.6.[2] As this table shows, the results here are significantly lower than the accuracy obtained in the paraphrase detection task and the improvement achieved by using RMRS is not as large in the 2000 question set. Examining the data, it can be seen that the

[1] `http://www.nichigai.co.jp/sales/mainichi/mainichi-data.html`

[2] Results for QAC-1 test set are shown to only two significant figures, since the data set was only 100 questions.

	Sekine question set (2000 questions)	**QAC-1 question set (100 questions)**
bag-of-words	0.481	0.35
RMRS	0.516	0.50
RMRS with SProUT	0.587	0.51
RMRS with pih	0.598	0.51

Table 6.6: Results from the answer selection task

language level is much more difficult in this task than in the previous task. Questions and sentences were on average three times longer, with ten elementary predicates per sentence in the dictionary definitions, compared with an average of 34 in the questions and newspaper sentences. The vocabulary was also more varied, with more formal, less familiar words so the parser often failed due to unknown words, and very few word pairs were found in the ontology. The number of question–sentence pairs where a JACY parse was available on both sides was insignificant, and hence CaboCha was used for every comparison. When these comparison tasks were evaluated over the 100 question set that was used in QAC-1, the accuracy was even lower, but interestingly, the bag-of-words comparison did significantly worse. It appears, when comparing the two question sets, that the manually constructed 2000 question set has a very high word overlap with the answer bearing sentences.

Many of the unknown words were named entities, such as names of people, companies and places and since these were a large source of match failure, it was decided to add named entity processing into the RMRS comparison process. First, SProUT, described in Chapter 3, was used to extract named entities in RMRS format. Since these provide supplemental, rather than replacement, information for the parsers, it was not possible to fall back to the SProUT output. Instead, the merging algorithm described in Section 5.3.1 was used to merge the SProUT RMRS output with that from CaboCha. These supplemented RMRSs were then compared as before. They gave a significant improvement over both the bag-of-words and the original RMRS comparison as shown in Table 6.6.

Looking forward to the eventual aim of the comparisons, a different named entity

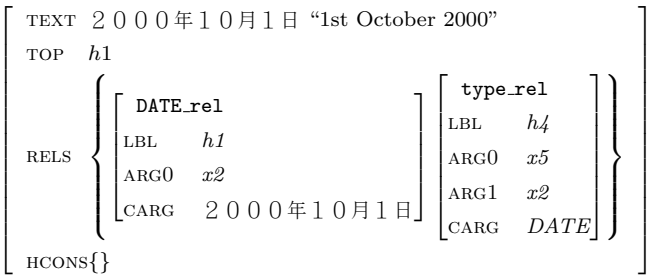
tagger was found that used the named entity tags found in Sekine's Extended Named Entity Hierarchy, since these were the ones used in both the gold standard data and in earlier question classification experiments. This named entity tagger, called pih, was written by Professor Sekine and has not yet been publically released. In its original form, the input expected is the default output of Juman, a Japanese tokeniser and part of speech tagger. There are 3 implemented output formats, with the most useful for the purposes of this research using IOB style tags to tag each separate word with its named entity type. I modified the original program to accept other Juman output formats, specifically those that include the byte count of each word within the text, so I could output the byte counts and then merge the output with the CaboCha output, which indexes each word by character count. The modified output format is shown in Figure 6.10 and displays the byte offsets of the start and end of the word, the surface form, the phonetic reading, and the base form of each word and its named entity tag.

I then wrote a filter to transform the pih output into RMRS format. For each named entity, an RMRS containing 2 elementary predicates was constructed. The first predicate is similar to, but more specific than a `named_rel` predicate, with the relation name being made from the named entity type (eg, `DATE_rel`) and the CARG value from the base form of the complete named entity. The entity may consist of more than one word, and hence in merging may replace several consecutive `named_rel` predicates. The second predicate in the RMRS has a relation name of `type_rel` and the CARG value is the named entity type (eg, *DATE*). The ARG1 feature of this second predicate is the index (ARG0) of the first predicate. This arrangement is compatible with the way SProUT produces named entity RMRS output. The pih output in Figure 6.10 contains two named entities, one DATE and one N_ORGANIZATION. The RMRS output produced from this is shown in Figure 6.11. In the same way as above, the RMRS output from pih was merged with the CaboCha output and then the merged RMRSs were compared. The accuracy over the 2000 question set was slightly better than that from SProUT, and given the benefits of using a compatible named entity type set, pih was used as the named entity tagger for all further experiments.

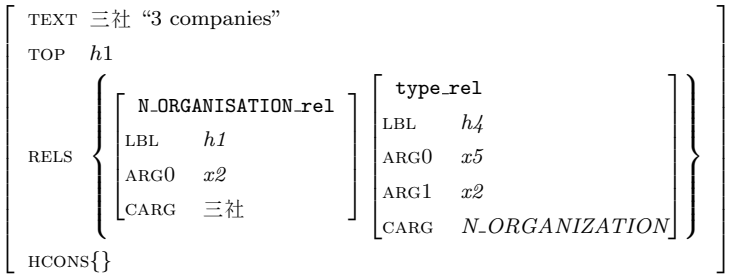
The comparison algorithm used for paraphrase detection generalises quite well for discovering sentences which are similar to questions when the sentence representation

0	8	２０００	にぜろぜろぜろ	２０００	名詞	B-DATE
8	10	年	ねん	年	接尾辞	I-DATE
10	14	１０	いちぜろ	１０	名詞	I-DATE
14	16	月	がつ	月	接尾辞	I-DATE
16	18	１	いち	１	名詞	I-DATE
18	20	日	にち	日	接尾辞	I-DATE
20	22	に	に	に	助詞	OTHER
22	26	合併	がっぺい	合併	名詞	OTHER
26	30	する	する	する	動詞	OTHER
30	34	こと	こと	こと	名詞	OTHER
34	36	が	が	が	助詞	OTHER
36	44	決まった	きまった	決まる	動詞	OTHER
44	48	通信	つうしん	通信	名詞	OTHER
48	50	三	さん	三	名詞	B-N_ORGANIZATION
50	52	社	しゃ	社	接尾辞	I-N_ORGANIZATION
52	54	は	は	は	助詞	OTHER
54	58	どこ	どこ	どこ	指示詞	OTHER
58	62	です	です	だ	判定詞	OTHER
62	64	か	か	か	助詞	OTHER
64	66	。	。	。	特殊	OTHER
EOS						

Figure 6.10: Modified pih output



(a)



(b)

Figure 6.11: RMRS created from the pih output shown in Figure 6.10

is enriched with named entity information. This was necessary due to the high density of named entities within newspaper data, and easily accomplished via RMRS merging, with no alteration necessary to the comparison algorithm. The next section explains how the similarity score from a question–sentence comparison can be used as part of an answer extraction task.

6.4 Answer Extraction

The next set of experiments again compares questions with sentences, but rather than return the most similar sentence, the similarity score of each sentence is used as part of the score for potential answer candidates from that sentence in an answer extraction task. In order to use the comparison algorithm for answer extraction I added a new matching function to the comparison pipeline that matched question words to named entities, taking advantage of the expected answer type (EAT) of the question if it was known. This match function ran if one of the predicates to be matched referred to a question word and returned a match score of 1 if the other predicate was a `type_rel` with a CARG that matched the EAT. As a more lenient match, a question word could be matched with any `type_rel` with a match score of 2.

Another matching function was added that allowed 'soft' matching between named entity types, that is named entity types match if they are in the same branch of the named entity hierarchy or shared a common parent. This is the soft decision strategy shown in Figure 3.1(b). Types which were not an EXACT match, but matched according to this strategy were given a match score of 2.

6.4.1 Method

To discover the most likely answer to each question in a given document, each entity within the document that was extracted by pih was given a score, and the entity with the lowest score for one question was returned as the answer. The score was made up three components. The first component was a score for the match

between the entity type and the gold standard expected answer type. This score was 0 for an EXACT match, 1 for a soft match using the same soft decision strategy as above, and 2 otherwise. The second scoring component was a score given if the entity was matched to a question word during comparison. This component was equal to twice the score of the match. If the entity was not matched to the question word, this component had a weight of 6—twice the weight of an unmatched predicate. The last component of the entity score was based on the similarity score between the question and the sentence the entity is in. Since this component gave more information than just the type, it was given a weight of 3. Therefore for a question with an EAT of DATE, a DATE entity that matched to the question word (*itsu* "when") with a score of 1.1 in a sentence that had a similarity score of 1.446 will have a final score of $0+2\times1.1+3\times1.446 = 6.540$. If the entity was in a less similar sentence, with a score of 2.031, then the entity score would be $0 + 2 \times 1.1 + 3 \times 2.031 = 8.292$. For another question with an EAT of PERSON, the most similar sentence (with a similarity score of 1.265) had a GAMES entity that was matched to the question word *dare* "who" during comparison. This entity had a score of $2+2\times2.1+3\times1.265 = 9.997$. Another less similar sentence matched a PERSON entity to the question word and thus had a final score of $0 + 2 \times 1.1 + 3 \times 2.112 = 8.536$, which then gave the correct answer (the PERSON entity) despite not being in the most similar sentence.

6.4.2 Results

The results of answer extraction for both question sets are shown in Table 6.7. The MRR score is given so comparisons can be made to the original results from QAC-1. This MRR would have ranked 6th out of 15 systems at QAC-1. Of course, since oracle IR was used, this may not be an accurate comparison, but as there were passage retrieval modules that achieved almost 100% accuracy, it is not unrealistic. Error analysis of the small question set showed that an extra 8 answers that were marked as incorrect by the automatic script were in fact valid alternate answers, bringing the accuracy on the QAC-1 set up to 0.36 and the MRR to 0.4108 (which would have been fourth at QAC-1). Of the remaining errors, four would be marked as

	Accuracy	**MRR**
Sekine question set (2000)	0.345	0.396
QAC-1 question set (100)	0.28	0.34

Table 6.7: Question answering results using similarity

INEXACT, and 36 were not detected as entities by the named entity tagger. This means that 24 entities were detected, but were not returned as the first answer. Of these, 16 were in the top five answers, showing that the scoring method works reasonably well, with 52 out of 60 entities detected were returned high in the rankings.

6.4.3 Effect of different named entity taxonomies

In Chapter 3, named entity taxonomies of different sizes and shapes were used in question classification. In order to see the effect of these different taxonomies on the end result of question answering, the above experiments were re-run, but with each named entity type replaced with its equivalent type for each hierarchy. The results are shown in Table 6.8. Predictably, when gold standard EATs were used, the bigger the taxonomy, the greater the success in producing the correct answer. However, one problem with using a large taxonomy in question answering is the probability of inaccuracy at the question classification stage hurting the overall performance. To investigate this, answers were extracted for the 100 question QAC-1 set, using the EAT determined by the TiMBL machine classifier described in Section 3.3. The results are shown in the last column of Table 6.8. Surprisingly, despite the low question classification accuracy for the ENE taxonomy, it still provided the best results for answer extraction. This may be because in the larger taxonomies, question classification accuracy is higher over the more common types, where it matters more. Misclassifications were often amongst types that were not as helpful for answer extraction anyway. The hierarchical nature of the taxonomy was also beneficial, as it allowed the soft decision making strategy to be used, making the effective classification accuracy the same as the lenient evaluation results in Section 3.4. These results suggest that a large, hierarchical named entity taxonomy should be preferred for answer extraction,

	Sekine question set	**QAC-1 question set**	
	Gold Standard EAT	**Gold Standard EAT**	**Question Classification EAT**
ENE	0.345	0.28	0.24
NE28	0.308	0.28	0.22
NE15	0.295	0.25	0.21
IREX	0.270	0.23	0.19
NE5	0.242	0.19	0.18
NE4	0.226	0.18	0.16

Table 6.8: Results showing the effect of named entity taxonomies on question answering

for the discriminatory power it provides.

6.5 Summary

In order to take advantage of the robustness and language independent benefits the RMRS format can provide, I developed a sentence comparison algorithm that uses RMRS as its sentence representation. This algorithm was then used to match senses in a dictionary, by comparing their definition sentences. The RMRS comparison method proved to be more accurate in this task than the baseline bag-of-words model, even without using an ontology to detect synonyms.

The comparison algorithm was then used to detect the sentence most similar to a question, hoping to validate the claim by Hartrumpf (2005) that the answer to a question was often contained in a sentence that was semantically similar to the question. While this method only found an answer bearing sentence for about half of the questions, this was still significantly better than using the bag-of-words model in the same way. Adding named entity tagging to the RMRS structures to be compared increased the accuracy by 10%.

The question answering task was then attempted, using the comparison method to generate similarity scores for sentences, and for matching named entities to question words. Running this system over the question set for QAC-1, an accuracy of 0.28 was

achieved. During error analysis, it was found that the automatic evaluation script misclassified 8 correct answers and so the question set was hand evaluated. This produced an accuracy of 0.36 with an MRR of 0.4108 which would have made it the 4th ranked system at QAC-1.

Following up on Chapter 3, the effects of different named entity taxonomies on question answering performance were evaluated. Predictably, when using gold standard expected answer types, the accuracy of answer extraction was higher for larger taxonomies. Surprisingly, despite the inaccuracy of question classification, this trend continued when the gold standard EATs were replaced with those determined during question classification. This result shows that the discriminative power of a large, hierarchical taxonomy is highly beneficial to the question answering process.

Chapter 7

Conclusion

Question answering (QA) is a field that aims to enable a user to access the information they require, using a natural language question as the input to help pinpoint the exact information need. Cross-language question answering (CLQA) is an extension of QA where the question and the information source may not be in the same language. Looking towards the ultimate goal of a CLQA system, two potentially conflicting needs arise. The first comes from the fact that to take advantage of question-based inputs, at least some level of language understanding is required, and this indicates a need for using linguistically motivated methods. This has the potential to conflict with another need, that of language independence. As long as all techniques used in question answering are highly language specific, the goal of cross-language question answering will only be possible with significant amounts of work having to be repeated for each language that is to be used. With language-independent methods, we can hope to avoid this overhead, minimising the effort required to integrate a new language into a combined system. The aim of this thesis then was to investigate techniques that could use linguistic information while maintaining language independence wherever possible. This was accomplished by separating the methods and the data as much as possible, so the methods used linguistic information through the conduit of a common representation, namely Robust Minimal Recursion Semantics (RMRS).

In reviewing current question answering systems, it was found that the better performing systems did indeed use linguistically motivated methods in both question

processing and answer extraction, but that these methods, particularly in question classification, required huge amounts of manual effort that was not then reusable for a different task or language. It was also notable that Japanese question answering systems lagged behind those built for English and other European languages, with one potential reason being that in general they did not use linguistically motivated methods for answer extraction. This was one factor that motivated the use of Japanese question answering as the focus of this research.

Question classification is a task where language specific techniques such as pattern matching are often used. Machine learning techniques have been proven to perform well when identifying the expected answer type of a question, but they are often not used due to a perceived lack of training data. In this research I found that even with a relatively small amount of training data, machine learning outperformed pattern matching techniques in Japanese question classification, with much less manual effort. Reviewing similar experiments over English questions, it appears that putting effort into creating semantic information resources to be used as features in machine learning would lead to greater accuracy in classification than putting the same effort into creating more specific patterns for pattern matching, and would produce a more reusable resource.

In the context of question classification, I also explored the effects of using named entity taxonomies with different sizes and shapes. As expected, overall classification accuracy decreased when the taxonomies got larger, but surprisingly the accuracy over most of the common and potentially more useful types actually got higher with a larger taxonomy. Evaluations using soft decision making techniques, which are possible in answer extraction when hierarchical taxonomies are used, gave much better results over the larger taxonomies suggesting that a large hierarchical taxonomy should be preferred if the classification results are to be used for answer extraction. This suggestion was confirmed by the results achieved in the answer extraction experiments.

Many of the better question answering systems reviewed used question and sentence similarity as a focus for finding answers, motivating a review of similarity measurement techniques. RMRS was selected as the best sentence representation to use

when measuring similarity since it was expressive enough to represent the information provided by a full semantic parser, but could also represent information at the word-based or syntactic level for full robustness.

A novel comparison method that used this RMRS representation was developed that could make use of linguistic resources such as ontologies and gazetteers. This method performed much better than a bag-of-words based comparison method in detecting semantic similarity in the form of paraphrases, so was then used to find sentences that were most similar to questions in a question answering data set. These similar sentences contained answers in approximately 60% of cases. For the more difficult task of extracting the correct answer, the comparison algorithm was used to produce a similarity score for each sentence in the answer document, and then this score was used as part of the answer candidate score for any candidate found within that sentence. Results achieved were not as high as hoped, since the answer candidate extraction missed many candidates, but the ranking algorithm performed very well. As mentioned above, various named entity taxonomies were used in the answer extraction task, and the best results were achieved by using the full Extended Named Entity hierarchy, a large hierarchical taxonomy.

Further Work

While this research has confirmed that using linguistic resources in a language independent fashion is viable for question answering, a number of different research directions would contribute to the development of a fully featured question answering system. Various issues that were discussed during previous chapters could be addressed. These include, for question classification, looking at additional resources of semantic information (eg, alternative ontologies or distributional thesauri) and for answer extraction, improved candidate identification, since 40% of the errors in answer extraction were due to the correct answer not being within the potential answer candidates that were ranked.

There are three deeper issues that I think would be interesting to explore. In Chapter 3 and Chapter 6, named entity taxonomies were discussed. In this work,

the effect of size and flat versus hierarchical taxonomies was investigated, but the results raised the question of what type distinctions should be made. Despite the lower classification accuracy on larger taxonomies, these taxonomies still produced better question answer accuracy. The reason for this appeared to be higher accuracy on particular types. Research exploring what type distinctions are useful for question answering could lead to more 'efficient' taxonomies, by only distinguishing where this is beneficial. This research could start by looking at the confusion matrix for both entity recognition in sentences, and expected answer type classification. A review of the thesis results suggests, for example, that types that were variations on NUMBER were both easy to tag in a sentence and often accurately identified as the expected answer type, hence these are useful distinctions. The ORGANIZATION type however was difficult to classify accurately, probably because they can be treated as places and also as active entities. Perhaps it wold make more sense to classify the EAT in that fashion—active entity versus place—and allow recognised ORGANIZATIONS to answer either category. This idea would require much more investigation to verify its validity, but could lead to much better question answering results.

Another issue that warrants much further investigation is not directly linked to question answering, but to the goal of deep, robust linguistic processing. During this research reasonably naive methods were used to enhance the robustness of deep processing: firstly shallow fallback, and then, in a slightly more intelligent manner, merging complementary RMRS formatted outputs. In Chapter 5, reference was made to an idea of merging partial JACY parses with full CaboCha parses when JACY failed to produce a parse. This would allow the use of deep information where it's available, within the sentence. At a more general level, the idea of using shallow processing output to help link partial deep parses has a lot of scope for future research that would be of benefit to the whole NLP community. There are multiple stages to this task, first the identification of good partial parses, since a deep parser may produce many partial parses during processing, and not all will be valid. Methods for matching the partial parse to the correct section of the shallow output would then need to be explored, and finally how the parts should be joined. Issues of confidence in the shallow output would need to be investigated, since one of the reasons for using deep

processing is accuracy, and the dilution of this accuracy with shallow processing would need to be carefully controlled.

Finally, given the emphasis on maintaining language independence, the next interesting question would be to see how well these methods generalise to other languages, first in a monolingual system, and then moving towards the goal of a full cross-language question answering system. While reference was made, throughout the thesis, of RMRS contributing towards the goal of language independence, RMRS output is not language independent in itself. It is, however, considered an easier format from which to transfer between languages, and this claim needs to be investigated. A initial attempt might be a naive use of a bilingual dictionary, although this method assumes that corresponding words have the same predicate argument structure. The extent to which this is true needs to be investigated. Further techniques, such as those described in Copestake *et al.* (1995) could also be used.

Bibliography

Akiba, Yashiro, Kenji Imamura, and Eiichiro Sumita. 2001. Using multiple edit distances to automatically rank machine translation output. In *Proceedings of the MT Summit VIII Conference*, pp 15–20, Santiago de Compostela, Spain.

Amaral, Carlos, Helena Figueira, André Martins, Afonso Mendes, Pedro Mendes, and Cláudia Pinto. 2005. Priberam's question answering system for Portuguese. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The BerkeleyFrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pp 86–90, Montreal, Canada.

Bar-Haim, Roy, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pp 1–9, Venice, Italy.

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

Becker, Markus, Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2002. SProUT - Shallow Pro-

cessing with Typed Feature Structures and Unification. In *Proceedings of the International Conference on NLP (ICON 2002)*, Mumbai, India.

Bond, Francis, Sanae Fujita, Chikara Hashimoto, Kaname Kasahara, Shigeko Nariyama, Eric Nichols, Akira Ohtani, Takaaki Tanaka, and Shigeaki Amano. 2004a. The Hinoki treebank: A treebank for text understanding. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pp 158–167, Hainan Island, China.

——, Eric Nichols, Sanae Fujita, and Takaaki Tanaka. 2004b. Acquiring an ontology for a fundamental vocabulary. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pp 1319–1325, Geneva, Switzerland.

Bouma, Gosse, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. 2005. Question answering for Dutch using dependency relations. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Breck, Eric, Marc Light, Gideon S. Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Michael Thelen. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of theACL-2001 Workshop on Open-Domain Question Answering*, pp 1–8, Toulouse, France.

Burger, John, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees, and Ralph Weishedel, 2001. Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A). `http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc`.

Callison-Burch, Chris, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp 17–24, New York City, USA.

Callmeier, Ulrich, 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Saarland University.

Clifton, Terence, and William Teahan. 2004. Bangor at TREC 2004: Question answering track. In *The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, USA.

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

——. 2003. Report on the design of RMRS. Report D1.1a, Deep Thought, Cambridge, U.K.

——, 2004. Robust minimal recursion semantics. working paper.

——, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan A. Sag. 1995. Translation using minimal recursion semantics. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*, pp 15–32, Leuven, Belgium.

——, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation* vol 3.pp 281–332.

Corley, Courtney, and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pp 13–18, Ann Arbor, USA.

Cui, Hang, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen Kan. 2004. National University of Singapore at the TREC-13 question answering main task.

In *The Thirteenth Text REtrieval Conference (TREC 2004)*, pp 34–42, Gaithersburg, USA.

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

Dagan, Ido, Oren Glickman, and Bernado Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pp 1–8, Southampton, U.K.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*. New York, USA: Academic Press.

——, John Lamping, and Vijay Saraswat. 1993. LFG semantics via constraints. In *Proceedings of the Sixth Conference of the European Association for Computational Linguistics*, pp 97–105, Utrecht, The Netherlands.

Dowty, David. 1989. On the semantic content of the notion of thematic role. In *Properties, Types and Meaning*, ed. by G. Cherchia, B. Partee, and R. Turner, volume 2, pp 69–129. Dordrecht, The Netherlands: Kluwer.

Dras, Mark, and Kazuhide Yamamoto (eds.) 2005. *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, Jeju Island, South Korea.

Dridan, Rebecca, and Francis Bond. 2006. Sentence comparison using Robust Minimal Recursion Semantics and an ontology. In *Proceedings of the Workshop on Linguistic Distances*, pp 35–42, Sydney, Australia.

Emms, Martin. 2006. Variants of tree similarity in a question answering task. In *Proceedings of the Workshop on Linguistic Distances*, pp 100–108, Sydney, Australia.

Firth, J. 1957. A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis* Special volume of the Philological Society.pp 1–32.

Frank, Anette, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. 2006. Question answering from structured knowledge sources. *Journal of Applied Logic* Special Issue on Questions and Answers: Theoretical and Applied Perspectives.pp 1–29.

Fuchigami, Masachika, Hiroyuki Ohnuma, and Atsushi Ikeno. 2004. Oki QA system for QAC-2. In *Working Notes of the Fourth NTCIR Workshop Meeting*, Tokyo, Japan.

Fukumoto, Jun'ichi, Tsuneaki Kato, and Fumito Masui. 2002. Question Answering Challenge (QAC-1): An Evaluation of Question Answering Task at NTCIR Workshop 3. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, pp 77–86, Tokyo, Japan.

Fukumoto, Junichi, Tsuneaki Kato, and Fumito Masui. 2004. Question Answering Challenge for five ranked answers and list answers - Overview of NTCIR4 QAC2 Subtask 1 and 2. In *Working Notes of the Fourth NTCIR Workshop Meeting*, pp 283–290, Tokyo, Japan.

Gonzalo, Julio, Paul Clough, and Alessandro Vallin. 2005. Overview of the CLEF 2005 interactive track. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Greenwood, Mark A., Ian Roberts, and Robert Gaizauskas. 2002. The University of Sheffield TREC 2002 Q&A system. In *The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, USA.

Han, Kyoung-Soo, Hoojung Chung, Sang-Bum Kim, Young-In Song, Joo-Young Lee, and Hae-Chang Rim. 2004. Korea University question answering system at TREC 2004. In *The Thirteenth Text REtrieval Conference (TREC 2004)*, pp 446–455, Gaithersburg, USA.

Harabagiu, Sanda, Dan Moldovan, Christine Clark, Mitchell Bowden, Andrew Hickl, and Patrick Wang. 2005. Employing two question answering systems in trec-2005. In *The Fourteenth Text REtrieval Conference (TREC 2005)*, Maryland, USA.

Hartrumpf, Sven. 2005. University of Hagen at QA@CLEF 2005: Extending knowledge and deepening linguistic processing for question answering. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Herrera, Jesús, Anselmo Pe nas, and Felisa Verdejo. 2004. Question answering pilot task at CLEF 2004. In *Proceedings of the CLEF 2004 Workshop*, pp 581–590, Bath, United Kingdom.

Hovy, Eduard, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pin pointing. In *Proceedings of the HLT 2001 Conference*, pp 1–7, San Diego, California.

——, Chin-Yew Lin, and Liang Zhao. 2005. A BE-based multi-document summarizer with sentence compression. In *Proceedings of the ACL-2005 Workshop on Multilingual Summarization Evaluation*, Ann Arbor, Michigan.

Inui, Kentaro, and Ulf Hermjakob (eds.) 2003. *Proceedings of the Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP2003)*, Sapporo, Japan.

Isozaki, Hideki. 2004. NTT's question answering system for NTCIR QAC2. In *Working Notes of the Fourth NTCIR Workshop Meeting*, pp 326–332, Tokyo, Japan.

Judge, John, Michael Burke, Aoife Cahill, Ruth Donovan, Josef van Genabith, and Andy Way. 2005. Strong domain variation and treebank-induced LFG resources. In *Proceedings of the LFG05 Conference*, pp 186–204, Bergen, Norway.

Kaisser, Michael, and Tilman Becker. 2004. Question answering by searching large corpora with linguistic methods. In *The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, USA.

Kasahara, Kaname, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. 2004. Construction of a Japanese semantic lexicon: Lexeed. SIG NLC-159, IPSJ, Tokyo, Japan.

Kato, Tsuneaki, Jun'ichi Fukumoto, and Fumito Masui. 2004. Question answering challenge for information access dialogue - Overview of NTCIR4 QAC2 Subtask 3 -. In *Working Notes of the Fourth NTCIR Workshop Meeting*, pp 291–297, Tokyo, Japan.

Kimura, Yasumoto, Kinji Ishida, Hirotaka Imaoka, Fumito Masui, Marcin Skowron, Rafal Rzepka, and Kenji Araki. 2005. Three systems and one verifier - HOKUM's participation in QAC3 of NTCIR-5. In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*, Tokyo, Japan.

Kingsbury, Paul, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn treebank. In *Proceedings of the Human Language Technology 2002 Conference*, pp 252–256, San Diego, California.

Kudo, Taku, and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*, pp 63–69, Taipei, Taiwan.

Kurata, Gakuto, Naoaki Okazki, and Mitsuru Ishizuka. 2004. GDQA: Graph driven question answering system - NTCIR-4 QAC2 Experiments. In *Working Notes of the Fourth NTCIR Workshop Meeting*, pp 338–344, Tokyo, Japan.

Laurent, D. 2005. Qristal at QA@CLEF. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Laurent, Dominique, Patrick Séguéla, and Sophie Nègre. 2005. Cross lingual question answering using QRISTAL for CLEF 2005. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Lehnert, Wendy G. 1981. A computational theory of human question answering. In *Elements of Discourse Understanding*, ed. by Aravind K. Joshi, Bonnie J. Weber, and Ivan A. Sag, pp 145–176. Cambridge, U.K.: Cambridge University Press.

Leidner, Jochen L. 2002. Question answering over unstructured data without domain restriction. In *Proceedings of TaCoS 2002*, Potsdam, Germany.

Li, Xin, and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering* 12.pp 209–228.

Lin, Chin-Yew, and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the ACL*, pp 605–612, Barcelona, Spain.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL98*, pp 768–774, Montreal, Canada.

Magnini, Bernado, Simone Romagnoli, Alessandro Vallin, Jesûs Herrera, Anselmo Peñas, Victor Peinado, Felisa Verdejo, and Maarten de Rijke. 2003. The multiple language question answering track at CLEF 2003. In *Working Notes for the CLEF 2003 Workshop*, pp 471–486, Trondheim, Norway.

Matsumoto, Yuuji, Akira Kitauchi, Tatsu Yamashita, and Yoshitake Hirano. 1999. Japanese morphological analysis system ChaSen version 2.0 manual. Technical Report NAIST-IS-TR99009, NAIST, Nara, Japan.

Maybury, Mark T. (ed.) 2004. *New Directions in Question Answering*. Cambridge, USA: The MIT Press.

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography* vol 3.pp 235–244.

Müller, Karin. 2004. Semi-automatic construction of a question treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Narayanan, Srini, and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pp 693–701, Geneva, Switzerland.

Nicholson, Jeremy, Nicola Stokes, and Timothy Baldwin. 2006. Detecting entailment using an extended implementation of the Basic Elements overlap metric. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pp 122–127, Venice, Italy.

Nii, Yasuo, Keizo Kawata, Tatsumi Yoshida, Hiroyuki Sakai, and Shigeru Masuyama. 2004. Question answering system QUARK. In *Working Notes of the Fourth NTCIR Workshop Meeting*, Tokyo, Japan.

Nishio, Minoru, Etsutaro Iwabuchi, and Shizuo Mizutani. 1994. *Iwanami Kokugo Jiten Dai Go Han [Iwanami Japanese Dictionary Edition 5]*. Tokyo, Japan: Iwanami Shoten.

Pantel, Patrick, and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *HLT-NAACL 2004: Main Proceedings*, pp 321–328, Boston, USA.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings*

of the 40th Annual Meeting of the ACL and 3rd Annual Meeting of the NAACL (ACL-02), pp 311–318, Philadelphia, USA.

Paşca, Marius. 2003. *Open-Domain Question Answering from Large Text Collections*. Chicago, USA: The University of Chicago Press.

Pedersen, Ted, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, pp 38–41, Boston, USA.

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

Quaresma, Paulo, and Irene Rodrigues. 2005. A logic programming based approach to the QA@CLEF05 track. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Rinaldi, Fabio, James Dowdall, Kaarel Kaljurand, Michael Hess, and Diego Molla. 2003. Exploiting paraphrases in a question answering system. In *Proceedings of the Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP2003)*, pp 25–32, Sapporo, Japan.

Ritchie, Anna. 2004. Compatible RMRS representations from RASP and the ERG. Technical Report UCAM-CL-TR-661, University of Cambridge, Cambridge, U.K.

Sakai, Tetsuya, Yoshimi Saito, Yumi Ichimura, Makoto Koyama, and Tomoharu Kokubu. 2004. Toshiba ASKMi at NTCIR-4 QAC2. In *Working Notes of the Fourth NTCIR Workshop Meeting*, pp 387–394, Tokyo, Japan.

Sasaki, Yutaka, Hsin-Hsi Chen, Kuang hua Chen, and Chuan-Jie Lin. 2005. Overview of the NTCIR-5 cross-lingual question answering task (CLQA1). In

Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access, Tokyo, Japan.

Sato, Satoshi, and Hiroshi Nakagawa (eds.) 2001. *Workshop Proceedings for Automatic Paraphrasing: Theories and Applications*, Tokyo, Japan.

Schäfer, Ulrich. 2006. Middleware for creating and combining multi-dimensional NLP markup. In *Proceedings of the EACL-2006 Workshop on Multi-dimensional Markup in Natural Language Processing*, pp 81–84, Trento, Italy.

Sekine, Satoshi, and Hitoshi Isahara. 1999. IREX project overview. In *Proceedings of the IREX Workshop*, pp. 7–12, Tokyo, Japan.

——, Kiyoshi Sudo, and Chikashi Nobata. 2002a. Extended named entity hierarchy. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, pp 1818–1824, Las Palmas, Spain.

——, Kiyoshi Sudo, Yusuke Shinyama, Chikashi Nobata, Kiyotaka UCchimoto, and Hitoshi Isahara. 2002b. NYU/CRL QA system, QAC question analysis and CRL QA data. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, pp 79–86, Tokyo, Japan.

Shinyama, Yusuke, and Satoshi Sekine. 2003. Paraphrase acquisition for information extraction. In *Proceedings of the Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP2003)*, pp 65–71, Sapporo, Japan.

——, ——, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the Human Language Technology 2002 Conference*, pp 40–46, San Diego, California.

Shirai, Kiyoaki. 2002. Construction of a word sense tagged corpus for SENSEVAL-2 Japanese dictionary task. In *Proceedings of the Third International Conference*

on Language Resources and Evaluation (LREC-2002), pp. 605–608, Las Palmas, Spain.

Siegel, Melanie, and Emily M. Bender. 2002. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics*, pp 31–38, Taipei, Taiwan.

Soriano, José Manuel Gómez, Manuel Montes y Gómez, Emilio Sanchis Arnal, Luis Villaseñor Pineda, and Paolo Rosso. 2005. Language independent passage retrieval for question answering. In *Proceedings of MICAI 2005: Advances in Artificial Intelligence, 4th Mexican International Conference on Artificial Intelligence*, pp 816–823, Monterrey, Mexico.

Spreyer, Kathrin, and Anette Frank. 2005. The TIGER RMRS 700 bank: RMRS construction from dependencies. In *Proceedings of the 6th International Workshop on Linguistically Interpreted Corpora (LINC 2005)*, pp 1–10, Jeju Island, Korea.

Tanev, Hristo, Milen Kouylekov, and Bernardo Magnini. 2004. Combining linguistic processing and web mining for question answering: ITC-irst at TREC-2004. In *The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, USA.

Tiedemann, Jörg. 2005. Integrating linguistic knowledge in passage retrieval for question answering. In *Proceedings of EMNLP 2005*, pp 939–946, Vancouver, Canada.

Tomás, David, José L. Vicedo, Maximiliano Saiz, and Rubén Izquierdo. 2005. Building an XML framework for question answering. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Vallin, Alessandro, Bernardo Magnini, Danilo Giampiccolo, Lili Aunimo, Christelle Ayache, Petya Osenova, Anselmo Pe nas, Maarten

de Rijke, Bogdan Sacaleanu, Diana Santos, and Richard Sutcliffe. 2005. Overview of the CLEF 2005 multilingual question answering track. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria.

Vanderwende, Lucy, Deborah Coughlin, and Bill Dolan. 2005. What syntax can contribute in entailment task. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pp 13–16, Southampton, U.K.

Voorhees, Ellen M. 1999. The TREC-8 question answering track report. In *The Eighth Text REtrieval Conference (TREC-8)*, pp 77–82, Gaithersburg, USA.

—— 2003. Overview of the TREC 2003 question answering track. In *The Twelfth Text REtrieval Conference (TREC 2003)*, pp 54–68, Gaithersburg, USA.

—— 2004. Overview of the TREC 2004 question answering track. In *The Thirteenth Text REtrieval Conference (TREC 2004)*, pp 59–69, Gaithersburg, USA.

Witten, Ian H., Alistair Moffat, and Timothy C. Bell. 1999. *Managing gigabytes: compressing and indexing documents and images*. San Francisco, USA: Morgan Kaufmann Publishers, 2nd edition.

Appendix A

Named Entity Taxonomies

A.1 NE4

PERSON
LOCATION
NUMBER
THING

A.2 NE5

PERSON
LOCATION
NUMBER
THING
ORGANIZATION

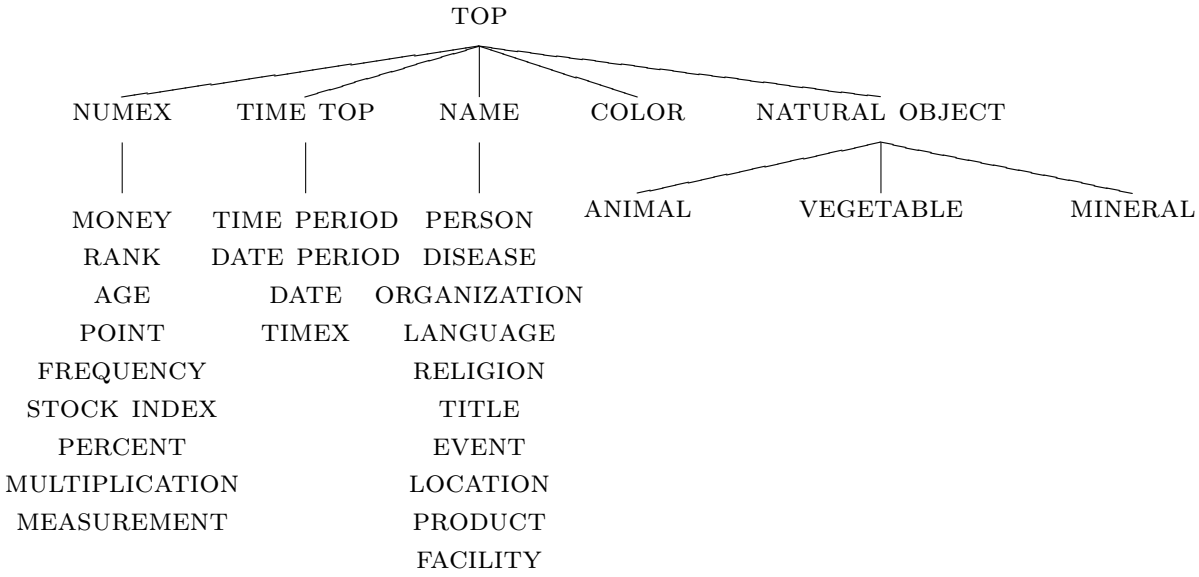
A.3 IREX

PERSON
LOCATION
NUMBER
THING
ORGANIZATION
MONEY
PERCENT
TIME
DATE

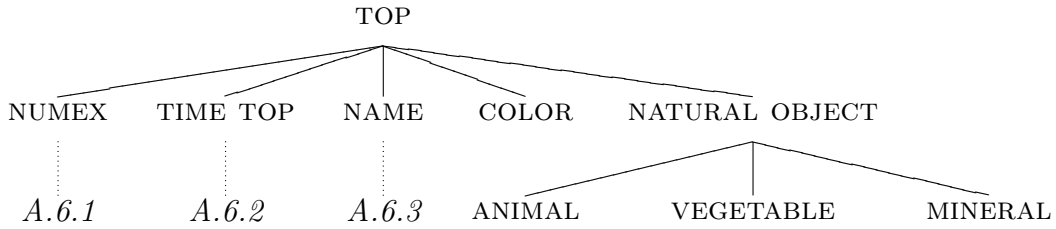
A.4 NE15 Taxonomy

PERSON
LOCATION
ORGANIZATION
NUMBER
TIME
TIME PERIOD
DISEASE
LANGUAGE
RELIGION
COLOR
EVENT
PRODUCT
ANIMAL
VEGETABLE
MINERAL

A.5 NE28 Taxonomy



A.6 Sekine's Extended Named Entity Hierarchy



A.6.1 NUMEX branch

MONEY	
STOCK INDEX	
POINT	
PERCENT	
MULTIPLICATION	
FREQUENCY	
RANK	
AGE	
MEASUREMENT	LENGTH AREA VOLUME WEIGHT SPEED INTENSITY TEMPERATURE CALORIE SEISMIC INTENSITY
COUNTX	N PERSON N ORGANIZATION N LOCATION N COUNTRY N FACILITY N PRODUCT N EVENT N ANIMAL N VEGETABLE N MINERAL

A.6.2 TIME TOP branch

TIMEX	TIME DATE ERA
PERIODX	TIME PERIOD DATE PERIOD WEEK PERIOD MONTH PERIOD YEAR PERIOD

A.6.3 NAME branch

DISEASE		
LANGUAGE		
RELIGION		
PERSON	LASTNAME	
	MALE FIRSTNAME	
	FEMALE FIRSTNAME	
EVENT	GAMES	
	CONFERENCE	
	PHENOMENA	
	WAR	
	NATURAL DISASTER	
	CRIME	
TITLE	POSITION TITLE	
ORGANIZATION	COMPANY	
	COMPANY GROUP	
	MILITARY	
	INSTITUTE	
	MARKET	
	ETHNIC GROUP	
	NATIONALITY	
	GROUP	SPORTS TEAM
	POLITICAL ORGANIZATION	GOVERNMENT
		POLITICAL PARTY
		PUBLIC INSTITUTION
LOCATION	REGION	
	GEOLOGICAL REGION	LANDFORM
		WATERFORM
		SEA
	ASTRAL BODY	STAR
		PLANET
	ADDRESS	POSTAL ADDRESS
		PHONE NUMBER
		EMAIL
		URL
	GPE	CITY
		COUNTY
		PROVINCE
		COUNTRY

PRODUCT	DRUG		
	WEAPON		
	STOCK		
	CURRENCY		
	AWARD		
	THEORY		
	RULE		
	SERVICE		
	CHARACTER		
	METHOD SYSTEM		
	ACTION MOVEMENT		
	PLAN		
	ACADEMIC		
	CATEGORY		
	SPORTS		
	OFFENSE		
	ART	PICTURE TV PROGRAM MOVIE SHOW MUSIC	
	PRINTING	BOOK NEWSPAPER MAGAZINE	
	VEHICLE	CAR TRAIN AIRCRAFT SPACESHIP SHIP	
FACILITY	PARK		
	MONUMENT		
	LINE	RAILROAD ROAD WATERWAY TUNNEL BRIDGE	
	GOE	SCHOOL MUSEUM AMUSEMENT PARK WORSHIP PLACE	
		STATION TOP	AIRPORT STATION PORT CAR PARK