Topic Indexing and Retrieval for Open Domain Factoid Question Answering

Kisuh Ahn

Doctor of Philosophy

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh

2009

Abstract

Factoid Question Answering is an exciting area of Natural Language Engineering that has the potential to replace one major use of search engines today. In this dissertation, I introduce a new method of handling factoid questions whose answers are proper names. The method, *Topic Indexing and Retrieval*, addresses two issues that prevent current factoid QA system from realising this potential: They can't satisfy users' demand for almost immediate answers, and they can't produce answers based on evidence distributed across a corpus.

The first issue arises because the architecture common to QA systems is not easily scaled to heavy use because so much of the work is done on-line: Text retrieved by information retrieval (IR) undergoes expensive and time-consuming answer extraction while the user awaits an answer. If QA systems are to become as heavily used as popular web search engines, this massive process bottle-neck must be overcome.

The second issue of how to make use of the distributed evidence in a corpus is relevant when no single passage in the corpus provides sufficient evidence for an answer to a given question. QA systems commonly look for a text span that contains sufficient evidence to both locate and justify an answer. But this will fail in the case of questions that require evidence from more than one passage in the corpus.

Topic Indexing and Retrieval method developed in this thesis addresses both these issues for factoid questions with proper name answers by restructuring the corpus in such a way that it enables direct retrieval of answers using off-the-shelf IR. The method has been evaluated on 377 TREC questions with proper name answers and 41 questions that require multiple pieces of evidence from different parts of the TREC AQUAINT corpus. With regards to the first evaluation, scores of 0.340 in Accuracy and 0.395 in Mean Reciprocal Rank (MRR) show that the Topic Indexing and Retrieval performs well for this type of questions. A second evaluation compares performance on a corpus of 41 *multi-evidence questions* by a question-factoring baseline method that can be used with the standard QA architecture and by my Topic Indexing and Retrieval method. The superior performance of the latter (MRR of 0.454 against 0.341) demonstrates its value in answering such questions.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Kisuh Ahn*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Vast amount of electronic text data is available today. While it is relatively easy to access the data due to the Internet infrastructure, the access to the desired information is not as easy, as more data means often more search and scrutiny[1].

At present, search engines such as Google dominate the means of information access with respect to the World Wide Web. However, search engines today are primarily the means of searching and locating data (physical web pages, image files, etc.) rather than information, and the fulfilment of the information needs of the users is only achieved by sifting through the data returned by the search engines, a task that would become more difficult if larger amount of data would have to be examined.

Automatic Question Answering, on the other hand, promises to deliver the information itself, in the form of answers to users' natural language questions. As of yet, there are still many challenges that have to be overcome for Question Answering to become a viable technology that can be used for real life application in place of the search engines of today.

The fundamental problem is that many questions require some degree of intelligence, for the machine to really comprehend the question, to understand and infer from the available evidence and to formulate a coherent response. A logical step toward solving this difficult problem is to first tackle the questions that are relatively simple and require less intelligence. Hence, factoid questions (questions that can be answered by simple phrases) have been the main targets of research for the past several years as these questions often require not much more than techniques to identify and extract similar patterns of text with respect to a question from a large text corpus. They

[1] According to Wikipedia (http://en.wikipedia.org/information (January 10th 2008)), "Information is the result of processing, gathering, manipulating and organising data in a way that adds to the knowledge of the receiver."

are also the focus of this thesis.

As simple as factoid questions are however, the QA operations involved are still anything but simple. Many complex and often expensive NLP operations are performed including parsing, named entity recognition, etc. At present, many of these tasks are performed on-line, after the question has been put, due to the process pipeline of the common architecture that most QA systems subscribe to. For real life applications, the expensive on-line operations involved may very well cause a scaling problem due to each question taking up significant resources and time. For instance, tens of thousands of concurrent users is the kind of scale that is common to Google Search Engine that a practical QA system must be able to match ultimately.

In this thesis, I argue that the QA architecture for factoid questions with proper name answers can be simplified with respect to on-line processing. I claim, for such questions, the Question Answering can be turned into a very fine-grained Information Retrieval that retrieves the answers directly without sacrificing the accuracy of the answers. This is in contrast to most QA systems that employ Information Retrieval to fetch the base material for the eventual answer extraction operation. This is achieved, I claim, through the proposed method in this thesis, namely, *Topic Indexing and Retrieval for Question Answering*. This method involves an extensive preprocessing of the base material (textual corpus) to identify potential answers (topics), to gather textual evidence for each of them and to index them for direct retrieval as answers to questions.

A further claim regarding this method is that it achieves another difficult task, namely answering questions that depend on evidence distributed throughout the corpus. This is a task that has not previously been paid much attention, but that, as I show in Chapter 6, is difficult to accomplish using the conventional QA architecture.

Thus, the main claims for the thesis are as follows:

1. The method of Topic Indexing and Retrieval enables direct retrieval of answers using Information Retrieval technique for questions with proper name answers.

2. This method can handle questions which require distributed evidence more effectively as compared to the more conventional QA architecture based systems.

The thesis consists of seven chapters. This first chapter has presented the main topic and the claims of the thesis. In Chapter 2, a general introduction to the field of Question Answering is given by reviewing different areas within the field of Question

Answering and briefly overviewing its historical development. Then focusing on the open domain textual QA, which is the subject of the thesis, the common architecture found in many recent QA systems for this type of task is explained in detail.

Chapter 3 presents the related work. First, Document IR, which is commonly used in QA, is explained and some of the limitations pointed out, which provide the motivation for this thesis. Then, alternative approaches to IR for QA are discussed as they relate to the thesis' own approach.

Chapter 4 gives the main method of the thesis, which I call *Topic Indexing and Retrieval for QA*. Following the discussion of the theoretical framework underpinning this method, I describe how to implement this method, which involves the preparation of base material (preprocessing) and the answer-retrieval operations.

Chapter 5 is one of the two evaluation chapters, together with Chapter 6. Here, the effectiveness of the method with respect to simple factoid TREC questions is evaluated. The superior performance on these questions compared to a baseline system of a more conventional architecture gives support to my first claim that factoid QA can be turned into a fine-grained information retrieval without sacrificing accuracy of answers.

Chapter 6 provides evidence in support of my other claim, that the method is effective in dealing with questions requiring distributed evidence. This chapter first defines what these questions are and what special strategy can be adopted to tackle them by the more conventional systems. Taking the experimental results of this special strategy based QA system as the baseline, the performance of the main method is compared, which demonstrates that this method is more effective on this type of questions.

Finally, Chapter 7 concludes the thesis with a final assessment of thesis' claims and discussions of limitations and future work.

Chapter 2

An Overview of Open Domain Question Answering

This chapter presents a general introduction to Automatic Question Answering via a brief overview of its development from the early database oriented Question Answering to today's text oriented Question Answering driven by TREC (*TExt Retrieval Conference*). The more current practice of Question Answering, which can be characterised as *open domain textual QA*, is investigated in depth, especially focusing on the general architecture of the QA systems that belong to this class.

2.1 Introduction to Question Answering

2.1.1 What is Question Answering?

Web search engines such as Google are often used for meeting information needs of users. A search engine basically returns a ranked list of documents (web pages) as the result of a user's query. In contrast, a Question Answering system promises to deliver the most relevant information without the burden of having to read an entire document (or even worse, a set of documents) which contains it, in other words, precise and to-the-point answer to the user's question.

In Question Answering, different information needs can be met by different questions. For example, there are questions that require short factual answers such as "Who is the president of France?", and there are questions that ask for general information about certain topic (e.g. "What is the cause of World War II?"). The former is called *Factoid QA* (the focus of this thesis), and the latter is a more in-depth Question An-

swering related to *query-oriented summarisation*.[1] More generally, Question Answering can be subdivided into different tasks depending on the structure and coverage of information being queried in the following ways.

First, depending on whether the domain of expertise of a QA system is some specific area such as medicine or is more general (or less specific), Question Answering is divided into *restricted domain QA* and *open domain QA*. Restricted domain QA (or domain specific QA) would be useful for people whose information need is expert knowledge, and consequently a restricted domain QA system has to be able to deliver highly specific information on the chosen domain, which would be achieved via an expert knowledge-base often augmented with purpose built domain specific ontology (Mollá and Vicedo, 2007). On the other hand, open domain Question Answering would be useful for users having the information need of looking up general facts.

Next, the source of knowledge is also another criteria: *database QA* relies on a pre-built structured database as its source of knowledge whereas *textual QA* relies on unstructured text (ideally vast amounts of it). The former is also called *natural language interface to database* and is often closely related to the restricted domain QA (Androutsopoulos *et al.*, 1995). For example, in the medical domain, data on diseases would have already been collected and stored into a database by domain experts, and the goal would be to query this database using natural language questions, for example by medical practitioners who are experts in medicine but not in database querying. In contrast, textual Question Answering makes direct use of raw textual data such as newspaper articles stored in an archive or as publicly available web documents. Open domain QA would more likely rely on this type of data as the wide varieties of facts contained in the vast but consequently unprocessed raw text data would be very useful. However, even in restricted domain, a lot of text data remains unprocessed (e.g. a digital archive of medical journal articles such as found in *PubMed Central* (http://www.pubmedcentral.nih.gov/)) but merely stored electronically, and thus, the techniques for raw text QA would benefit both areas of applications (Lee *et al.*, 2006).

If the goal of QA is to be able to answer questions about some specific passage, then the QA task is called *Reading Comprehension*. This is contrasted to the general Question Answering in that the goal is to test the level of comprehension of the text by a machine via subjecting it to answering questions, like SAT or TOEFL reading comprehension assessments where the goal is to test the level of text comprehension

[1] Document Understanding Conferences (or simply DUC) have been one of the venues for this type of Question Answering. (Dang, 2006)

by human students. Hence, Question Answering in this case is more of a tool for assessment than for the provision of information to users.

Finally, the degree of interactivity, whether the question answer session consists of one or more independent QA cycle or not is another criteria. *Interactive QA*, which requires some dialogue facility, is distinguished from non-interactive QA in that one can refine the question or provide feed-back to the machine as regards to the answers produced, and the context of the QA would be preserved for any follow-up questions. This area of QA is attracting considerable interests recently as witnessed by the introduction of *CiQA* (*Complex interactive Question Answering*) task into TREC 2006 QA (Kelly and Lin, 2007).

Each type of QA outlined above serves different purpose and offers plenty of challenges, and none has been conclusively solved. For this thesis, I only consider the non-interactive, text-based, open domain, factoid QA as my area of focus. This area is arguably the most basic type of Question Answering, which however can still benefit tremendously from further research and development.

2.1.2 Some of the Early QA Systems

The early practical QA systems could be broken into three types: *natural language interface to database systems*, *text based systems* and *reading comprehension systems*.

First, the QA systems that were a-front-end systems to database provided a natural language interface to pre-assembled data stored in a database of some restricted domain so that people could query the database in natural English without having to understand the structure of the database or the specific ways to build the queries for it. Baseball (Green *et al.*, 1961) and Lunar (Woods *et al.*, 1972) are two of the examples of these early systems.

Baseball was a question answering system designed to answer questions related to baseball games played in a season of American League Baseball. All the necessary data (e.g. batting statistics) had already been assembled and stored in a database. Once a question is put by a user, the system analysed the question and turned it into a database query. It was a relatively sophisticated system being able to answer questions such as “How many games did the Yankees play in July?”

Lunar answered questions about chemical analyses of soil and rocks from moon surface brought back by the Apollo 11 space mission. The system relied on a database that contained all chemical analyses done to-date on the Apollo 11 samples. Like Base-

ball, an English question by a user was parsed into database query, via an augmented transition network parser. The system was quite successful being able to answer many of the questions put up by the users at the site of demonstration.

The main challenge of this type of QA is how to turn the natural language question into the appropriate database query. The mapping is far from trivial and is often only achieved by an extensive set of rules crafted by hand that matches a given type of question into a given type of query. For example, in Lunar, a very specialised dictionary was constructed with immense effort in order to map the terms possibly contained in a question to the relevant queries to the database. This makes the process too tied up to specific database, as the set of rules crafted for a specific database may well lack generality. This is the reason why this type of QA is more applicable to restricted domains with existing databases rather than open domain QA.

There were also text based QA systems, which are the direct precursors to today's textual QA systems. The Oracle System (Phillips, 1960) transforms simple sentences of the corpus into canonical forms marking the subject, verb and object as well as time and place indicators. The question is also transformed likewise after having been converted into a declarative form. A simple matching algorithm finds the best matching sentence in the corpus to the question via the similarity of the converted forms. Although more complex sentences are not analysed in this system, the basic method of Oracle is an important precedence to the basic method of today's text based QA systems. The Automatic Analyzer (ALA) (Thorne, 1962) is another example of early text based QA system that works essentially in a similar fashion as the Oracle. Although directly relevant to today's corpus based QA systems, at the time, the lack of the existence of large electronic corpus hindered the further development of this approach.

There were also QA systems focused on Reading Comprehension. In Lehnert (1978), passage is analysed and parsed into a conceptual dependency representation (Schank, 1972), which is then used to answer questions about the target passage. The goal is to test how well the text has been understood, and Question Answering is used as a means to assess this. Although this type of QA has some similarity to the more current text based QA in that the source of knowledge is raw text, the text is much shorter, and the goal is not necessarily to build a system that can answer general questions or questions related to facts outside of the small target passage.

All these early systems were interesting and pioneering in their respective focus. However, the level of research activity in Question Answering declined for a while until the resurgence of interests due to the phenomenal expansion of the on-line text

data (the web) and the incorporation of Question Answering into TREC from 1999.

2.2 Factoid Question Answering ala TREC

Starting from 1992, Text REtrieval Conference (TREC) has provided the platform on which information retrieval and related systems could perform and compare large scale evaluations. From 1999, Question Answering was added as one of the principal track of investigation in TREC (Voorhees, 1999), and has served to fuel the resurgence and drive the development of the research in this field ever since. [2]

Initially, the task was to find a small paragraph that contained the answer to a question. However beginning from TREC 10 (Voorhees, 2001) in 2001, the task has been made more difficult by requiring to produce succinct answers, usually no more than phrases along with the documental evidence (from the corpus) that support the answers. Although the focus for TREC had been primarily factoid question answering, the emphasis now (from TREC 2006) is moving away from the factoid to more interactive and complex QA. However, I will only concern with the factoid Question Answering here.

Factoid Question Answering is the task of answering factual trivia-type questions. An example of this type of question is "Who killed JFK?". Such questions can be answered by a simple phrase such as "Lee Harvey Oswald". The answers to this type of questions are mostly noun phrases, often named entities (proper names) such as the names of persons, locations etc. In textual QA, answering questions of this type relies on textual source (e.g. a passage) that contains the answer along with the supporting evidence. The following passage is an example:

"... The committee concluded that Lee Harvey Oswald fired three shots at President John F. Kennedy. The second and third shots he fired struck the President. The third shot he fired killed him. ..." [3]

Presuming that the above passage came from a collection of much larger text, the first important task is to locate it. This is often achieved via Information Retrieval that retrieves candidate passages via a query constructed from the target question.

Once a passage has been located as a candidate, it is necessary to further narrow

[2]There are also other similar venues such as CLEF (*Cross Language Evaluation Forum*, http://www.clef-campaign.org) where the emphasis is on cross-lingual IR/QA and NTCIR (NTCIR Cross-Language Question Answering, http://www.slc.atr.jp/CLQA/), a Japan based IR and QA initiative focused on Asian languages.

[3]From Wikipedia Article "John F. Kennedy assassination" (10/9/2007

down the search to the relevant part of the text. Since the part that states the relevant fact may not be identical to the way the question is stated, NLP techniques such as anaphora resolution or entailment inference can to be used to locate question and answer. For a large enough corpus however, it is often possible to locate near verbatim match between some part of the text and the question (in the declarative form) so as not to need any sophisticated NLP operations.

Finally, the element that is the short succinct answer to the question ("Lee Harvey Oswald" in this case) must be identified and extracted from this passage using techniques such as named entity identification.

What can be seen from this toy example is that textual factoid QA is an operation that repeatedly narrows down to the final element from a much larger text. For this reason, textual QA can be regarded as a form of fine-grained information retrieval as compared to document based IR. Thus, the corpus is very important for this type of QA from which the answer is to be eventually located.

In order to facilitate an objective comparison, TREC requires all the participating systems to use the same corpus. Up until 2007, the source corpus has been AQUAINT, which is a collection of news articles from 1996 to 2000 consisting of Xinhua English news reports, Associated Press news reports and NY Times newspaper articles. The task is essentially that described in the toy example, but since TREC 2004 (Voorhees, 2004), an additional element, namely *targets* are introduced. Below is a sample of the questions from TREC 2005 in the original XML format.

```
<target id="74" text="DePauw University">
  <qa><q id="74.1" type="FACTOID">
     What type of school is DePauw?
  </q></qa>
  <qa><q id="74.2" type="FACTOID">
     Where is DePauw located?
  </q></qa>
  <qa><q id="74.3" type="FACTOID">
     When was DePauw founded?
  </q></qa>
  <qa><q id="74.4" type="FACTOID">
     Who was president of DePauw in 1999?
  </q></qa>
  <qa><q id="74.5" type="FACTOID">
     What was the approximate number of students attending in 1999?
  </q></qa>
  <qa><q id="74.6" type="LIST">
     Name graduates of the university.
```

```
    </q></qa>
    <qa><q id="74.7" type="OTHER">
       Other
    </q></qa>
  </target>
```

The target, in the above example, "DePauw University", is the topic of all the questions in this group. All the answers to these questions must relate to this topic one way or the other.

This example also shows that, although the majority are factoid questions requiring single short answers, there are also list questions and one definition question (labelled as type 'OTHER'; Note that the question types are provided.) List questions are essentially the same as factoid questions in that they require short succinct factual answers to factual question where the difference is that a list question has multiple correct answers, all of which must be retrieved from the corpus and "duplicates" recognised and eliminated to be absolutely correct. This requires additional procedures. Definition questions on the other hand are not factoid question but require the retrieval of text nuggets consisting of essential facts about the topic in question. Definition and list questions are areas of interest on their own, but in this thesis, I only concentrate on single factoid questions. (Chapter 7 gives reasons for not doing list or other questions at present.)

Because this type of QA relies on a large text corpus, it is also called *Data-Intensive Approach to Question Answering* (Brill *et al.*, 2001). There are two potential advantages to this approach: First, a large corpus would contain a lot of facts of interest. These facts can be directly exploited by textual Question Answering systems without having to manually construct extensive and thus expensive database as in the database oriented Question Answering. Considering that a lot of electronic text remains "raw" as in the World Wide Web, this approach to Question Answering is the most economical way of exploiting the textual resources for wide-coverage QA in particular. Second, if the corpus is large enough, the same fact would be expressed in different ways (called *the redundancy of facts*) so as to be able to easily locate the relevant text from the question. This avoids the need for too much sophisticated NLP or inference techniques, which are often brittle (see for example Clarke *et al.* (2001) and also Brill *et al.* (2001)). This renders the QA operation essentially that of finding and locating the answer as opposed to generating one, in other words, an IR task. The thesis follows this approach and aims to make QA even more of an IR task by enabling direct answer retrieval.

2.3 General QA Architecture

TREC QA Evaluation exercises have been showing the latest in the Question Answering systems over the past few years. Although every system is different, the basic processes involved and the pipelining of these processes are basically the same across most of the systems (Voorhees, 2005). The typical *QA architecture* consists of the following four main processes, which sequentially make up the question answering pipeline: *Question Analysis*, *Information Retrieval*, *Answer Extraction* and *Answer Selection (Validation and Re-ranking)*. In addition to these on-line processes, there is another important process that is performed off-line, namely the preprocessing. Together, a diagram of the architecture is illustrated in Figure 2.1. It has to be emphasised that what is common to the QA systems is the generic pipeline; what the individual systems do in each stage (especially post IR) of the pipeline is what differentiate the systems.

1. **Preprocessing**

   The source of knowledge for textual Question Answering comes from a collection of raw text documents (the corpus). Often, some extraneous materials such as HTML tags must be removed if they are not relevant. It might be necessary to have the texts tokenised and stored if parsing is required. Also a decision has to be taken as to what meta-information to include if any. Finally the text collection must be indexed, often using widely available off-the-shelf IR tools such as MG (Witten *et al.*, 1999) or Lemur (Ogilvie and Callan, 2002). Also for indexing, there are some decisions to be made pertaining to the granularity of the entities being indexed (sentences, passages or documents), what terms to index on (to include stop words or not), whether to stem or to lemmatise, whether to index the positions of the terms, etc. In general, the more that is done here the more time will be saved later on.

2. **Question Analysis**

   When a question is put to a system, it needs to know what type of answer the question is asking about. For this purpose, often a question type ontology (e.g. Hermjakob (2001), Li and Roth (2002) and Zhang and Lee (2003)) and a question classifier are used to classify the question. Based on the type of the question identified, a query is formulated for information retrieval. Extra-knowledge

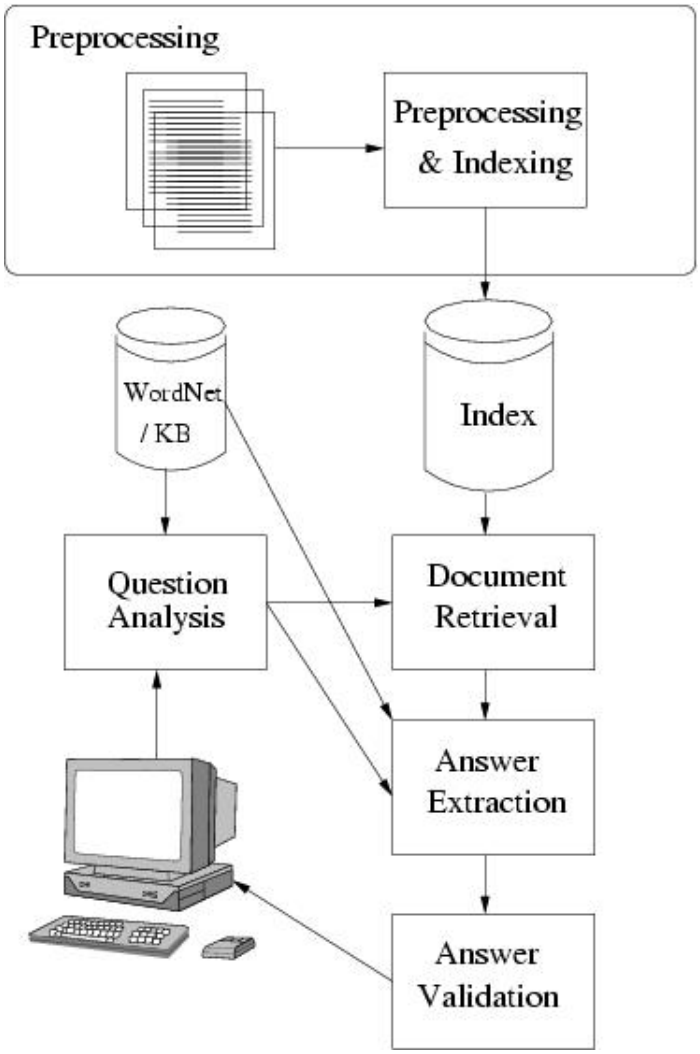

Figure 2.1: The General Architecture of Typical Question Answering System

sources are often employed here for the purpose of query expansion, e.g. WordNet (Miller, 1995). Question analysis also plays a vital role in the answer extraction stage; some systems utilise answer patterns identified by the corresponding question types (Banko *et al.*, 2002), while other systems such as Ahn *et al.* (2004) produce a DRS representation from the question for more elaborate answer generation.

3. **Information Retrieval**

   Having analysed the question and formulated a query, relevant parts of the corpus are fetched by the retrieval unit. Off-the-shelf IR tools are commonly used for that purpose. Various techniques may be applied to isolate the portion of the text likely to contain answers, e.g. passage based retrieval (Callan, 1994) or minimal

span weighting based retrieval (Monz, 2003b). When the retrieved results are documents, a further process takes place such as tiling as in Leidner *et al.* (2003) to segment the texts and thus further narrow down the relevant parts. Also the ranking of the retrieved results are important. One can either use the default ranking produced by the IR or re-rank according to some other criteria.

4. **Answer Extraction**

   Answer extraction is the core of most of the Question Answering systems. While interesting things could be done in the prior/posterior stages, this is where most systems differ. There are roughly two approaches: *Deep* and *Shallow*. A deep approach as used in QED system (Leidner *et al.*, 2003) for example relies heavily on semantic analysis and uses logical reasoning to come up with the answer. The LCC system (Harabagiu *et al.*, 2003) uses abductive reasoning to infer the answers. This approach is very time consuming yet the deeper analysis sometimes affords more sophisticated analysis of the evidence. A shallow approach on the other hand relies primarily on pattern matching. For example, the AskMSR system (Banko *et al.*, 2002) uses pre-stored answer patterns based on morpho-syntactic information generated by machine learning. This approach is relatively fast and also effective if it relies on a very large data source such as the World Wide Web (Soricut and Brill, 2004).

5. **Answer Selection/Reranking**

   More often than not, more than one answer candidate are produced and it is necessary to assess and rank them in the order of plausibility. Conversely, sometimes there is no answer for a question within the pool of the answer candidates due to the lack of any supporting material in the knowledge source although a system might produce answer candidates nevertheless. Hence, it is necessary to have some operations here, which are commonly called answer validation and reranking. Web validation is one technique that does this using the extensive data redundancy of the Web to confirm or reject an answer, e.g. (Magnini *et al.*, 2002b).

2.4 Summary

In this chapter, I have succinctly surveyed the field of Question Answering with respect to the history and some of the current developments with respect to TREC. I have also

discussed the common architecture of many QA systems. The purpose is not only to introduce the subject area that the thesis deals with but also to provide the core motivation for my own work via exposing some of the limitations of this architecture with respect to its use of Information Retrieval. That is the subject of the next chapter.

Chapter 3

Related Work

This chapter gives an overview of the use of Information Retrieval for Question Answering (commonly called *IRQA* or *IR4QA*). This overview provides the core motivation of the thesis and places the thesis' own approach within context.

3.1 Introduction

Chapter 2 describes the common core architecture shared by many QA systems relying on textual corpus as the source of knowledge. This architecture is characterised by the two main processes, *Information Retrieval* that retrieves a set of documents, and *Answer Extraction* that identifies and extracts candidate answer phrases from the documents returned by IR. This architecture is referred as *IR+AE* architecture, due to the prominent roles and the ordering of these two main processes.

As the name "IR+AE" implies, Information Retrieval plays a very important role in this architecture: without successful Information Retrieval, there is no material for answer extraction. The reason that Information Retrieval is needed at all is due to the fact that the document collection is too big to vet in its entirety. Information Retrieval in QA produces a subset of the collection in which answers are more likely to be found. More sophisticated and time-consuming processes of Answer Extraction then operate on this reduced corpus to locate and extract candidate answers. Thus, unlike in document retrieval, information retrieval for QA is not an end in itself but only the means for eventual answer extraction. This is referred to as IR having a *pre-fetch* role in Question Answering (Monz, 2003a).

Before discussing the specific approaches to IR for QA, it is important to establish what the basic requirements are. As the primary goal of pre-fetch is to lessen the

burden of Answer Extraction, the first requirement is to be able to retrieve the most relevant material from the source corpus related to the question (text that contains the answer with supporting evidence) within the least amount of text. Naturally, the less able IR is at pin-pointing relevant material in the corpus, the more text has to be retrieved, thereby increasing the work load for Answer Extraction. Thus, the ability of IR to retrieve highly relevant material with the minimum of text is very important.

Another requirement of IR4QA is efficency in the speed of retrieval. IR is generally very efficient in this regard, especially for general document retrieval. Any improvement in retrieval effectiveness must not negatively affect speed efficiency because IR is meant to speed up the overall process, and not add to the burden later on.

In the rest of the chapter, I will review the various approaches to information retrieval for Question Answering starting with the most common: plain document retrieval. Some of the advantages and limitations of this approach are pointed out. The limitations, in particular, provide the motivation for this thesis.

There are works that go beyond plain document retrieval with QA specific customisation for IR. I give an overview of some of these works as they relate to the thesis' own approach. These include adjusting the granularity of retrieval (passage retrieval and answer retrieval), customising the index (predictive annotation), and combining different evidence. These works provide the background to the core method of this thesis to be introduced in the next chapter.

3.2 Document Retrieval for Question Answering

Plain Document Retrieval for QA is the use of general purpose document retrieval in QA without any special customisation. This is basically treating IR for QA as an ad-hoc document retrieval, where a ranked list of documents are retrieved based on keywords provided by users.

While document retrieval can be regarded as the most basic form of IR for QA, it is widely used in many QA systems because they use an off-the-shelf document retrieval engine such as Lemur, Lucene, Okapi or MG, without modication. The main advantage to this is fast retrieval, as most off-the-shelf document retrieval systems are highly optimised for this task. Another advantage is that there is a minimum fuss due to the out-of-box usage provided by many of these engines.

In doing document retrieval for QA, the first task is to construct the query. This is done by extracting keywords from the question. Query expansion often takes place

using WordNet or other resources where synonyms of the keywords are included in the query in order to improve the recall. Using the query so constructed, a ranked list of documents is retrieved, which are then tiled into smaller passages in order to further narrow down the relevant parts of the text for answer extraction.

The retrieved documents must contain the answer phrases for a question for the QA system to be successful. Otherwise everything else that follows the document IR fails because nothing can be done about the lack of answers in the base material by the subsequent operations, no matter how good and sophisticated they may be. To prevent such situation, typically hundreds of documents need to be retrieved (Saggion *et al.*, 2004). The main catch is that processing so much text consumes valuable on-line time (the time the user is waiting for the answer after having put the question). This is why QA systems that rely on plain document retrieval cannot be easily scaled up to the demands of the real life application. Web search engines such as Google can handle tens of thousands of hits per second and fetch relevant information from a corpus consisting of billions of web pages (Brin and Page, 1998). This is possible because the list of documents (web pages) retrieved do not need to be further processed[1]. However, this is not the case for QA. The returned documents need to be investigated, and more sophisticated and time consuming operations of locating the answer have to be performed. This scaling problem, in addition to the accuracy problem, will become far more acute if a QA system is to be used in the scale common to the popular web search engines of today, a consideration that does not get much attention currently but will become an important issue in future.

There is a second, somewhat more subtle, problem related to the current use of document retrieval, which has to do with how the returned documents are regarded as evidence by the conventional QA systems. Most Question Answering systems produce an answer to a question by locating a passage or a document and extracting the relevant part as an answer as judged by some similarity this passage bears to the original question. Hence, it is assumed, and indeed required, that the evidence for an answer is localised within the same piece of text as the answer. However, for factoid questions whose answers depend on evidence from different text spans distributed in different places in the corpus, answers can not be extracted without some extra work-around method, which I will discuss in Chapter 6.

These two problems, *the lack of scaling potential* and *the locality of evidence con-*

[1] i.e. apart from the snippet generation process, which is much simpler than Answer Extraction process.

straint, provide the motivation for the approach taken by this thesis.

3.3 Customising IR for QA

Customising Information Retrieval for QA is the key to improving the performance of IR for QA and can possibly contribute to overcoming the two limitations mentioned in the previous section. There are works focusing on different aspects of customisation. In this section, some of these works are examined as they relate to the thesis' own approach.

3.3.1 Retrieval Granularity

The first problem, the limited scaling potential, is related to the issue of how much text IR needs to retrieve.

The discussion of plain document retrieval in the previous section only related to retrieving whole documents. An extension of the document retrieval is passage retrieval. While whole document retrieval retrieves a list of documents, often the text that provides the evidence for an answer is more localised, being a small part of a whole document in the range of one to three sentences (passages). Thus, the documents retrieved by IR need to be partitioned into smaller passages before the answer extraction takes place. This tiling operation takes up additional on-line time.

Passage retrieval is an approach where this tiling operation in effect takes place beforehand during preprocessing. Producing passages from whole documents and indexing these passages rather than the whole documents enable the direct retrieval of passages, saving the extra step of tiling. There are different ways of producing passages from the corpus. Tiedemann and Mur (2008) report that the best performing passage retrieval is by partitioning documents into simple non-overlapping windows of sentences rather than either by sliding windows (with overlapping sentences) or by liniking up sentences by co-referential chain.

Since passages are smaller than whole documents, the amount of text retrieved by passage retrieval is less than that by the whole document retrieval. One problem of passage retrieval is, however, that the retrieval effectiveness is not as good as whole document retrieval. According to Roberts and Gaizauskas (2004), doing whole document retrieval and tiling of documents into passage after the retrieval gives better performance than the passage retrieval. The likely reason behind this is that, since

passages contain less amount of text than whole documents, a passage may miss the multiple expressions expressing the same idea in a whole document (e.g. “the French president” and “the head of French government” within the same document). This disadvantages the retrieval because when the question contains an expression that differs from that of the potential supporting text (e.g. ”Which French president was a resistance fighter?” and “the current head of French government, Mitterrand, was a resistance fighter ..”), the retrieval does not succeed.[2]

One notable form of passage retrieval is single sentence retrieval. According to White and Sutcliffe (2004), the majority of the questions they have investigated, 44 out of 50 questions from TREC03 question set, could be answered unambiguously by single sentences as support implying that single sentence retrieval could be effective. Murdock and Croft (2005) confirms this by reporting that the single sentence retrieval can perform comparable to passage retrieval of a larger size. Thus, it can be said that sentence level passage retrieval is more economical than passage retrieval of a larger size without sacrificing retrieval performance.

An even more fine-grained approach to QA is answer retrieval. Fleischman *et al.* (2002) retrieve answers directly, but not via the conventional information retrieval. In what they call *the answer repository approach to Question Answering*, highly precise relational information is extracted from the text collection using information extraction techniques based on part of speech patterns. The extracted concept-instance pairs of person name-title such as “(Bill Gates, Chairman of Microsoft)” are then stored in database for fast retrieval. These are used either solely or in conjunction with a common QA system in producing answers. Using it as part of a more conventional Question Answering system, TextMap (Hermjakob *et al.*, 2002), the method resulted in a significant improvement (25 percent) in correctness of the answers and also significant improvement in speed, from 9 hours per 100 questions down to 10 seconds for compatible questions as compared to when TextMap alone was used. A similar work, Jijkoun *et al.* (2004) extract name-title relations off-line, and use them in answering questions within a multi-stream QA system (Jijkoun and de Rijke, 2004). “multi-stream” is the key here. Previously extracted relations are not sufficient for all of question answering, just for specific questions such as for name-title relations (Who is the chairman of Microsoft?). As useful as this is, this approach is best served as a complimentary part to an ordinary IR based QA system in so far as the wide-coverage QA is concerned:

[2]The linguistic gap between the question and the supporting text is what makes QA difficult in general.

This approach only deals with specifically anticipated question type, but for any other types, a conventional QA system must be used as fall-back.

In general, the smaller the granularity of retrieval, less text is retrieved and thus there is less material for answer extraction. In case of direct answer retrieval, answer extraction can be by-passed altogether, potentially speeding up the whole QA process. As one of the two goals in this thesis is to enable fast scalable Question Answering, this thesis also explores direct answer retrieval. Works such as Fleischman *et al.* (2002) show that direct answer retrieval is possible. However, methods based on information extraction has their limits with respect to the types of questions they can handle. The thesis will explore ways to make direct answer retrieval as efficient as that based on information extraction but that can handle more types of questions.

3.3.2 QA-tailored Indexing

Adjusting the granularity of retrieval as I have just discussed may contribute to the problem of limited scaling potential. A more fundamental approach is to construct indices that are specifically tailored to QA.

Queries in IRQA differ in a significant way from queries in ordinary document retrieval, in that the focus of IRQA queries (which corresponds to the WH-phrase of the question) is not known and therefore is absent from the query. Hence documents returned from the search process might contain a lot of things about the keywords contained in the query but nothing about the answer. For example, given the question, "Which religion did Confucianism replace in Korea?", Google highly ranks documents about Confucianism in Korea, Confucianism as religion, and the replacement of Confucianism in Korea (due to modernisation). A document about Buddhism, which Confucianism replaced and which is the answer is not found until rank 30 because Google IR system looks for documents (web pages) with high frequencies of keywords from the query but not for what is missing (e.g. Buddhism). (Due to the fact that answer terms frequently co-occur with terms of the question in a document, an answer bearing document may be the accidental result of document search.)

An approach that takes into account this consideration is named entity based indexing. Predictive annotation (Prager *et al.*, 1999) is the original work that set the precedence in this area. In this work, the text of the target corpus is pre-processed in such a way that phrases that are potential answers are marked and annotated with

respect to their answer types (or QA-tokens as they call them) including PERSON$[3], DURATION$, etc. Then the text is indexed not only with ordinary terms but also with these QA-tokens as indexing elements. The advantage of this approach is two-fold. First, by having done named entity recognition off-line, there is significantly less work that needs to be done on-line. The second advantage is the incorporation of QA-token as part of the index and hence a more QA tailored query formulation; the 'missing' keyword (the WH-word) is denoted by a QA-token of the identified answer type and treated as part of the query in the manner of essentially a wild card. This wild card is constrained semantically with respect to its named-entity type. For example, the question "Who is the president of France?" would produce a query that not only contains the key terms 'president' and 'France' but also a QA-token 'PERSON$'. The retrieved text based on this query would contain, in addition to the terms 'president' and 'France', at least one term that satisfies this specific QA-token, namely PERSON$. This technique improves the retrieval performance for passages.

Kim *et al.* (2001) is a direct application of the predictive annotation method but with the difference that the objects of retrieval are answer candidates as in Fleischman *et al.* (2002). Kim *et al.* (2001) have built a Korean language QA system that can answer factoid questions based on the potential answers (named entities plus email addresses, home-pages and telephone numbers) identified from a text source (web pages in html) through predictive annotation technique. For every potential answer, a set of 14 features such as apposition, POS and grammatical roles are extracted, and form the entries of a special index (a set of 14 databases). These features make up the parameters for a scoring formula that scores answer candidates with respect to a question. They report good performance (accuracy of 0.44 for 50 custom made questions), but the general complexity of the indexing and scoring scheme makes it rather unclear from their paper as to how they obtained the scoring formula and how the different parameters affect the overall performance. However, this work shows that direct retrieval of answers using what is essentially an IR technique is possible.

In Bouma *et al.* (2005), the index is more extensively customised with respect to what they call linguistically informed IR. The target corpus is given a full syntactic analysis in order to exploit linguistic information as a knowledge source for IR. The document collection along various linguistic dimensions, such as part of speech tags, named entity classes, and dependency relations are indexed with respect to paragraphs. With this index, the retrieval is not only performed on keywords, but with respect to

[3]In their notation, the Dollar sign at the end indicates that this is a QA token rather than a term.

these features. Although no information is given on the IR performance itself, the overall QA system that incorporates this IR component shows a good overall performance. As with Kim *et al.* (2001), it is not very clear how the different features (or linguistic layers as they call them) contributed to the performance of different question types and by how much. The complexity of the indexing scheme makes the analysis relatively difficult, and an extensive optimisation seems to be required in order to get the best balance of the parameters for the scoring formula. Also, there is one performance aspect that is not mentioned in this paper: The speed of retrieval. Information Retrieval of such complexity may well be much slower than ordinary document retrieval, a point that is tested in Chapter 5.

In summary, the predictive annotation technique is the primary basis for customising index for QA, which the core method of this thesis also incorporates. A more linguistically informed approach takes into account, in addition to the named entity types, POS information, grammatical roles, dependency relations etc. These information are put into index for either more precise passage retrieval (Bouma *et al.*, 2005) or even direct answer retrieval (Kim *et al.*, 2001). However, the more complex the index, the harder it is to understand how the successful scoring formula obtains or how the different parameters affect different types of questions. In addition, the retrieval speed might suffer due to the complexity of the index. For ordinary document retrieval, on the other hand, the scoring formula is relatively well established. Thus, the thesis makes use of indexing technique that does not depart too much from that of ordinary document retrieval. The novelty of the core method of this thesis is rather in turning a potential answer into a document, and thereby making it possible to use document IR (with the proven efficiency) in directly retrieving the answers.

3.3.3 Combining Different Evidence

The works reviewed thus far focus on improving the standard retrieval effectiveness and therefore relate to the first problem of limited scaling potential. Here in this subsection, some of the work related to the second problem, the limitation of locality constraint, is investigated.

The limitation of locality constraint arises from the fact that a particular question needs several pieces of evidence, which are not contained within the same source of knowledge at the QA system's disposal. (This notion is more fully explained in Chapter 6.) Such questions tend to be complex as they contain multiple constraints.

For example, Bouma *et al.* (2005) discusses 'Which' questions such as "Which ferry sank south-east of the island Uto?". They characterise this type of question as difficult to answer as the answer is constraint by more than one condition, namely that the answer has to be a ferry (or a specific type of ferry), and also it has to be associated with the fact of having sank in a particular location. The method resolves a particular answer candidate, which already satisfies the verbal constraint (e.g. "X sank south-east of the island Uto"), to the constrained imposed by the head noun of the WH-phrase (e.g. 'ferry') by simply looking it up on external knowledge sources, Euro-WordNet (http://www.illc.uva.nl/EuroWordNet/) and a specially constructed IS-A database in order to find out whether it is a ferry or some sub-type of it. This thesis follows a similar approach for these types of questions (also including "What is the X that ..") by making use of a very comprehensive external knowledge source based on Wikipedia and WordNet (explained in the next chapter).

More generally, complex questions not only involve simple head noun constraints (such as for "Which X" questions) but involve joining together different verbal evidence. The strategy proposed by Katz *et al.* (2005) for dealing with such questions is first decomposing a complex question into sub-questions, answering each sub-questions, and combining the answers of each sub-questions. The paper discusses the various levels of decomposition that an automatic process can pursue. The paper mentions that some of the automatic decomposition techniques discussed here have been incorporated into the publicly available *START* QA system (Katz *et al.*, 2002) although no formal evaluation has been performed yet. Once the answers to sub-questions have been produced, they need to be combined to produce the overall answer. Chu-Carroll *et al.* (2003) discusses a strategy where multiple sets (two sets) of answers produced by different QA agents can be combined to produce the overall winner.

Another work in open-domain Question Answering to deal with the issue of complex questions requiring evidence from different parts of the corpus is Saquete *et al.* (2004). This work addresses temporal questions, such as ones of the form "Who was $< X >$ when $< Y >$?", where $< X >$ is the description of a person and $< Y >$ is the description of an event. The problem is that there may be local evidence for answering the "Who was $< X >$?" question in a particular year, but not local evidence that also associates event description $< Y >$ within that year. As in Katz *et al.* (2005), such a complex question is decomposed into simpler sub-event sub-questions, first answering the one that provides temporal grounding (e.g., When did $< Y >$ occur?).

These works all involve decomposing a question into sub-questions, performing

IR+AE on each of the sub-questions and then combining the results. These works are of interest mainly because the thesis provides an alternative approach, where the IR retrieves the answer candidate based on pre-collated evidence for an answer candidate. In Chapter 6, a version of this sub-question strategy is implemented to serve as a baseline to the thesis' main method.

3.4 Conclusion

This chapter gives an overview of the use of Information Retrieval in Question Answering as related to the thesis main theme. Although document retrieval is the most common form of IR for QA, this use of IR results in significant efficiency problem limiting the scaling potential and also causing the locality constraint by not being able to deal with scattered evidence without any special work-around. To overcome these limitations are the core motivations of this thesis. To improve IR for QA, the customisation of IR for QA reviewed in this chapter involves adjusting the retrieval granularity, customising index and combining different evidence to deal with complex questions. Some of these works reviewed here provide the inspiration for the thesis' own approach.

Chapter 4

Topic Indexing and Retrieval for Question Answering

This chapter introduces the core idea of the thesis, central to which is the method of *Topic Indexing and Retrieval for Question Answering (TOQA)*. The idea is motivated by the problems of the use of document retrieval as pre-fetch in common QA systems as described in Section 3.2. After outlining the conceptual framework underlying Topic Indexing and Retrieval for QA in Section 4.1, the method is described in details with respect to (1) the off-line *preprocessing* (Section 4.2) and (2) the on-line topic retrieval operations (Section 4.3).

4.1 Introduction

4.1.1 Goals and Motivating Principles

The answers to many factoid questions are named entities. For example, "Who is the president of France?" has as its answer a name referring to a certain individual.

The basic idea of *Topic Indexing and Retrieval* for Question Answering is to extract the expressions of this kind off-line from a textual corpus as potential answers and process them in such a way that they can be directly retrieved as answers to questions. Thus, the primary goal of Topic Indexing and Retrieval for QA is to turn factoid Question Answering into fine-grained Information Retrieval, where answer candidates are directly retrieved instead of documents/passages. The primary claim here is that for simple named entity answers, this can make for fast and accurate retrieval.

One of the reasons for doing this is to leverage the proven efficiency of established

Information Retrieval techniques in retrieving relevant items of information from a very large amount of data in a very short time. The conventional QA pipeline, as pointed out in Section 3.2, does not take advantage of the efficiency of Information Retrieval because Document pre-fetch is followed by extensive on-line processing. In contrast, directly indexing possible answers (*topics*) at the time of preprocessing enables direct retrieval of answers without the need for an extensive and time-consuming post-processing operation.

In addition, as the process gathers and collates all the relevant textual evidence for a possible answer from all over the corpus, it becomes possible to answer a question based on *all* the evidence available in the corpus regardless of its locality. (This is discussed in Chapter 6.) The guiding principles that underlie the proposed method is summarised as follows:

- Shift as much work off-line as possible through preprocessing.
- During preprocessing, identify all possible answers (topics) and all the supporting material for every topic from the whole corpus, collate them and index them for efficient retrieval.
- On-line, use Information Retrieval techniques of proven efficiency to directly retrieve the answer candidates from so created index.

4.1.2 What are Topics?

Central here is the notion of *topics*, as Topic Indexing and Retrieval for QA regroups the content of a whole corpus around topics. An encyclopedia provides good examples of topics, with articles on notable persons, organisations, places, things, concepts, intellectual properties, sport teams, programming languages, treaties, etc. While topics themselves such as “Bill Clinton” or “France” are proper names, by virtue of being named entities, they also have types (classes/categories) such as those of PERSON, LOCATION, etc. Type information is very important because questions often require the answers to be of certain types.

An encyclopedia also shows that a topic is not merely a name but the locus of information. For example, an encyclopedia article on the topic “Albert Einstein” contains information about this topic. While an ordinary corpus consisting of documents is not an encyclopedia, it is still possible to identify topics and gather information about them, since topics are present in any meaningful discourse. For example, a book on

Roman history would contain an index at the end of the book, which contains a list of topics along with pointers as to where the relevant information about them could be found. Such information can be called the *content of a topic*. As compared to the content of a document (which is whatever text falls within the span of the document), the content of a topic has to be found within the document collection. As there is no clear delimitation for the content of a topic, one can simply take a statement that mentions a topic as being related to it and hence a part of its content.

The content of a topic is important for Question Answering because this content, in effect, provides all the evidence available for that particular topic to be an answer to a given question. If the content does not contain the material that supports this topic as an answer, then the topic cannot be an answer to this question. Therefore, I will also call the content of a topic the *textual evidence* for the topic. How these topics, their types and content, relate to actual Question Answering is the subject to be discussed next.

4.1.3 Question Answering with Topics

The underlying idea of topic oriented Question Answering can be described in terms of sets. Hamblin (1958) proposed that the semantic meaning of a question is the set of all possible answers. According to this view, for example, the meaning of the question "Which US presidents were Catholic?" is the set of all entities that are answers to this question, i.e. $\{JFK\}$. Taking this idea literally, the set of all topics pre-gathered from a corpus can be regarded as the set of all possible answers under this corpus for all possible factoid questions that can be answered, or in other words, the *domain of answers*. Then answering a particular question amounts to essentially selecting a subset from this domain that constitutes its answers.

A question provides various kinds of information by which the domain can be reduced to a small subset of answer candidates containing the correct answer to this question. For example, a question has an *answer type*, which, for a proper name question, is the named entity type that the correct answer must match. For example, "Who is the president of India?" requires the answer to be of type person. The answer type induces a subset from the base domain of all topics because among all the entities in the domain, only those that have the right named entity type can be the answers. Also, a question may contain a proper name, which I call the *question topic*. In the above example question, "India" is the question topic. Not all questions contain question top-

ics, for example, “Who is the richest person in the world?”. However, when present, question topics imposes the constraint that the answer topics should not be one of the question topics since the answer is something that is not known already. (Trick questions are an exception: For example, the answer to the question “Who did Bill Gates play in the short film presented at the CES Las Vegas before his keynote speech?” is “himself”, i.e. Bill Gates. Hence this constraint is not a logical necessity but more of a general tendency.) A question topic also requires the answer to be in certain relationship to it. The textual content of the question imposes another requirement that a possible answer must satisfy. Since a topic has textual content, it provides the evidence for it to be an answer to a question. Some part of the textual content of the topic must match the textual content of the question. Unlike the constraints imposed by the answer type and the question topics, the required match between the textual content of the question and that of a topic can be more fuzzy and graded as there could be different degrees and forms of match. In fact, a scoring function giving the similarity between the textual content of a question and that of a topic, is the practical means to measure this match in textual QA. More complex relations between the two are also possible via logical and/or textual entailment. However, that is not within the scope of research for this thesis. So based on the various information (constraints) provided by questions, the overall QA strategy for topic based QA is as follows: From the domain of all topics, a subset of answer candidate is identified by the constraints imposed by the question topics and the answer type, and from this subset, each candidate is examined and scored with respect to its textual content. The highest scoring candidate will be chosen as the answer to the question. Figure 4.1 illustrates this process of inducing the answer candidate subset from the domain. This conception of Question Answering is somewhat similar to Roth *et al.* (2002), which views the selection of the answer and its justification as an incremental constraint satisfaction process”. One example given in this work is “Who was the first woman killed in Vietnam War?”. In connection with this example, it is mentioned that due to the fact that the answer type here must be ‘woman’, a lot of noun phrases (that do not indicate women or at least persons) need not be considered as answer candidates. This is similar to conception of QA as answer domain reduction. The rest of the chapter discusses how to implement this conceptual framework into a working QA system.

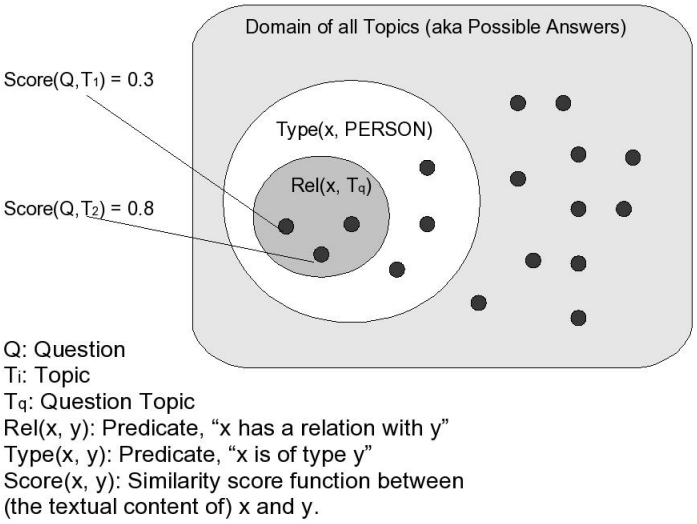

Figure 4.1: Answer Domain

4.2 Preprocessing

This section discusses the creation of the material that is the basis for the retrieval of topics as answers to questions. This is done off-line in preprocessing. The material that is created consists of (1) *Topic Repository*, which stores the variant names of topics and their named entity types, (2) a set of indices created by indexing the content for these topics via what I call *topic documents*, and (3) the topic document collection itself. This section describes the detailed operations involved in creating them and their uses for question answering.

4.2.1 The Make Up of Topic Repository

Topic Repository is where the variant names associated to topics and their named entity types are stored for efficient look-up. For fast access of the data, hash tables are used as the primary medium of storage since they support very fast look-up of simple (key,value) pairs. Unlike a full database, a hash table only supports two fields per entry, but it is significantly more efficient.

In order to build a topic repository from a corpus of text, the first order of business is to identify topics. Since topics are named entities, they are proper noun phrases (proper names), e.g. "Bill Clinton". Recognising proper names requires the target text being POS tagged and chunked. A named entity recogniser, C&C NER (Curran and Clark, 2003), is then run on the target text in order to identify and extract topics. This process

also identifies the types of topic, PERSON, LOCATION and ORGANISATION for the topics, which are later used as the base types to build separate indices corresponding to each topic type, which is explained in Section 4.2.2

After recognising a named entity, the next question is whether this particular proper name represents a new topic or is an instance of a topic that has already been identified and stored. There are two problems that complicate this process. One is the problem of proper name resolution, which arises because the same entity (a topic) can be expressed by multiple different names, e.g. "Bill Clinton" and "William Clinton." The other problem is the problem of ambiguity: One name can refer to multiple entities of the same or different types, e.g. "George Bush" (where there are two, senior and junior), and "Roaring Forties" (which is the name of a wind, a cheese, and a vineyard in Tasmania.)

The problem of ambiguity is not dealt with at the moment, and the consequences of this decision and the future work relevant to handling it are discussed in Chapter 7. The first problem is dealt with (to some degree) using simple heuristics and a knowledge base derived from Wikipedia (http://en.wikipedia.org). Wikipedia is an on-line encyclopedia that is edited by ordinary people all over the world. In addition to more than one million topics, it also has extensive list of alternative names for many topics in its *redirect table*. This redirect table is downloadable (along with the entire encyclopedia articles) from the web in the form of database dump, which can be directly imported into MySQL (or other similar) database tables. The redirect table contains all the variant names as keys, whose values are canonical names, with respect to which the variant names are alternative expressions. Another table contains the unique id number for each entity referred by a canonical name. So linking up these tables, I have created a topic-name hash table, using which it is possible to simply look up which topic a name refers to, and thereby to resolve it to a unique topic. For example, the topic-name hash table contains an entry like ('George Clooney', 1745442) where the name 'George Clooney' can be used to look up the unique topic id 1745442.

Also, in order to store the specific named entity-type of a topic, a separate topic-type hash table is created. Since topics are named entities, each belongs to some specific named entity type (ontological category / class). This type information is essential in judging a topic as an answer to a question because the question demands its answer to be of some specific type. For example, most questions starting with 'Where' demands the answer to be of location type (or more specifically any of its subtypes such as country, city, etc.). Having identified and stored the type information for each topic

(through means to be discussed next), this type information of a candidate topic can be matched up to the answer type demanded by the question. The base types (PERSON, LOCATION and ORGANISATION) are already identified by the Named Entity Recogniser used for the identification of topics as have mentioned previously. However, more fine-grained types are needed for a more precise match. In order to find out the fine type of a topic, an external ontology database, Yago (Suchanek *et al.*, 2007) is used. Yago contains an extensive data of fine types for the Wikipedia topics. It is derived by mapping the category information about target topic supplied by a Wikipedia user to the appropriate WordNet concept. (Wikipedia categories are not consistent and uniform, and they are more like tags that characterise a topic rather than strictly classify it.) Using this Yago ontology to look up each topic-type (i.e. the corresponding WordNet concept) and by tracing up the WordNet concept hierarchy, I created a very fine grained and multi-level (w.r.t ISA) topic-type database for all the topics in the topic repository. The following is an excerpt of an actual entry in the table for "Albert Einstein". (The original entry is even longer.) In this excerpt, types marked with a '@' indicate original Wikipedia categories whereas '::' indicates the inverse of a WordNet ISA relation:

```
person_100007846
   ::scientist_110560637
      ::physicist_110428004
         @Physicists
         @German_physicists
         @Swiss_physicists
person_100007846
   ::traveller_109629752
      ::absentee_109757653
         ::exile_110071332
            ::refugee_110516016
               @German_refugees
person_100007846
   ::intellectual_109621545
      ::scholar_110557854
         ::humanist_110191192
            @Humanists
person_100007846
   ::scientist_110560637
      @Jewish-American_scientists
      @Jewish_scientists
person_100007846
   ::native_109620794
      @Natives_of_Baden-Wuerttemberg
```

```
person_100007846
   ::national_109625401
      ::citizen_109923673
         @Naturalized_citizens_of_the_United_States
person_100007846
   ::acquirer_109764201
      ::recipient_109627906
         ::honoree_110183757
            ::laureate_110249011
               @Nobel_laureates_in_Physics
person_100007846
   ::adult_109605289
      ::pacifist_110390199
         @Pacifists
```

As this example shows, this hash table not only contains detailed type information pertaining to a topic, but it also contains a significant amount of knowledge typically associated to the topic due to the nature of Wikipedia categories being similar to descriptive tags. For example, here 'refugee' would not be a named-entity type of the topic, 'Einstein', per se (and would never be identified by a Name-Entity Recogniser as a type no matter how fine-grained), but nevertheless it is entered into the table due to the topic being tagged as 'German refugees' in Wikipedia. Other such cases include 'Bill Gates' being 'CEO' (a title-role), 'GM' being 'automobile manufacturer' (a description) and 'Pusan' being 'a province of Korea' (a geographical knowledge). Such diverse and significant amount of knowledge as well as the breadth and depth of the fine types contained in the topic-type hash table enable a very powerful match between the answer type from the question to that of a candidate topic.

In summary, the topic repository contains two hash tables with which the various names of a topic and its type can be looked up. For example, given the name 'Albert Einstein', using the topic-name hash table, it can be looked up to be of topic id 2983371. By using the topic-type table, topic ID 2983371 is identified as type person and scientist (etc). The topic repository (along with an example topic document that is explained in the next subsection) is illustrated in Figure 4.2.

4.2.2 Creating and Indexing Topic Document Collection

In Section 4.1.2, I have mentioned that topics have information associated with them, which I call the *content* of a topic. I take the content of a topic to be the set of all sentences that mention it. Since anaphora resolution is not yet performed, this misses

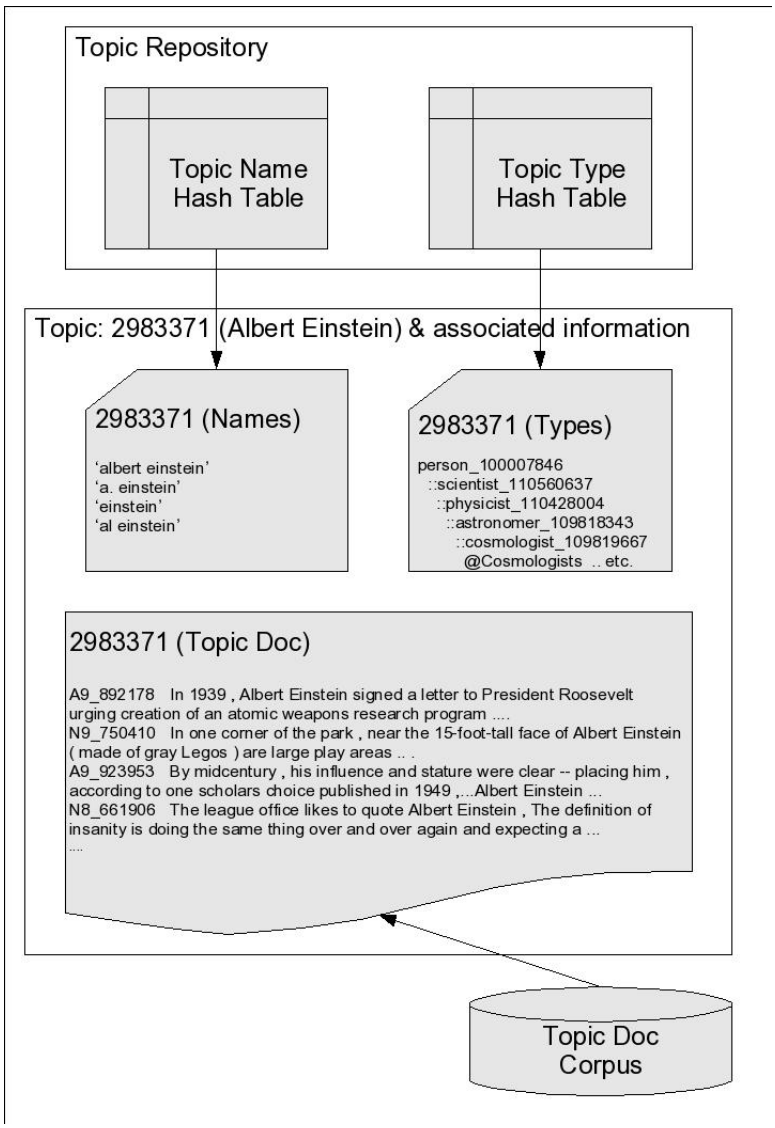

Figure 4.2: Topic Repository for the topic, 'Albert Einstein'

the sentences that mention a particular topic only anaphorically. This point is discussed further in Chapter 7 under Future Work. With the set of all sentences that mention a particular topic assembled into one physical location (a file), this can literally be regarded as a document on its own with the topic name as its title. I henceforth call such a document, a *topic document*.

The process of generating the collection of topic documents for all identified topics is shown schematically in Figure 4.3. The starting point is a corpus of original documents. Each document in the corpus is POS tagged, chunked and annotated by NER. Each sentence in each document is then examined as to whether it contains at least one proper name. If so, then whether this proper name represents a topic already found

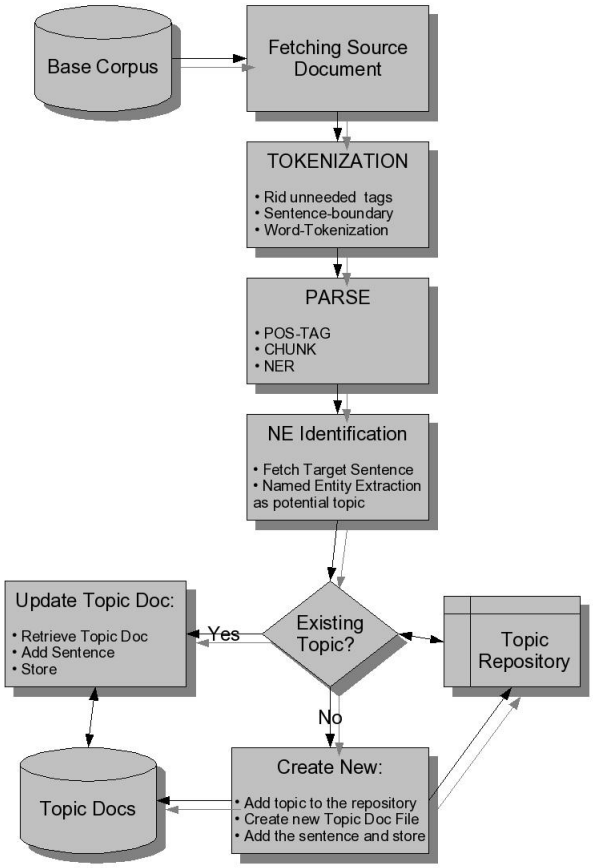

Figure 4.3: Building Topic Document Collection

in the topic repository is examined. If so, then this sentence is appended to the topic document that matches the topic. If not, then a new topic is instantiated and added to the topic repository and a new corresponding topic document is created with this sentence. If more than one proper name is identified in a sentence, then the same process is applied to every one of them. (See Figure 4.4 as an illustration of this process)

A topic document that is created out of this process is not a real document in the proper sense, e.g. like having cohesive discourse structure/story, intelligible to humans as a whole, etc, but a 'quasi' document whose sole purpose is to provide the evidence for Question Answering with respect to this topic as mentioned in Section 4.1.2. Hence, it is not important whether a human can read a topic document and make any sense out of. What is important is that the topic document provides enough context about the topic so that it can provide relevant evidence for a question. The original location of each sentence in a topic document is also stored with a pointer. This pointer at present doesn't serve any purpose but it could be potentially useful. This point will

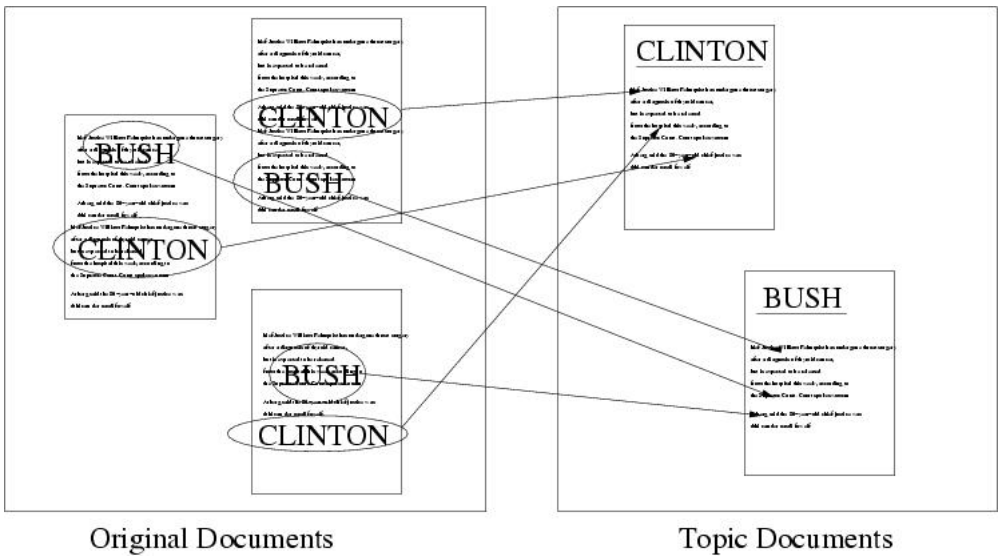

Figure 4.4: Topical Reorganisation of the Original Corpus

be discussed under Future Work in Chapter 7. Figure 4.5 is an actual example of a topic document for the topic, 'Dolly the sheep'. The topic document collection created using the topic document method is the base material for Question Answering in the sense that the retrieval of a particular topic document admits a potential answer to a question. In order to facilitate the retrieval process, this topic document collection is indexed using an indexer. As noted, according to the base types of the topics identified from Named Entity Recognition, three separate indices corresponding to PERSON, LOCATION and ORGANISATION are created. In addition, an index for all topic documents regardless of types, *TOTAL*, is also created for questions from which the answer type cannot be determined or for which their answer types differ from the three basic types. An appropriate index is chosen depending on the answer type identified from the question at the time of retrieval, which is discussed in the next section. Of note here is that separate indices are created only for these base types as it is not desirable to create each and every index for each of the numerous fine types in the topic-type hash table. These fine-types are only used for reranking operation after the candidate topics have been retrieved from the base indices.

4.3 Topic Retrieval Operations

In the previous section, I described the processes of preparing the base material for topic retrieval. In this section, I describe the retrieval operations based on this material.

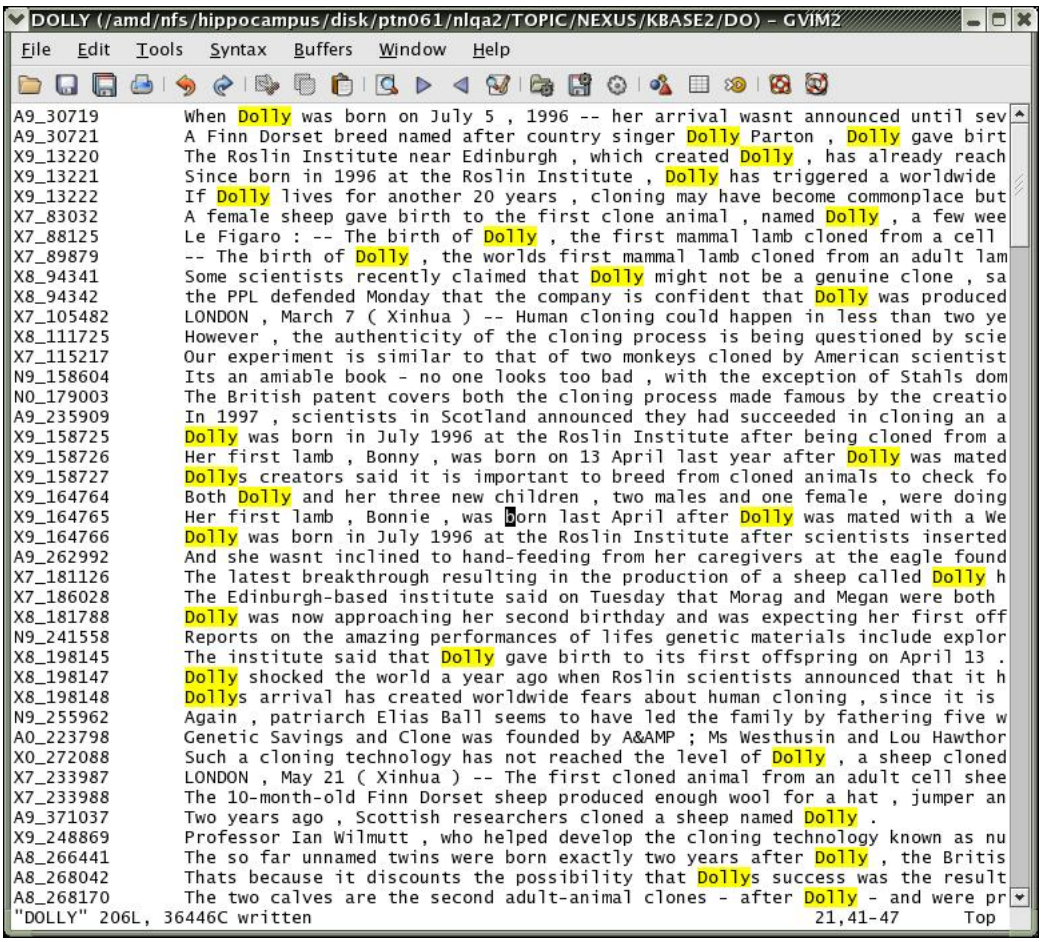

Figure 4.5: An Example Topic Document: Dolly the sheep

4.3.1 The QA Operations based on Topic Retrieval and Indexing

The conceptual framework of Question Answering based on Topic Indexing and Retrieval was introduced in Section 4.1.3: The QA operation on the domain of answers is an operation that "zeros in" to the ultimate candidate answer using the information contained in a question: the *answer type*, the *question topics* and the *textual content* of the question.

Within this framework, I have taken the efficiency to dictate the actual answer retrieval strategy to be used on this framework. Since the set of all topics consists of many entities (the method I described produces 251065 topics out of AQUAINT corpus), it is not practical to examine every entity in the set individually. The indices enable the examination of entities that only pertain to those that have some relevance with respect to the keywords contained in a particular question. Also the separate indices built for the different types of topics, i.e. PERSON, LOCATION, ORGANI-

SATION and TOTAL (for indeterminate or other types), serve to limit the search space by containing only those entities of a given type that match the broad answer type as identified from the question. This process is equivalent to subsetting the answer domain with respect to the answer types as described in Section 4.1.3. Indices are built only for the broad types because I found it impractical to build index for every fine-grained entity type. A post-retrieval operation is performed on the retrieved list of topics in order to further select the answers based on the finer-grained entity types using the topic repository. Question topics are also made use of during this stage. This is explained in Section 4.3.3.

The overall processes of topic retrieval for Question Answering is illustrated in Figure 4.6. The figure shows also some of the preprocessing elements including the topic repository consisting of topic-name hash table, topic-type hash table and the indices of topic documents. The actual retrieval processes (shown at the bottom of Figure 4.6) consist of analysis of user's question, retrieval of topics and post-retrieval operations including filtering question topics and reranking based on topic types.

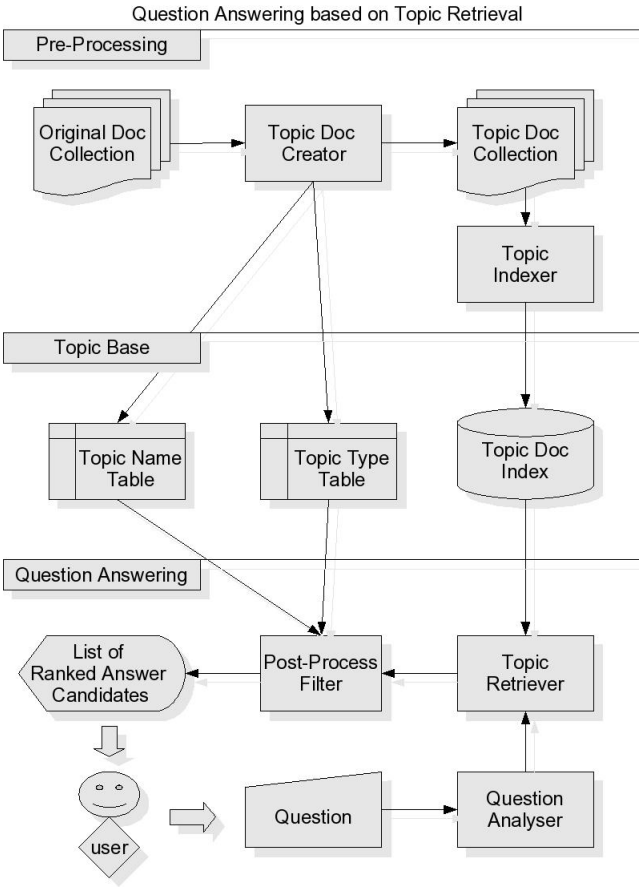

Figure 4.6: QA Processes based on Topic Indexing and Retrieval

The first operation is the *Question Analysis*:

1. Pos-tagging and Chunking
2. Question Type Identification
3. Answer Type Identification
4. Question Topic Identification

The purpose of question analysis is to identify the *question type* (such as definition question, factoid question, list question, etc.), the *answer type*, the *question topics* (if any) and to parse the question (only with respect to phrase boundaries) for query formulation. The *question topic identification* is straightforward: any proper name present in a particular question is a question topic. For *answer type* identification, I use a simple rule based algorithm that looks at the WH-word (e.g. "Where" means location), the head noun of a WH-phrase with "Which" or "What" (e.g. "Which president" means the answer type is of president), and if the main verb is a copula, the head of the post-copula noun phrase (e.g. for "Who is the president ..", here again "president" is the answer type.) WordNet is used to identify the base type of the answer type identified from the question when it is not one of the base types (PERSON, LOCATION, ORGANISATION). For example, "president" is traced to its base type, "PERSON". The following illustrates the output of the question analysis[1] :

```
Who is the president of Germany? -->
('FAC', 'president', ['germany'], [['is'], ['president', 'of','germany']])
```

The first item in the output indicates the question type. In this example, 'FAC' indicates that the question is factoid question (this is just a formality since the method only deals with factoid named entity questions). The second item is the answer type, which, for this question, is 'president'. The third item, '[germany]' is the list of the question topic(s) found in the question (here only one). Finally, the last item, [['is'], ['president', 'of', 'germany']], is the textual content of the question partitioned according to phrase boundaries as analysed by a shallow parser which is the basis for a query formulation.

The next operation is the retrieval of topics as answer candidates for a given question. This involves:

[1] NLProcessor from Infogistics (http://www.infogistics.com/textanalysis.html) was used as a part of the question analyser to shallow parse the question.

1. identifying the appropriate index,

2. formulating a query, and

3. retrieving topic documents.

An appropriate index is chosen based on the answer type. In the example query above, since the answer type, 'president', is not the base type, WordNet is used to trace from 'president' to a base type (person) and the corresponding index is selected. If none of the three base types is found by this process, the total index is used.

Subsequent operations involve query formulation from the parse generated from the question analysis, retrieval of topic documents and post-processing operations. These are described in details in Section 4.3.3.

4.3.2 Scoring Topic Document with the Inference Network Model

Using the index of the topic document collection, topic documents are retrieved using the query generated from the question under consideration.

There are different ways to score the documents with respect to a query depending on the model of retrieval used. The model of retrieval used for this method is the document inference network model invented by H. Turtle (H.R.Turtle, 1991) and implemented in InQuery retrieval system within the Lemur Tool Kit (Ogilvie and Callan, 2002). The InQuery search engine provides several advantages for the purpose of this work. In addition to the top state-of-the art retrieval performance as shown by TREC exercises, InQuery supports a powerful and flexible query language that can be exploited for the QA task at hand, the details of which is discusses in the next subsection.

Here I give a succinct overview of this model.[2] Inference network model for document retrieval is Bayesian networks. A Bayesian network (or a belief network) is a probabilistic model represented by a directed acyclic graph (DAG) where each node represents a random variable that is conditionally independent. Each directed arc represents the influence relation between two nodes connected by this arc; a node with an incoming arc is only directly influenced by its source node. The probability of a set of events (nodes) is calculated by multiplying the probabilities of all the parent nodes that lead up to these nodes.

[2]Throughout this subsection, all the formalism and equations used are from Baeza-Yates and Ribeiro-Neto (1999) unless otherwise noted.

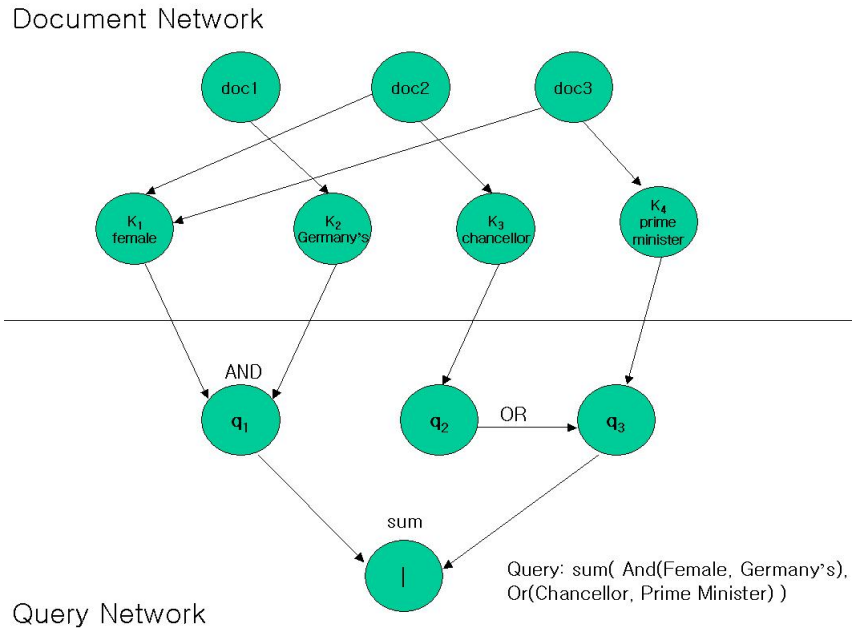

Figure 4.7: Document Inference Network

$$P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i|parents(X_i)) \tag{4.1}$$

As a form of a Bayesian network, the document inference network, in the basic form, associates binary random variables with document, index term, user query and finally the information need of the user as denoted by d, k, q and I respectively.[3] A binary random variable has only two values, 1 or 0 (True or False) according to whether the event denoted by the random variable is observed or not. In that sense, this variables can be regarded as propositions.

The document inference network consists of largely two parts.

The first part is the document network (the upper part of the graph in Figure 4.7). This network represents the document collection. This is built at the indexing time, and once it is built, it is not modified at the time of querying. In the document network, the nodes denoted by d (i.e. d_1, .., d_n) in Figure 4.7 are binary random variables associated with each document in the document collection. When a particular document is observed (i.e. when this particular document is being ranked and scored for retrieval), the value of this random variable is 1 and the probability associated to this random vari-

[3]When there is more than one random variable associated with the same type of item, e.g. a document, a random variable is denoted with an index such as d_1 to distinguish it with other instances of the same item type.

able, $P(d)$[4], is the prior probability of this document. An index term node, k, likewise represents a binary random variable, which has the value 1 if an index term associated to k is observed. The probability of this node is a conditional probability dependent upon document nodes that connect to it, as in Figure 4.7, representing the fact that the k being observed means that it is observed from a document (or documents). Hence, the probability of this node is the conditional probability $P(k|d)$. I denote the set of all index term nodes as a vector $\vec{k}$, which instantiates a vector space of 2^m dimension when the number of terms is m (since every index term node represents a random variable with only two values, 1 or 0).

Next, the query network consists of the nodes that represent the user's actual query, q, and the information need I. Whereas the random variable q represents the fact that a user has put a query, the node I is more complex. Several different queries can be combined to express the eventual information need and I is needed as a formal device to tie up these different queries. This query network is constructed at querying time for each and every query by a user and connects to the document network dynamically at real time.

The document inference network is used to score documents with respect to a query by a user; the highest scoring documents would be retrieved as the result of the querying. In inference network, the score of a particular document (d_j) with respect to a query is equivalent to the value of the probability that the query and the document are observed together, namely the joint probability $P(q \wedge d_j)$. This probability is computed by the following equation.

$$P(q \wedge d_j) = \sum_{\forall \vec{k}} P(q|d_j \times \vec{k}) \times P(d_j \times \vec{k}) = \sum_{\forall \vec{k}} P(q|\vec{k}) \times P(\vec{k}|d_j) \times P(d_j) \tag{4.2}$$

$$P(\neg(q \wedge d_j)) = 1 - P(q \wedge d_j) \tag{4.3}$$

Note that in the above equation, $P(q|d_i \times \vec{k_i}) = P(q|\vec{k_i})$ since the node d_j has no influence in the node q being d-separated by the node k_i. (See Figure 4.7) Another thing to note is that each node representing the index term random variable k_i is independent to other index term random variable in the inference network since they are d-separated by the q nodes in the upper layer. Thus the probability of the index term vector $\vec{k}$ with respect to a particular document d can be computed as the product of each index term node probability as in the following equation.

$$P(q \wedge d_j) \quad = \quad \sum_{\forall \vec{k}} P(q|\vec{k}) \times P(\vec{k}|d_j) \times P(d_j) \tag{4.4}$$

[4]For convenience sake, I will henceforth denote $P(d = 1)$ as just $P(d)$ and $P(d = 0)$ as $P(\neg d)$

$$= \sum_{\forall \vec{k}} P(q|\vec{k}) \times (\prod_{\forall_i | g_i(\vec{k})=1} P(k_i|d_j)) \times (\prod_{\forall_i | g_i(\vec{k})=0} P(\neg k_i|d_j)) \times P(d_j) \quad (4.5)$$

(4.6)

5

In order to actually calculate this probability, the component probabilities need to be known. First, the prior probability, $P(d_j)$, can be estimated as just $\frac{1}{N}$ where N indicates the number of total documents in the collection, which makes the simplifying but convenient assumption that every document in the collection is equally probable.

In calculating the other probabilities, the original inference network model uses TF-IDF based estimation. TF-IDF weighting scheme captures the intuition that the significance of a term with respect to a document is the frequency of the term observed in the document weighted by the intrinsic value of the term, which is determined by the inverse document frequency of this term over the whole collection.

First, the raw frequency (f) is the number of times that a term of interest, k_i, appears in a particular document, d_j. For the actual term frequency weight (TF), it is desirable to normalise this raw frequency against the number of all terms in the document (denoted as max_l) in order to discount the effect of document length. So TF is defined as:

$$TF_{i,j} = \frac{f_{i,j}}{max_l f_{l,j}} \quad (4.7)$$

Next, the inverse document frequency of the term, k_i, as denoted by IDF_i is a log ratio of the number of total documents, N, over the number of documents in which this term appears, n_i.

$$IDF_i = log \frac{N}{n_i} \quad (4.8)$$

Thus the TF-IDF weight of a term k_i with respect to a document d_j can be stated as the following:

$$w_{i,j} = TF_{i,j} \times IDF_i = \frac{f_{i,j}}{max_l f_{l,j}} \times log \frac{N}{n_i} \quad (4.9)$$

Now, the component probabilities in 4.4 are estimated as follows.

$$P(q|\vec{k}) = IDF_i \quad (4.10)$$

[5] g_i is a function that returns 1 if the ith component element of a vector that is the argument to this function is not 0, and returns 0 otherwise.

Operator	Action	Computation
not	Negation of a term	$bel_{not}(I) = 1 - P_1$
or	Disjunction of terms	$bel_{or}(I) = 1 - (1 - P_1)(1 - P_2)..(1 - P_n)$
and	Conjunction of terms	$bel_{and}(I) = P_1 P_2..P_n$
sum	Mean of the beliefs	$bel_{sum}(I) = \frac{(P_1 + P_2 + .. + P_n)}{n}$
wsum	Mean of the weighted beliefs	$bel_{wsum}(I) = (\frac{w_1 P_1 + w_2 P_2 + .. + w_n P_n}{w_1 + w_2 + .. + w_n})$
max	The maximum belief	$bel_{max}(I) = max(P_1 P_2..P_n)$

Table 4.1: Boolean Belief Operators supported by InQuery

$$P(k_i|d_j) = TF_{i,j} \tag{4.11}$$

$$P(\neg k_i|d_j) = 1 - P(k_i|d_j) = 1 - TF_{i,j} \tag{4.12}$$

So 4.4 is computed by plugging in the values for the component probabilities just described. The resulting equation (after some algebraic manipulation) is thus:

$$P(q \wedge d_j) = C_j \times \frac{1}{N} \times \sum_{\forall_i k_i = 1} TF_{i,j} \times IDF_i \times \frac{1}{1 - TF_{i,j}} \tag{4.13}$$

where $C_j = \prod_{\forall_i} P(\neg k_i|d_j)$.

The equation provides the way to score a document with respect to straightforward keywords query. However, the inference network model supports the integration of other elements other than pure keywords via the structured query syntax provided in Lemur: Simple queries can be combined into a more complex query.

So for a complex query that combines simple queries (straightforward bag of words queries) into 'and' and 'or' combinations, its belief probability with respect to a target document (d_j) can be computed, e.g. in case of Boolean 'and' via $p_I = p_1 \times p_2$ and for 'or' $p_I = 1 - (1 - p_1) \times (1 - p_2)$ where q_i denoted a simple query, and I denotes the final information need node. The Table 4.1[6] shows a list of Boolean operators supported by InQuery.

4.3.3 Topic Retrieval Operation

In addition to the Boolean Belief Operators outlined in the previous section, InQuery also supports proximity based operators using the position information stored at the

[6] from H.R.Turtle (1991)

time of indexing. For example, the '#ow' operator (ordered window) requires the keywords (its operands) in a query to be within 2 word-span such as in '#ow(operating system)'. Another operator, '#uw', is similar to '#ow' but it does not take into account the order (which word comes first or last), hence, it is called *unordered window operator*. Such operators are useful because they enable the retrieval based on a little more structure than just bag of words. In fact, the ordered window operator with the window size corresponding to the number of keywords within a phrase is called the "phrase" operator.

In the actual query formulation for topic retrieval, the 'sum' operator is used to average the beliefs of all the individual keywords and the phrases in the query. So for example, the query for the question, "Who is the president of Germany?" would be formulated from the question analysis output, which as noted in Section 4.3.1, is [['is'], ['president', 'of', 'germany']] such as[7]:

```
\sum(\sum(is, president, of, germany, \phrase(president, of, germany)))
```

In this example, "president of germany" forms a phrase, and it is inserted as part of the query element with a phrase operator. However, the individual keywords of the phrase are also represented as bag of words query items since doing so gives better performance according to the trials I have run. With this query, the search is performed and a ranked list of topics is retrieved.

4.3.4 Post-processing Operations

Having retrieved a set of ranked topics as answer candidates, a further set of operations applies on this set, which makes up the post-processing. The goal is to further improve the accuracy by possibly pushing up the more likely topics as top answer candidates in the initial set and eliminating the unlikely candidates. This is to be achieved by making use of the question topic and the fine answer types identified from the target question.

First, Question topic presents a special consideration. It was mentioned in section 4.1.3 that whatever topic contained in a question is not likely to be the answer to this question. For example, the question, "Who is the president of Germany?", contains a topic, "Germany", and therefore this topic is by default not a potential answer to this question. However, it is possible that this topic "Germany" is present as one of the retrieved results as this is one of the keywords in the query formulated from the

[7]The commas are not originally part of the query but inserted here to clearly indicate the separation of keywords

question. When that happens, this topic would have to be eliminated as an answer candidate from the set. Thus, the post-processing operation related to question topic is essentially a filtering operation as any question topic in the list of candidates is to be simply filtered out.

Second, the fine-answer type from the question provides an information that can be used to rerank the candidates. While at the time of question analysis, the answer type of the target question was identified, both with respect to the fine answer type (e.g. president) and the base answer type (e.g. person), only the base answer type was made use of at the time of topic retrieval operation in selecting the apporpriate base index. The fine-answer type was not used. In the reranking operation, the fine-answer type identified from the question is to be matched up against the fine *topic type* of a candidate topic in the retrieved list. This fine topic type can be simply looked up by using Topic-Type hash table. The topic type looked up from this table is actually a set of hypernymy chains, for example, "PERSON::SCIENTIST::PHYSICIST, PERSON::THINKER::IDEAOLOGIST::SOCIALIST, etc". In matching up the fine answer type with topic type, the only requirement is that this fine type matches one of the element of a chain in the set. So if the fine answer type is PHYSICIST, and the fine topic type is this set in the example, the fine answer type matches the topic type as one element of a chain, and therefore the target topic (say 'Albert Einstein') is a matching topic. The reranking is done by identifying all the topics in the retrieved list that do *not* match the fine-answer type and downranking them below all the topics that do match. More sophisticated matching algorithm could be possible but this simple rule turns out to work well enough as will be seen in the next Chapter. This is due to the great comprehensivity of the information contained in the topic-type hash table.

4.4 Conclusion

This chapter introduced a method of Question Answering based on topics. Topics are named entities and loci of information. Topics are also potential answers to a class of questions (factoid named entity questions). The idea of *Topic Indexing and Retrieval for QA* is to identify topics in a corpus through off-line preprocessing, to associate textual content and type information for each of these topics, to index the topics with this information and to retrieve one these topics as answer candidates to a question.

The method consists of two distinct operations: preprocessing and retrieval. During preprocessing, the topics are extracted, their types identified and their content (the

sentences that mention the particular topic) assembled into a topic document. The collection of all such topic documents is then indexed for retrieval. The actual retrieval operation uses InQuery retrieval engine, which is based on the document inference network formalism, a form of Bayesian Belief Network. The structured language query supported by InQuery makes it possible to formulate a query that represents the question along the various information that it contains. A list of topics retrieved from such a query is then reranked according to the fine answer types identified from the question. The method promises to turn factoid named entity QA into a fine-grained Information Retrieval. Its ability to do so to what extent is an empirical question to be studied in the next chapter (Chapter 5).

Chapter 5

Evaluation I: TREC Questions

The previous chapter introduced the method of Topic Indexing and Retrieval for QA (henceforth to be called TOQA) and made two claims: that this method enables direct retrieval of answers for proper name questions and that it overcomes the locality constraints. In this chapter, the first of these claims is tested. The second claim will be tested in Chapter 6.

The chapter proceeds as follows. First, the objectives of the evaluation are stated, and the experimental configuration (i.e. the question set, the corpus and evaluation metrics) is laid out. Next, the baseline QA system is described against which the thesis' core method is tested, and the evaluation results of this system presented. Turning to TOQA, I describe its core implementation and the two different setups. Then evaluation results of these two setups are presented and analysed in depth. Finally, the chapter ends with a discussion of whether the results obtained meet the success criteria laid out at the beginning and what the implication of this is with respect to improving IR for QA.

5.1 Introduction

Proving the claim that Question Answering can be turned into fined-grained Information Retrieval for proper name questions that retrieves answers directly amounts to demonstrating that it has comparable performance to state-of-the-art QA systems of more conventional architecture. Thus, the first goal of the evaluation is to compare my method's performance on select TREC questions with that of a specially developed baseline QA system that has a more conventional IR+AE architecture but with some of the more advanced IR features. The comparison between this baseline system and

the system based on my method is with respect to both the correctness of answers and the overall speed of the operations. The second goal is to find out what aspects of the method contribute to its performance as revealed by the evaluation. The cases of success and failure are investigated in depth. Based on this investigation, I discuss how the retrieval model should be refined.

5.2 Evaluation Resources and Metrics

5.2.1 The Corpus and the Question Set

The following resources from TREC are used for the evaluation: (1) the AQUAINT corpus and (2) the question and answer set of TREC. The AQUAINT corpus consists of 1033461 newspaper articles in English spanning the period of the years from 1996 to 2000. All together, the corpus takes up about 3 Gigabytes of text.

The questions used in the evaluation comprise the subset of TREC questions pooled from TREC 2003, TREC 2004 and TREC 2005 that have a single named entity as an answer (excluding dates). Whether this limitation is intrinsic to the method will be discussed in Chapter 7. The subset comprises 377 questions. Among these, 133 come from TREC 2003, 85 from TREC 2004, and 159 from TREC 2005. (See Appendix **??** for the whole list.) Questions from TREC 2004 and TREC 2005 are grouped around what are called *targets*. A target is basically the question topic, e.g. "When was he born?" where "he" refers to the target, e.g. "Fred Durst". Where questions require co-reference resolution, I have employed a simple heuristic of replacing any pronoun in a question with the target (after shallow parsing the question). The 377 questions are questions with named entity answers. Named-entitles can be classified into different types, most generally PERSON, LOCATION, ORGANISATION and OTHER. The basic types of all the answers to these questions are shown in Table 5.1.

The answer types are important because ultimately the topics as answer candidates must match them. The answers of type PERSON include actors, baseball players, CEOs, presidents, generals, boxers, scientists, etc. The answers of type ORGANISATION encompass the sub-types of companies, universities, musical bands, groups, sport teams, etc. Answers of type LOCATION include, in addition to towns, cities, provinces, states and countries, more specific place names such as those of stadiums, museums and airports. The rest (OTHER) include fictional characters, animals, flags, roller coasters, coral reefs, desserts, international treaties, landmarks, currencies,

TYPE	NUM
PERSON	111
ORGANISATION	51
LOCATION	153
OTHER	62
TOTAL	377

Table 5.1: Answer Types of the 377 Questions used in the evaluation

awards, spacecrafts, dams, religions, gods, substances, albums, films, plays, books, etc.

5.2.2 Performance Metrics

In assessing the performance, four different measures, *Accuracy*, *A@N*, *Mean Reciprocal Rank* and *Average Rank of First Correct Answers*, will be used, each of which serves different purpose.

First, *accuracy* measures the percentage of the questions correctly answered. This measure is used in TREC for the performance assessment of factoid questions, and therefore this will be used in this evaluation as well.

Since systems generally produce a rank-ordered list of possible answers, it is also informative to see how close a system comes to answering a question correctly. Thus one can also measure how often an answer is found within the top 3 candidates, the top 5 candidates, the top 10 candidates and so on, up to the twentieth ranked candidates. Correct answers beyond rank 20 are not very meaningful and thus will not be considered in scoring. This measure is called the *accuracy at N* or simply A@N measure (Monz, 2003b). So a value of 0.500 at A@5 means that 50 per cent of all questions were correctly answered within the 5 top ranked candidates by a particular system. This A@N measure is used to give a broader look at the performance.

Another measure used for the evaluation is *Mean Reciprocal Rank (MRR)*. Like the accuracy measure, this measure gives a single score for the performance but by averaging the inverted rank of answers up to rank 5 for all questions, MRR takes into account correct answers beyond the number one ranked answer giving a broader perspective than the accuracy measure which is also a single number measure. The formula for the MRR is as follows:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} InvRank(q) \tag{5.1}$$

where $InvRank(q)$ is 0 if there is no correct answer for the question, q, within the rank 5, or otherwise:

$$InvRank(q) = \frac{1}{RankA(q)} \tag{5.2}$$

where $RankA(q)$ indicates the rank of the first correct answer for the question q.

Finally, I define and use what I call the *Average Rank of First Correct Answers (ARC)* measure. The purpose of this measure is to give an idea of where in the rank order, a correct answer would be found by the system if one were to be found within the top 20 candidates. Thus, questions without a correct answer within the top 20 are excluded. The formula is:

$$ARC = \frac{1}{|A|} \sum_{a \in A} Rank(a) \tag{5.3}$$

where A is the set of questions for which an answer was found among top 20, and $Rank(a)$ indicates the rank of the answer a.

In addition to these more or less conventional performance measures that give scores, significance tests between different sets of results are performed when there is a need to see whether the performance differences between the sets are really statistically significant. In general, the test data for the statistical test is the set of differences in ranks for all the questions between two different sets of results being compared. I use the *Wilcoxon Matched Signed Rank Test* (Wilcoxon, 1945), which is a non-parametric test for testing two related samples. The widely used Student's paired t-test is not suitable for this evaluation because one cannot assume that the underlying probability distribution is normal for the evaluation data (differences in ranks between result sets) as required by the t-test. The Wilcoxon Matched Signed Rank Test, on the other hand, does not assume any distribution at all. The null hypothesis for the test is that the results of the two sets of results being tested, namely the ranks of the answers of the two respective sets, do not differ significantly. The alternative hypothesis is the rejection of this null-hypothesis, namely that there is a significant difference. The α for the evaluation is set to 0.05. If the *p-value* is less than this, the null hypothesis is rejected. The actual computation of the tests was performed on the website of University of Amsterdam in the Netherlands, which provides online testing facility (http://www.fon.hum.uva.nl/Service/Statistics/Signed_Rank_Test.html).

5.3 The Baseline QA System

The baseline system I have developed follows the conventional IR+AE architecture, which the core method of the thesis contrasts with. To make the comparison more meaningful, the IR component of the baseline system incorporates some advanced features available within the conventional framework, thereby challenging my claim that the conventional QA with advanced IR is not sufficient for system efficiency. This assertion, basically the null-hypothesis, is what the core method must prove to negate in this evaluation.

5.3.1 System Description

The baseline QA system (*BAQA* henceforth) follows the conventional IR+AE architecture as introduced in Chapter 2 in which a question is analysed, the relevant documents (passages/sentences) are retrieved, a set of answer candidates is extracted from the retrieved documents and finally reranked according to some additional criteria. Before the online Question Answering operations, the base corpus is subjected to preprocessing to produce the index. The more advanced IR features of this baseline system are:

- Predictive Indexing and Retrieval
- Sentence Retrieval
- Retrieval with Indri Structured Query Language

In Predictive Indexing and Retrieval (introduced in 3.2), the potential answers (named entities) are identified and marked up in the corpus with their named entity types during preprocessing. The annotated corpus is indexed not only with respect to the terms (words) but also their named entity types (PERSON, LOCATION, ORGANISATION). In retrieval, the entity in question (answer) can be specified as to its type and proximity to other terms in the question. Sentence retrieval is a technique that uses sentences as the units of retrieval rather than whole documents. The advantage is the smaller amount of text retrieved as compared to whole document retrieval. Sentence retrieval requires the indexing unit to be the sentence, and thus the base corpus needs to be tokenised into sentences beforehand. Thus the preprocessing of the corpus involves named-entity annotation and sentence tokenisation (also annotated inline with XML tags). My core method uses the same data.

The indexing and retrieval operations are performed by Lemur IR Tool-Kit (version 4.6). The state-of-the-art Indri querying and retrieval system in the tool kit is a further development of Inquery where the TF-IDF scoring formula is replaced with a unigram language model based formula under the same overall inference network formalism. In addition to the advanced querying features such as the combined weighting of individual term likelihood and proximity of multiple terms, Indri Query Language allows the predictive annotation based retrieval and sentence retrieval within its querying syntax.

However, as the implementation of the core method of TOQA (Nexus) uses InQuery rather than Indri (as explained in Section 4.3.2), an explanation is needed as to why the baseline system uses a different IR system to the one to be tested against. After all, shouldn't the comparison be between like-for-like systems? The answer to this question is simply that Indri supports a stronger baseline than Inquery while maintaining enough similarity to the original InQuery engine. Indri and InQuery both uses document inference network as the underlying search model. Where they differ is in how they estimate one of the component probabilities: Indri uses unigram language model based estimation rather than TF-IDF based one as used in Inquery. As the former is superior to the latter, Indri can be simply regarded as an upgraded version of InQuery. Further more, as a matter of convenience, Indri engine supports some of the XML information retrieval features which are very handy for the kind of advanced IR operations to be used in the baseline system (as will be shortly explained). Nexus, however, does not use these features, and more over, was developed when Indri was not yet available. The only negative aspect of Indri over InQuery is that it is relatively new and thus it may be speed-wise less-optimised, a compromise that must be weighed against the advantages it offers in terms of rich features and improved retrieval performance. All in all, Indri presents a stronger baseline as compared to when InQuery would have been used because the comparison is between a conventional QA system with more advanced IR features (the baseline system) and a QA system with a radically different use of IR (TOQA system).

The extant operator of the Indri Query Language enables the retrieval of individual sentences rather than documents if the sentence elements (e.g. $< SENT >$) have been defined as extants at the time of indexing. The field operator (:) limits the type of entity specified by a term (e.g. Bush:PERSON). Combined with the #any operator, which matches any term of some specific type (e.g. #any:PERSON can match any human being), it supports, in essence, the predictive annotation tokens via a sort of a semantic wild card.

In order to produce Indri query, however, the natural language question has to be analysed and parsed. The question analysis unit is identical to the one used for my core method implementation (Section 4.3.1) except that the translation to the Indri query also involves automatically inserting the #any, field and the extant operators. The following is an example of the Indri query produced from the question "Who is the president of Germany?":

```
#combine[sent](#any:PERSON is the president of Germany
#uw3(president of Germany))
```

Here the #combine[sent] operator produces a weighted sum of the query elements within a sentence (as indicated by [sent] extant), which consists of the bag of words (is the president of Germany), the wild card (#any:PERSON), and the proximity operator simulating the phrase constraint (unordered window of three words: #uw3(president of Germany). I have explored different weighting scheme (the assignment of different weights to different keywords), but I was not able to come up with a better weighting scheme than the one presented here (equal weighting). For example, the retrieval performance worsened when the weights were adjusted so that a question topic, supposedly a more salient keyword in a question, got a higher weight (1.0) as against other keywords (0.5). Query optimisation as such is not a trivial task, especially for Question Answering. More study is needed for the future work in this area. Thus, for the present purpose, a more or less straight forward query formulation method is used as found in other works using Lemur toolkit (for example in Strohman *et al.* (2005)).

Once the question has been analysed and the Indri query produced, the IR unit retrieves a set of one hundred sentences (the mark up of XML tags from preprocessing are carried over here for easy answer extraction). From this set, I have implemented the baseline system to use the following rules to extract answer candidate phrases:

1. Extract all the named entity marked up with the same type as the base answer type.

2. If there is more than one named entities in a sentence, choose the one with the shortest average distance to other terms of the question as found in the sentence.

3. Rank the extracted answer candidates according to the IR sentence ranking and select the top 20 for assessment.

The ranked list of answer candidates are basically topics, and they are reranked essentially using the same reranking mechanism as described in 4.3.4 using the fine answer type identified from the question.

5.3.2 Evaluation Results of BAQA

The evaluation results for the baseline QA system are assessed with respect to three criteria. In order to assess the IR performance of the system, the first set of scores is for sentences retrieval (i.e. without answer extraction). So if a retrieved sentence contains somewhere the correct answer for a question at a rank 3, then the question is marked correct with the answer ranked at 3. This assessment (labelled "SNIPPET") gives an idea of the IR performance without the possible errors in the following answer extraction and reranking operations. The second assessment considers the answers extracted from the sentences by the answer extraction module. (designated as "ANS-EX") So now the answer candidates themselves have to match the correct answers in order to be marked correct; it is not enough for the retrieved sentences to contain the correct answers. The final assessment look at the final answer candidates after they have been reranked according to the fine-answer types. This set of results is labelled "ANS-RR".

Table 5.2 summarises the results of all these assessments, where the performance of each set of results is laid out in a side-by-side fashion so that they can be easily compared. The left-most column indicates the cut-off point (N) for A@N performance scores. The next three columns indicate the A@N performance score data for the sentences (SNIPPET), the extracted answers without reranking (ANS-EX) and the answers with reranking (ANS-RR) respectively at this particular cut-off point. Here each entry consists of two scores separated by a colon: The ratio of correctly answered questions over all questions and the number of correctly answered questions. The values in parentheses represent absolute gain or loss (in terms of the number of correctly answered questions) over the value to its left, such as the gains in ANS-RR over ANS-EX. The three rows at the bottom summarise the results by giving the accuracy (ACC), which is equivalent to the accuracy at A@1, the Mean Reciprocal Rank score (MRR) and finally the average rank of the first correct answer (ARC).

The results show that the accuracy performance is not very good for ANS-EX. The simple answer extraction rules used were not adequate, reducing the overall performances by losing correct answers contained in the sentence snippets. This means

A@N	SNIPPET	ANS-EX	ANS-RR
1	0.188:71	0.114:43(-28)	0.186:70(+27)
2	0.236:92	0.143:57(-35)	0.249:94(+37)
3	0.276:104	0.188:71(-33)	0.276:104(+33)
4	0.300:113	0.239:90(-23)	0.308:116(+26)
5	0.334:126	0.268:101(-25)	0.332:125(+24)
6	0.377:142	0.289:109(-33)	0.355:134(+25)
7	0.390:147	0.308:116(-31)	0.361:136(+20)
8	0.403:152	0.321:121(-31)	0.366:138(+17)
9	0.419:158	0.332:125(-33)	0.371:140(+15)
10	0.430:162	0.345:130(-32)	0.379:143(+13)
15	0.475:179	0.382:144(-35)	0.390:147(+3)
20	0.520:196	0.398:150(-46)	0.395:149(-1)
ACC	0.188	0.114	0.186
MRR	0.246	0.169	0.245
ARC	5.339	5.008	3.125

Table 5.2: Results for Baseline QA System for all questions

A@N	PERSON	LOCATION	ORG.	OTHER	*OTHER
NumQs	111	153	51	62	62
1	0.108:12	0.281:43	0.235:12	0.048:3	0.145:9
2	0.189:21	0.359:55	0.333:17	0.065:4	0.226:14
3	0.234:26	0.392:60	0.333:17	0.065:4	0.258:16
4	0.279:31	0.438:67	0.333:17	0.065:4	0.274:17
5	0.279:31	0.490:75	0.353:18	0.065:4	0.274:17
6	0.306:34	0.523:80	0.373:19	0.065:4	0.290:18
7	0.306:34	0.536:82	0.373:19	0.065:4	0.290:18
8	0.306:34	0.549:84	0.373:19	0.065:4	0.306:19
9	0.315:35	0.556:85	0.373:19	0.065:4	0.306:19
10	0.315:35	0.569:87	0.373:19	0.080:5	0.323:20
15	0.333:37	0.575:88	0.392:20	0.080:5	0.339:21
20	0.342:38	0.575:88	0.412:21	0.080:5	0.419:26
ACC	0.108	0.281	0.235	0.048	0.145
MRR	0.175	0.353	0.288	0.056	0.200
ARC	3.553	2.886	3.095	2.800	6.154

Table 5.3: Results for ANS-RR for Different Question Types

that answer extraction from sentences is not as trivial as it might appear even at the granularity of sentences marked up with named entity types. ANS-RR is significantly superior to ANS-EX (W+ = 3486, W- = 0, N = 83, p <= 2.575e-15 according to Wilcoxon Matched-Pairs Signed-Ranks Test). The reranking operation based on fine answer type thus salvaged the overall QA results by up-ranking the correct answer candidates found at a lower rank by the answer extraction unit. (This is possible because of the redundancy of relevant evidence; multiple sentences in different ranks contained the correct answer.)

Table 5.3 summarises the results according to the base answer type for ANS-RR results. It reveals that the Answer Extraction unit had serious problems with answers of OTHER type, partly explaining the low performance scores of ANS-EX. Wrong base type analysis pertaining to OTHER resulted in low scores for all the questions with this answer type, which the reranking operation cannot do anything about. (Wrong base type necessarily results in wrong fine type). However, the rightmost column labelled

with OTHER* shows that the retrieved sentences (SNIPPETS) contained more correct answers than what the ANS-EX were able to extract. An investigation reveals that there was a simple error with the answer extraction module that failed to correctly extract most of the entities that belong to OTHER class from the snippet. In any case, this table shows that BAQA system has the best performance for LOCATION, followed by ORGANISATION questions.

What I would like to focus here however is on the Information Retrieval performance itself. Table 5.2 shows that up to rank 20, half of all the correct answer candidates were retrieved at the sentential level (SNIPPET), making it possible to suppose that a more sophisticated answer extraction unit would have put the performance of the whole QA system at a better level. Considering that hundreds of whole documents are known to be needed to guarantee enough coverage for IR, the single sentence retrieval result is not bad. However, where the performance really falls down is in the overall retrieval efficiency, having taken 15 hours and 14 minutes to do all 377 questions, roughly 2.42 minutes per question on an Intel(R) Pentium 4 CPU machine at 3.00GHz with 1 GB of RAM. (It took another 37 minutes 14 seconds to do answer extraction – 5.9 seconds per question.) The substantial time taken by IR is somewhat surprising considering the general speed and efficiency of document retrieval. But the IR here, on a closer look, is not an ordinary document retrieval so this time inefficiency may be due to one or more of the following three reasons:

1. Sentence retrieval substantially increases the number of entities in the index as each document is in essence subdivided into many mini-documents.

2. The #any operator is very expensive. Unlike a term, the #any operator can match any term. Even when #any operator is restricted with a field operator (e.g. #any:PERSON), it can match up with several entities, multiplying the overall number of scoring operations.

3. The Indri Query Engine is, speed-wise, less optimised than the Inquery engine it replaces.

Even with future improvements in the Indri Query Engine, the combination of one and two in the above makes the retrieval operation of a greater complexity than ordinary document retrieval operation demanding more time. Just exactly what the complexity amounts to, whether the efficiency can ever match that of the general document IR when everything is done, is something that needs a separate study.

The results presented here will be compared against TOQA evaluation results in Section 5.5.1.

5.4 TOQA Evaluation System

5.4.1 The Core of the Evaluation System: Nexus

The common core system that implements the answer retrieval method is called *Nexus* (Ahn and Webber, 2007), the general architecture of which was described in 4.3.1. This core system comprises (1) a question analysis module that analyses the question and produces the question type, answer type, the question topics and the keywords and (2) a retrieval module that generates the structured query, selects the appropriate index and retrieves the top 100 topics as answer candidates.

This core system performs the basic retrieval operations, which is one system configuration (setup) for the evaluation. The addition of fine-answer type based reranking to this core system makes up another system configuration. Despite this difference, all configurations share the following elements:

- Test question set.
- The corpus that provides a source of answers to the questions.
- The number of topics (100) to be retrieved for each question.

5.4.2 TOQA-A: The Bare-bone System

TOQA-A is the bare-bone Nexus system. The topic document collection that is indexed for this system is created by applying the topic document method described in Section 4.2.2 on the AQUAINT corpus. The resulting topic documents are divided into the three base types plus other type: PERSON, LOCATION, ORGANISATION, OTHER and TOTAL (for all topics including others) as summarised in Table 5.4.

The topic document collection is indexed to produce four separate indices, each of which corresponding to either a PERSON, an ORGANISATION, a LOCATION or a TOTAL type. The TOTAL index is used when the answer type of a question is not one of the three core types (OTHER) or indeterminate. (An 'OTHER' index is not created as the TOTAL index contains those elements). Otherwise, the index of one the three types is picked and bound to the retrieval module depending on the result of

KIND	NUM
PERSON	117370
ORGANISATION	67559
LOCATION	48194
OTHER	17942
TOTAL	251065

Table 5.4: Number of Topic Docs per Types

the question analysis and the particular QA strategy. A total of 100 ranked topics is retrieved for each question as answer candidates. Among the topics retrieved, if there is a topic that matches the question topic (thus being superfluous), it gets effectively filtered out by being pushed to the bottom rank, i.e. 100th.

5.4.3 TOQA-R: TOQA-A with Answer Reranking

TOQA-R adds fine-grained answer type reranking to set up A as a post-retrieval operation, similar to the re-ranking of answer candidates of the baseline system. So one hundred topics as answer candidates are retrieved as in TOQA-A, but the ranked list of topics is reranked depending on whether the fine type of each candidate topic matches the fine answer type identified from the question. Note here that only the coarse (base) answer type (one of the four base types, PERSON, LOCATION, ORGANISATION, TOTAL) was used at the time of retrieval as opposed to the fine type such as PRESIDENT or COMPANY due to the fact that separate indices exist only for these base types. The identification of the fine topic type of a candidate topic is done by looking up this information in the topic-type hash table as mentioned in Section 4.3.3.

5.5 TOQA Evaluation Results

5.5.1 Overall Performance

From Table 5.5, it can be seen that TOQA-R produced results that are superior to TOQA-A in all measures: accuracy, A@N (for N up to 20), MRR and ARC. The difference (superiority) between TOQA-R and TOQA-A is indeed significant as the Wilcoxon Matched Signed Rank test shows: The p-value of the test between the two

A@N	TOQA-A	TOQA-R
1	0.233:88	0.340:128(+40)
2	0.316:119	0.406:153(+34)
3	0.366:138	0.438:165(+27)
4	0.401:151	0.467:176(+25)
5	0.430:162	0.491:185(+23)
6	0.443:167	0.504:190(+23)
7	0.456:172	0.512:193(+21)
8	0.459:173	0.512:193(+20)
9	0.464:175	0.517:195(+20)
10	0.472:178	0.523:197(+19)
15	0.496:187	0.533:201(+14)
20	0.512:193	0.541:204(+11)
ACC	0.233	0.340
MRR	0.306	0.395
ARC	3.394	2.441

Table 5.5: Results for all setups for all questions.

A@N	A-R	R-A	A ∩ R
1	0	40	88
2	1	35	118
3	2	29	136
4	2	27	149
5	2	25	160
6	2	25	165
7	2	23	170
8	2	22	171
9	3	23	172
10	3	22	175
15	4	18	183
20	4	15	189

Table 5.6: Overlap between TOQA-A and TOQA-R

sets of results is much smaller than the α level of 0.05 (p = 1.761e-08), justifying the rejection of the null-hypothesis that there is no difference.

Tables 5.6 details the overlap between setups A and B. Table 5.6 shows that for every cut-off point below 20, TOQA-R answered more questions that TOQA-A wasn't able to answer, while TOQA-A only answered very few questions that TOQA-R wasn't able to answer (A and R in the table refers respectively to TOQA-A and TOQA-R). This shows that TOQA-R's significant superiority in performance over A comes from the fact that TOQA-R correctly answered most of the same questions as TOQA-A and other ones besides. Thus it can be concluded that the reranking of topics according to their match to the fine answer type produces unequivocal benefit.

5.5.2 Performance Variation over different Named-Entity Types

Here, I look at the scores with respect to the base named entity types (PERSON, LOCATION, ORGANISATION, OTHER) in order to see whether there is any score differences with respect to answer types. Table 5.7 shows the scores with respect to three base answer types plus other. The results here show that for LOCATION questions, TOQA-R performed less well as compared to questions of other base types. The investigation as to why LOCATION questions are harder for TOQA will be presented in

A@N	PERSON	LOCATION	ORG.	OTHER
NumQs	111	153	51	62
1	0.324:36	0.300:46	0.392:20	0.419:26
2	0.405:45	0.372:57	0.431:22	0.467:29
3	0.423:47	0.418:64	0.470:24	0.483:30
4	0.468:52	0.431:66	0.529:27	0.500:31
5	0.522:58	0.444:68	0.549:28	0.500:31
6	0.540:60	0.464:71	0.549:28	0.500:31
7	0.549:61	0.464:71	0.588:30	0.500:31
8	0.549:61	0.464:71	0.588:30	0.500:31
9	0.558:62	0.464:71	0.588:30	0.516:32
10	0.576:64	0.464:71	0.588:30	0.516:32
15	0.603:67	0.470:72	0.588:30	0.532:33
20	0.621:69	0.470:72	0.607:31	0.532:33
ACC	0.324	0.300	0.392	0.419
MRR	0.393	0.358	0.443	0.453
ARC	3.391	1.761	2.484	1.879

Table 5.7: Results for TOQA-R for different types of questions

Section 5.5.4.

5.5.3 Performance Speed

The evaluation of TOQA was run on the same machine as the baseline system. The topic retrieval operation for the 377 questions took 36 minutes and 42 seconds to complete, which is slightly less than 1 second per question. The reranking operation added 2 minute and 30 seconds total, or less than 0.4 second per question.

5.5.4 Analysis of TOQA Results

This subsection analyses the results of the TOQA evaluation. The goal is to understand the different circumstances in which TOQA system succeeds and fails to retrieve the correct answers. By attaining a better understanding, the retrieval method can be refined in the future for better retrieval performances for Question Answering. The chief questions of this investigation are:

1. When does TOQA succeed and fail to rank the correct topic as the top candidate?
2. Why does TOQA perform less well for LOCATION questions?
3. In light of the above two questions, how should the scoring formula be revised?

It may seem that the answer to first question is already known, since we already know the scoring formula used in retrieval as explained in Section 4.3.1. This is however not really so. The formula scores documents based on TF-IDF scores and term proximities, combining them under an inference network formalism. In other words, the actual scoring process is complex enough to be less than intuitive, and thus a post-retrieval analysis can provide clues to how the formula in practice operates. Moreover, the formula is originally devised for retrieving ordinary documents with ordinary queries, and therefore, an analysis of how the retrieval operates with respect to topics for QA is needed.

5.5.4.1 Methodology

The investigation will look at only the results of TOQA-A, namely the raw retrieval performance, and not at those of TOQA-R that reranks the originally retrieved set of topics based on fine answer-type. The reason is that here the chief interest for

investigation is the retrieval performance rather than the reranking performance, which depends on other factors.

Although the primary objective of the investigation is to find when the correct topics are ranked top and when not, it is not possible to compare the retrieval results of different questions directly, e.g. between a question for which the correct topic was ranked top and a question for which the correct topic was ranked lower. This is because the ranking within one retrieval run is relative: A topic is ranked higher than another topic when it gets a higher score than the others, even if its absolute score is very low, and a lower ranking topic might have a higher absolute score in another question than this one.

Thus, my analysis is based on comparing correct and incorrect answers (topics) from the *same* question. More precisely, the candidate that is the correct answer to a question will be compared to ones that are ranked higher (ranked top) in the retrieval but are incorrect and also ones that are ranked lower and incorrect. By doing so, it is possible to see when the correct topic is ranked lower than an incorrect topic (a undesirable case) and when the correct topic is ranked higher than an incorrect topic (a desirable case). So for each question, I will be considering the following:

1. *TopInc*: A topic that is ranked top by retrieval but is not the correct answer.
2. *MidCor*: A topic that is correct but not top ranked by retrieval.
3. *LowInc*: A topic that is not correct that ranks one lower than answer.

My analysis will be with respect to the following three cases:

1. TopInc Vs. MidCor (A case of failure): To see when the correct topic does not succeed in being ranked higher than an incorrect topic.
2. MidCor Vs. LowInc (A case of success): To see when the correct answer is ranked higher than an incorrect topic.
3. TopInc Vs. LowInc (A general case): To see when one incorrect topic gets ranked higher than another topic.

To make the comparison statistically meaningful, there should be more than one sample for comparison, which means more than one question. In selecting the sample questions, questions with top ranked correct topics are excluded as they lack “TopInc”. Questions with the correct topic ranked too low are also excluded, as the correct topics

for these questions hardly indicate success even when they are ranked higher than the LowInc topics. As such, I only consider questions where the correct answer ranks between 2 to 5 ranks in the retrieval. There are 74 such questions for TOQA-A, and all of them are used for this analysis.

Now the question is what measure(s) should be used for the comparison of topics. For the sake of simplicity and clarity, the investigation uses the following three key measures:

1. QS (Qterm Saturation): What percentage of terms (tokens) of a topic document of the target topic corresponds to the terms of a question (qterms) – Ratio of qterms over all terms of a topic document.

2. QD (Qterm Diversity): What percentage of the terms (identical to term types) of a question is found in the topic document of the target topic – Ratio of qterms found in topic document over all qterms in a question.

3. QSQD: A combined measure: QS times QD.

These two measures are simple enough to be intuitively clear. A correct answer (topic) to a question would contain a lot of the terms of the question within its topic document as indicated by Qterm Saturation measure. Also, the topic document would contain more of the qterms of a question than less likely topic as indicated by Qterm Diversity measure. QSQD is a combined measure that take into account both QS and QD. While there can be other measures of performance affecting factors, considering every possible factor makes the investigation overly difficult, and, as will be shortly shown, not really necessary as these measures by themselves turn out to be adequately effective for this investigation.

The analysis runs as follows:

1. Preparation: For each question, the three topics corresponding to TopInc, MidCor and LowInc are identified from the retrieval result from TOQA-A. Their corresponding topic documents are also retrieved for analysis.

2. Measurements: For each of these topics (TopInc, MidCor and LowInc), QS, QD and QSQD scores are computed on the target topic's topic document and tabulated into a table such as Table 5.8

3. Comparison: For each test case and each measure, compare the scores between two topics and count the times each topic wins (i.e. having greater score), as

Question	TopInc	MidCor	LowInc
Q1	0.021	0.018	0.016
Q2	0.031	0.022	0.018
..	..	..	..

Table 5.8: Qterm Saturation Scores (Example)

Question	TopInc vs. MidCor	MidCor vs. LowInc	TopInc vs. LowInc
Q1	TopInc > MidCor	MidCor > LowInc	TopInc > LowInc
Q2	TopInc > MidCor	MidCor > LowInc	TopInc > LowInc
..	..	..	..
TOTAL	42 vs. 32	51 vs. 13	52 vs. 12
Winner	TopInc	MidCor	TopInc
W. Signif.	no	yes	yes

Table 5.9: Qterm Saturation Comparison Per Cases

in Table 5.9. Pick the topic with majority win as the overall winner for that measure.

4. Significance Test: Compute the significance of "winning" by doing Wilcoxon paired signed ranked test using the data in Table 5.8. As usual the significance level.

5.5.4.2 Results and Discussion

Table 5.10 summarised the final comparison results (X:Y in the table entry indicates Winner Topic:(Not)Significant). Qterm Saturation seems to be the factor that deter-

Metrics	TopInc vs. MidCor	MidCor vs. LowInc	TopInc vs. LowInc
QS	TopInc:NSIG	MidCor:SIG	TopInc:SIG
QD	MidCor:SIG	MidCor:SIG	TopInc:NSIG
QSQD	MidCor:NSIG	MidCor:SIG	TopInc:SIG

Table 5.10: Comparison Results

mines which topic gets ranked higher. In all cases, the topic with higher QS score has a higher rank (but not significantly in the case of TopInc vs. MidCor.) This indicates that the original scoring formula for retrieval tends toward the ratio of the terms of the query (question) relative to other words in the topic documents in determining the retrieval ranking. Hence, QS shows that the correct topic gets ranked lower than incorrect topics when the ratio of qterms over all terms in its topic document is less favourable to some other topics.

On the other hand, Qterm Diversity measure indicates what the characteristics of a correct topic is like: QD score is higher for MidCor (i.e. the correct topic) whether it is of a higher or lower rank than other topics (i.e. TopInc and LowInc).

In determining when the correct topic gets ranked higher than incorrect topics, QS and QD are equally indicative, as MidCor topics have both significantly higher QS and QD scores than LowInc topic, but when it comes to when the correct topic gets ranked lower than an incorrect topic, the QD score of MidCor didn't seem to have an effect.

Thus a combination of high QS, which makes a topic ranked higher, and high QD, which picks out the correct topic, would rank the correct topic the highest. Indeed, the simple combined measure of QSQD shows that, had topics been scored based both on QS and QD, the correct topic would likely to have been ranked top – a strong hint as to how to revise the retrieval formula in the future.

The QS scores also suggests why LOCATION topics are harder for TOQA. The ranking formula used for IR highly ranks topics with high QS scores. A high QS score means that the terms of the question occupy a higher percentage of the total terms in a topic document. For example, if there is only one qterm A, then a topic document with a higher percentage of A over another topic document would have the higher QA score. What this means is that the more homogeneous a topic document is with respect to the varieties of the keywords within, the easier it is to get a higher QS score. It looks as though LOCATION topics, especially of higher granularity (e.g. a country) seem to contain too much diverse information (inhomogeneous) in general (e.g. Germany is where Albert Einstein is born, while it hosted the Olympic games in 1936, while it is a western European country, etc.) and thus be penalised when it comes to QS score. In order to test this hypothesis, I have sampled 30 LOCATION topic documents (countries and US states) and counted the number of unique terms. I also looked at 15 PERSON topic documents and 15 ORGANISATION topic documents. In average, there were 41606 unique terms for a LOCATION topic document in the sample, which contrasts to an average of 7804 unique terms for PERSON/ORGANISATION docu-

ments. This difference in the number of unique terms (roughly about 5 times more for LOCATION topics) indicates that LOCATION topic documents are far less homogeneous than non-LOCATION topic documents confirming the explanation as to why LOCATION questions are more difficult for TOQA.

5.6 Discussion and Conclusion

5.6.1 Comparison: TOQA against BAQA

The primary purpose of this evaluation is to gauge the performance of the core method (TOQA) against a QA method, the baseline system (BAQA), that has a more conventional IR+AE architecture but with some advanced IR features. The evaluation results in Table 5.2 and Table 5.5 show that TOQA is superior to BAQA in every measure, and that the difference in performance is statistically significant (W+ = 14980.50, W- = 13460.50, N = 238, p $<=$ 0.045 for TOQA-R vs. BAQA-ANS-RR) with respect to the correctness of the answer. For location questions, the difference in performance between BAQA and TOQA is much smaller (MRR of 0.358 (TOQA-R) vs 0.353 (BAQA-RR)), and while location questions are the worst performing answer type for TOQA, they are the best for BAQA. With respect to the overall processing speed, TOQA is also far superior (less than 1 second per question for TOQA vs. more than 2 minutes per question for BAQA). What can be concluded from these results?

First, BAQA shows that advanced IR features alone within the conventional QA framework is not sufficient to guarantee a good overall performance: It requires good Answer Extraction module, even when the IR takes up some of the functions of the Answer Extraction through sentence retrieval, predictive annotation tokens and proximity based query formulation and retrieval.

Second, BAQA shows that the advanced IR features within the conventional architecture does not speed up the overall process because they make IR costly and slow: Whatever time saving that comes for AE, it is more than negated by the heaviness of IR. Hence the conventional IR+AE architecture is not speed-wise improved by a few advanced IR features as incorporated in BAQA.

Third, TOQA shows a substantial speed advantage over BAQA because it uses a simple document Information Retrieval technique. The time saving comes from preprocessing, which enables the direct retrieval of topics without Answer Extraction by turning potential answers into documents.

System	Accuracy (Nil Adj.)
lcc05 Language Computer Corp.	**0.713 (0.736)**
NUSCHUA1 National Univ. of Singapore	0.666 (0.689)
IBM05L3P IBM T.J. Watson Research	0.326 (0.349)
ILQUA2 Univ. of Albany	0.309 (0.332)
Insun05QA1 Harbin Inst. of Technology	0.293 (0.316)
csail2 MIT	0.273 (0.296)
FDUQA14B Fudan University	0.260 (0.283)
QACTIS05v2 National Security Agency (NSA)	0.257 (0.280)

Table 5.11: TREC 2005 System Performances for Factoids in terms of Accuracy

Fourth, the small difference in performance for location questions between TOQA and BAQA indicates that the inhomogeneity of a topic document reduces the advantage that the core method has over the more conventional QA method.

Finally, the superior performance of the core method of this thesis demonstrates that the improvement in information retrieval for QA is more beneficially applied to the kind of novel QA architecture of the core method rather than on the conventional IR+AE architecture.

5.6.2 TOQA Performance in Perspective

How do the results of TOQA here compare to the systems participating in TREC? Although I do use TREC questions for the evaluation, direct comparison is not possible because I have used only a subset of the questions from TREC which are strictly compatible with the method. There is no information as to how the TREC systems have scored on individual questions, and therefore direct comparison of the accuracy scores is meaningless.

However, for illustrative purpose only, Table 5.11 shows the factoid performance of the top 8 systems participating in TREC 2005 (Voorhees, 2005). In total, 30 systems participated. Since the system doesn't attempt to handle the more difficult nil questions (i.e. ones that lack an answer in the corpus), I also include in parenthesis the nil-adjusted scores for these systems (this assumes that half of all nil questions are correctly answered amounting to a uniform boost of 0.023, which represents an optimistic view that if the nil question were a non-nil factoid question, half of the time it would

be answered correctly.). TOQA system's best performance on TREC 2005 questions are 0.351, which is superficially comparable to the third ranked system among 30 participants at TREC 2005. This, at least, gives some idea of TOQA's performance in general.

While this top third performance of TOQA does not seem to be too bad overall, the wide gap in scores as compared to the top two systems must be given some consideration. Would TOQA ever be able to score as well as the other two systems? Is there a performance cap for TOQA that is the result of the method's fundamental limitation? There are certainly rooms for improvements for TOQA as the system itself is essentially just a proof-of-concept system and therefore not without errors as shown in the next section. But apart from these more or less trivial errors, the core method could be improved in many ways, as are discussed in section 7.2 as future work. When these are done, the true potential of the core method will be revealed. So the verdict is still out.

5.6.3 TOQA and the Web

Topic Indexing and Retrieval method is a very specialised method which uses a closed corpus. However, since the web is the ultimate corpus with the benefits of frequent updates and the great breadth of information, and since the widely used web search engines such as Google are the primary means of accessing information on the web, it is necessary to consider whether there is a way to incorporate web search engines into TOQA. The chief problems are that the common web search engines are document retrieval systems in the conventional sense and that they will remain so in the foreseeable future. Thus, the issue becomes whether TOQA can be tweaked to incorproate a more conventional IR method with minimum costs.

The answer to this question, I believe, is that it is possible by using the search engines as the means to construct "topic document collection". Search logs of a web search engines contain popular search terms. Taking these terms as topics, each search term can be queried to a search engine, and the search results (the text snippets) for this term can be considered the textual content of this topic. So collecting all these search results and processing them in the fashion of topic document method, it is possible to have a Topic Indexing and Retrieval based QA on the web. This, in effect, amounts to creating a meta-index out of the original index. As the original index gets updated, this meta-index would need to be updated from time to time.

So in this fashion, the existing web search engines can be put to use for TOQA with all the benefits they provide.

5.7 Error Analysis

The failure to answer a question can be due to the characteristics/limitation of the method, which was discussed in the previous section. It can also be due to errors, either simple mistakes in the program or some imperfection of a component module that can be fixed or expected to be improved upon in the future. As Question Answering involves numerous operations, an error in each stage of operations can result in the overall failure to produce the right answer for a question. Here, such errors are investigated.

The first operation that can result in immediate failure is the question analysis, which is more or less identically shared by all the different systems/configurations (i.e. BAQA, TOQA-A, TOQA-B).

In three cases, the Question Analysis unit outright failed to parse the questions resulting in the questions being skipped altogether. This error turns out to be due to some peculiarity related to the parser being used: NLProcesser's parser failed to parse text less than about 20 bytes long. (e.g. "Where is it?")

By far the most serious errors relate to identifying the answer types. In identifying the answer type, the first thing that is looked at is the WH-word. The WH-word such as 'Who' is, by default, associated with the answer type PERSON. However, there are some cases where such an association does not hold. For example, in the question, "Who attacked the Finance Center in New York?", the answer can be of type ORGANISATION (e.g. Al Qaeda). Clearly a more sophisticated answer classifier will be of great benefit.

In the baseline QA system, an error in Answer Extraction resulted in a lot of questions with OTHER type answers being missed. The snippets returned by IR are tagged with the three main types (PERSON, LOCATION, ORGANISATION), and the entities of OTHER type are all the other proper nouns not marked up as one of these types. The answer extraction unit didn't correctly extract these proper names due to a bug in the program, a simple mistake that can be easily fixed for future.

All in all, the errors described thus far could be dealt with in the future by simply fixing bugs in the programs, or by adopting a better performing modules such as a better answer classifier.

Chapter 6

Evaluation II: Multi-Evidence Questions

This chapter addresses the second problem laid out in Chapter 4 as the main motivation for the thesis, namely the locality constraint inherent in the conventional IR+AE QA architecture. This problem is manifest in a particular class of challenging questions, which I call *Multi-Evidence questions*. I give the definition and examples of this class of questions and contrast it with the more usual Single-Evidence questions. In the next section, I explain as to why Multi-Evidence questions pose special challenges to the question answering systems that employ information retrieval in the "conventional" way, namely IR as document pre-fetch based on the usual passage/document indexing. The possible strategy that such a question answering system might adopt in dealing with these questions is formulated and tested using 41 questions of this type from a quiz site. This is described in Section 6.3.1. Using this result as a baseline, I evaluate and compare the performance of Topic Indexing and Retrieval method on the same set of questions, described in Section 6.4, and conclude that the latter is superior both in the correctness of the answers and the overall efficiency in dealing with MEQs.

6.1 Distinguishing Questions based on Evidence Locality

Current factoid Question Answering depends on the existence of at least one passage or text span in the corpus that can serve as sufficient evidence for a question because of both its content (so as to be able to extract the answer from it) and the way it

is expressed (so as to be able to identify the passage as evidence in the first place). Reliance on a particular passage to answer a question renders the task of question answering essentially a *local* operation with respect to the corpus as a whole. Whatever else is expressed in the corpus about an entity being questioned is ignored, or used (in the case of repeated evidence instances of the same answer candidate) only to increase confidence in particular answer candidates. This means that *factoid questions whose correct answer depends jointly on textual evidence located in different places in the corpus cannot be answered*. I call this *the locality constraint*. There is a class of questions which needs the locality constraint to be overcome in order to be answered.

A single piece of evidence may suffice to answer a question, or more than a single piece may be needed. By "a piece of evidence", I mean a snippet of continuous text, a passage, that supports or justifies an answer to the question posed. More practically, in factoid QA, a piece of evidence is a text span with two properties: (1) An Information Retrieval (IR) procedure can recognise it as relevant to the question and (2) an automated extraction procedure can extract from it an answer-bearing expression (aka an *answer candidate*). With respect to the first of these properties, White and Sutcliffe (2004) have analysed 50 TREC 2003 questions with respect to their supporting evidence in the AQUAINT Corpus that justifies their answers. Their analysis shows that, for most questions (44 of them), there is sufficient evidence within a single sentence for an IR procedure to recognise its relevance. With respect to the second property, the relation between the question and each piece of evidence is either:

- Morphological: Either the support sentence is identical to the question in its normal form or differs only in inflectional or derivational morphology – e.g. "French(adj)'' and "France(Noun)''
- Lexico-Semantic: The terms contained in the support sentence bear some simple lexico-semantic relation such as hyponymy or meronymy relation to the terms contained in the question – e.g. "father'' and "parent''.

With respect to a given corpus, I call questions with these two properties *Single Evidence Questions* or SEQs:

> A question Q is a SEQ if evidence E sufficient to select A as an answer to Q can be found in the same text snippet as A.

In contrast, I call a question that requires multiple different pieces of evidence (in multiple text spans with respect to a corpus) a *Multi-Evidence Question* or MEQ:

> A question Q is a MEQ if the set of evidence E_1, ..., E_n needed to justify A as an answer to Q cannot be found in a single text snippet containing A, but only in a set of such snippets.

For example, consider the following question:

> Which country that has a female prime minister has hosted the World Cup? (5.1)

This question would be an SEQ with respect to a corpus which contains a sentence like "Germany, which for the first time in her history has elected a female Chancellor, has hosted the World Cup 2006." With respect to a different corpus that didn't mix sports with politics, there may be no one passage that provides evidence for both Angela Merkell's election in Germany and Germany's hosting of the 2006 World Cup. I believe that factoid QA can and should be able to answer such questions, whether the evidence is local to or distributed through the corpus.

This distinction between SEQs and MEQs lies only in the locality of evidence within a corpus. It does not imply that the corpus contains only one piece of text sufficient for an SEQ: Often there are multiple text snippets, each with sufficient evidence for the answer. Such redundancy is exploited by many question answering systems to rank the confidence of an answer candidate (e.g. Brill *et al.* (2002)) but the evidence is redundant rather than qualitatively different. List questions are also not necessarily MEQs even though they may make use of multiple passages in their answers because each answer may result from a single piece of evidence or from multiple pieces.

I stress that whether a question is a MEQ or not depends entirely on the corpus. However, the more syntactically complex a question, the more likely that it is a MEQ, given that a complex question will have more terms and relations that need to be satisfied. For example, a question of the form "What/Which <NBAR> <VP>?" such as *Which* [NBAR *African country*] [VP *has been chosen to host the World Cup?*] has at least two predicates/constraints that must be established, the one or more conveyed by the NBAR, and the one or more conveyed by the VP. These multiple restrictions might call for different pieces of evidence depending on the particular corpus from which the answer is to be found. For example, consider the question from the community-based Q&A site http://ask.yahoo.com:

> Which European cathedral is nicknamed the Toppled Elephant? (5.2)

The NBAR of its subject is "European cathedral" and its VP is "is nicknamed the Toppled Elephant". Even in the community provided answer, evidence for (1) a cathe-

dral (Notre-Dame-de-Fourviere Basilica) being named the Toppled Elephant and its location (Lyon France) being in Europe are not contained in the same sentence.

In judging whether a question is a MEQ or not, one more consideration must be given to whether a complex question with multiple restrictions, which have different textual support throughout the corpus, is indeed a MEQ when one of the restrictions uniquely picks out an entity. For example, in the question "Which vegetarian is the discoverer of the theory of special relativity?", one restriction,"the discoverer of the theory of special relativity" picks out a unique entity. Thus, it might be argued that this question can be sufficiently answered by a single piece of evidence, namely a sentence that states that Albert Einstein is the discoverer of the theory of relativity, and therefore is not a MEQ. This would be indeed true, if we know already that there is only one discoverer of the theory of special relativity. But how is it possible for a QA system to know that beforehand?[1] Even if the system was able to come up with only one entity ('Albert Einstein') as meeting this restriction, as there is no way of knowing whether there could be other entities that can satisfy this restriction, it has to check whether the other restriction, namely that X is a vegetarian, is met as well to have an absolute confidence about this answer. For example, this point becomes obvious if we change the question to "Which meat-lover is the discoverer of the theory of relativity?" Then the fact that one entity uniquely meets one restriction does not entail that that is the right answer. The right answer here would be none. Thus, unless the question implies logically that there is only one entity that meets one of the restrictions, a question must be considered a MEQ if it meets the criteria already stated for MEQs.

In database QA, MEQs correspond to queries that involve *joins* (usually along with *selection* and *projection* operations). The database equivalent of question 5.2 might involve joining one relation linking the names of cathedrals with the city they are located in, another linking cities with the countries they belong to, another linking countries with the continent on which they're found, and another linking the names of cathedrals with their nicknames. Note that this involves breaking up the original query into a set of *sub-queries*, each answerable through a single relation. Answering a MEQ through subqueries therefore involve the fusion of answers to different questions. Fusion here simply means a *natural join* operation, with the the answer entailed by a conjunction of all pieces of evidence. This primitive "conjunction introduction'' operation (where one can infer from the separate propositions A and B that the conjoined proposition

[1] And in fact, the Dutch physicist Hendrik Lorentz has a strong claim to have come up with an identical theory independently. It is not known whether he was vegetarian.

A&B follows) is the only kind of inference I am discussing in this work.

The problem, of course, is that free text does not contain systematic, well-defined relations that can be formally joined. This is the problem of locality constraint. One solution is the subject of the next section. This will be contrasted with the Topic Indexing and Retrieval based solution.

6.2 Solving MEQs with Conventional IR+AE: A Strategy for Multi-Evidence Questions

In this section, I present a solution for the MEQs for the conventional IR+AE architecture for the purpose of comparing the performance implementing this solution to that of the Topic Indexing and Retrieval based one in Section 6.4.

The solution I explore here for answering MEQs with a conventional IR+AE architecture can be adopted by any existing system for factoid QA that also has this architecture. It involves (1) dividing the original Multi-evidence question into multiple partial sub-questions, each of which yields a rank-ordered list of answer candidates, and (2) applying appropriate operations to globally rank-order answer candidates from these sets and choose the best. More specifically, this strategy involves:

1. Dividing a MEQ into sub-questions SQ_1, ..., SQ_n, each of which is a simple question about the same question variable.[2]
2. Finding the answer candidate(s) for each sub-question, SQ_i.
3. Selecting the best overall answer from the answer candidates for the sub-questions.

6.2.1 Dividing MEQ into sub-questions

Decomposing a MEQ into simpler sub-questions about the question variable involves:

- Resolving all co-referring expressions within the question;
- Breaking up the question at clause boundaries (including relative clauses);

[2]I have only considered sub-questions that are *naturally joined* on the question variable: I have not considered ones that require joins on other variables as well, as would be the case if Question 5.1 above also required a join, say, on a relation linking a person's name and their gender. Extending to the equivalent of joins on several variables would require an even more complex strategy for handling the answer candidate sets than the one I describe in Section 6.2.2.

- Within a single clause, if there is a conjoined set of restrictors (e.g. " ... German physician, theologian, missionary, musician and philosopher .."), copying the clause as many times as the number of restrictors, so that each clause now contains only one restrictor;
- Finally, reformulating all the individual clauses into questions.

Some examples of MEQs which have been factored into sub-questions are as follows. For the full list of questions used in the evaluation, refer to Appendix **??**[3]:

- Which French-American prolific writer was a prisoner and survivor of the infamous Auschwitz German concentration camp, Chairman of the U.S. President's Commission on the Holocaust, a powerful advocate for human rights and a recipient of the Nobel Peace Prize?
  1. Who was a French-American prolific writer?
  2. Who was a prisoner and survivor of the infamous Auschwitz German concentration camp?
  3. Who was a Chairman of the U.S. President's Commission on the Holocaust?
  4. Who was a powerful advocate for human rights?
  5. Who was a recipient of the Nobel Peace Prize?
- She was an individual held in high esteem in her community, but on Aug 4th, 1892 in Fall River Massachusetts, that reputation became tarnished for she was charged with murder. Who is she?
  1. Who was an individual held in high esteem in her community?
  2. On Aug 4th, 1892 in Fall River Massachusetts, whose reputation became tarnished for she was charged with murder?

It should be clear that factoring a MEQ into a set of sub-questions is often tricky but doable. The intra-sentential anaphora resolution can be less than straight-forward as the second example above shows, where one of the anaphora resolution involves relative clause. Also, the second sub-question to the second example could have been broken into a further sub-question. So often, it is a matter of choice as to how to break

[3]These questions are from a pub-quiz site http://www.funtrivia.com

a question into how many sub-questions and at what boundaries. In the evaluation reported in Section 6.3, the MEQs were manually broken into sub-questions, based on the procedure outlined above, since my focus was on recomposing the answers to sub-questions rather than decomposing questions. My results may thus serve as an upper-bound to what a fully automated procedure would produce. (For work related to the automatic decomposition of a complex questions, see Katz *et al.* (2005).)

6.2.2 Selecting an answer to the MEQ from sub-question answer candidates

From the answer candidates to the sub-questions, an answer must be derived for the MEQ as a whole. The most obvious way would be to pick the answer candidate that is the answer to the largest number of sub-questions. So if a MEQ had three sub-questions, in the ideal case there would be one answer candidate that would be the answer to all of these sub-questions at the same time and thus be the most likely answer to the MEQ. This is like a simple voting scheme, where each sub-question votes for an answer candidate, and the candidate with the most votes win.

Complications from this ideal situation arise in two ways: First, a sub-question can be very general, because it results from separating away other restrictions from the question. Thus it becomes a list question, such as in Example 5.1 "What country has a female prime minister?" with multiple true answers. Hence, the answer to a sub-question must be regarded instead as a set of answers rather than an individual answer, several of which may be correct. In other words, a sub-question can vote for multiple candidates rather than just one.

Second, simply maximising overlap (i.e. the number of sub-questions to which an answer candidate is the answer, which I call simply **Votes** from now on) ignores the possibility that multiple answers can tie for the same maximum Votes, especially if this number is small compared to the number of sub-questions. Thus, there is the need for a method to rank the answers with the same number of Votes. In summary, a sub-question can vote for multiple candidates and more than one candidates can receive the same largest number of votes from the sub-questions. This means I need an additional way to break the ties among the candidates by ranking the candidates according to some other criteria. The answer selection method I have chosen is described below.

Let's assume that a question has been factored into N sub-questions, and each sub-question is answered with a (possibly empty) set of answer candidates. So for the set

of N sub-questions for the original question, there are N answer candidate sets. The most likely answer would be the answer candidate shared by all the N sub-questions (i.e. the answer candidate present in all the N sets of answer candidates.). To see if there is such a common answer candidate, these N sets of answer candidates are first intersected (via generalised intersection).

If the intersection of these N sets is empty (i.e., there is no one answer candidate that all the sub-questions share.), then it must be investigated whether there is a common answer candidate for N-1 sets of answer candidates (i.e. an answer shared by N-1 sub-questions.). There will be N cases to consider since there will be N cases of N-1 sub-question sets to intersect. If all these are empty, then all subsets of size N-2 are considered, and so on, until a non-empty intersection is obtained. This means considering the *power set* of the original set of answer candidate sets.

This process may result in one answer candidate or several with the same maximum number of Votes. If there is only one, this is chosen as the answer. Otherwise, there is a need for a further way to rank these answer candidates to produce the most likely as the answer. Specifically, if there is more than one non-empty intersection with the same maximum Votes, I do the expected thing of taking into account the original ranking of answer to the sub-questions. If an answer is found to be the second ranked answer to one sub-question and the sixth ranked answer to another, its overall rank is taken to be the simple mean of the ranks. So in this case, the answer would have the rank four overall. This algorithm can be more formally stated as follows[4].

- Step 1: Let S be the set of the sets of answers, A_1, .. , A_n, to the sub-questions, Q_1 .. Q_n respectively.

- Step 2: Produce the power-set of S, i.e. $P = POW(S)$.

- Step 3: Produce a set of ordered pairs, V, such that $V = \{\langle o, R\rangle \mid R \subset P \wedge \forall x \in R.\ |x| = o$ for every distinct $o = |y|$ of every $y \in P\}$

- Step 4: Pick the ordered pair, L such that $L = \langle o, R\rangle \in V$ with the largest o among the members of V, and do:

    1. Produce T from R such that $T = \bigcup\{x \mid x = \bigcap t$ for every $t \in R\}$. (Note that t is a set.)

[4]This will be easier to understand if considered together with the example that follows.

2. If R‘ is an empty set, repeat this step 4 for the ordered pair with the next largest o.
3. Else if T has a unique member, then pick that unique member as the answer and terminate.
4. Otherwise, go to the next step.

- Step 5: For each member, $x \in T$, get the ranks, r_1, .. , r_n in the sub-questions, Q_1 .. Q_n and compute the mean M of the ranks.
- Step 6: Pick the member of T with the lowest M score as the answer.

To illustrate the steps described above, consider a MEQ M that can be split into three sub-questions Q1, Q2 and Q3. Then:

- Step 1: Assume that the sets A1, A2, A3 are the answer candidate sets for sub-questions Q1, Q2 and Q3 respectively:

  A1 = {CLINTON, BUSH, REAGAN}
  A2 = {MAJOR, REAGAN, THATCHER}
  A3 = {FORD, THATCHER, NIXON}

  Let S = {A1,A2,A3}

- Step 2: P = POW(S) =
  {{A1,A2,A3},{A1,A2},{A2,A3},{A1,A3},{A1},
  {A2},{A3},{}}

- Step 3:

  V = $\{\langle 3,\{\{A1,A2,A3\}\}\rangle,$
  $\langle 2,\{\{A1,A2\},\{A2,A3\},\{A1,A3\}\}\rangle,$
  $\langle 1,\{\{A1\},\{A2\},\{A3\}\}\rangle,\langle 0,\{\{\}\}\rangle\}$

- Step 4:

  First pick $L = \langle o,R\rangle = \langle 3,\{\{A1,A2,A3\}\}\rangle$ based on the largest o in consideration (i.e. 3).

  Then, get T such that $T = \bigcup\{x \mid x = \bigcap t \text{ for every } t \in R\} = \bigcup\{A1 \cap A2 \cap A3 = \{\}\}$

  Since T is an empty set, no answer can be picked. So repeat this step for the ordered pair with the next largest Votes.

- So again Step 4:

  The second pick $L = \langle o, R \rangle = \langle 2, \{\{A1, A2\}, \{A2, A3\}, \{A1, A3\}\}\rangle$

  Now get T by $T = (A1 \cap A2) \cup (A2 \cap A3) \cup (A1 \cap A3) = \{\text{REAGAN}, \text{THATCHER}\}$

  Since R' is non-empty, which at the same time does not have a unique member, go to the next step.

- Step 5:

  REAGAN: 4th candidate for Q1 and 10th in Q2 – average rank 7

  THATCHER: 1st candidate for Q2 and 5th in Q3 – average rank 3

- Step 6:

  Take the candidate with the highest average rank as the answer – here, THATCHER.

6.3 Evaluation of the Sub-question Strategy

6.3.1 Evaluation Questions

I chose 41 "pub quiz" questions (i.e. ones designed to challenge and amuse people) as likely Multi-Evidence Questions (MEQs) with respect to the AQUAINT corpus. The questions come from http://www.funtrivia.com/, and were chosen based on the following criteria:

- Did the question contain multiple restrictions of the entity in question?
- Did the answer appear in the AQUAINT corpus?
- Was the answer a proper name?
- Was it a difficult question? (Questions at funtrivia.com have been manually marked for difficulty.)

The questions, varying in length from 20 to 80 words[5], include:

What was the name of the German physician, theologian, missionary, musician and philosopher who was awarded the Nobel Peace Prize in 1952?

Which French-American prolific writer was a prisoner and survivor of the infamous Auschwitz German concentration camp, Chairman of the U.S.

[5]This is long compared to TREC questions, whose mean length is less than 10 words.

President's Commission on the Holocaust, a powerful advocate for human rights and a recipient of the Nobel Peace Prize?

He was born in 1950 in California. He dropped out of the University of California at Berkeley and quit his job with Hewlett-Packard to co-found a company. He once built a "Blue Box" phone attachment that allowed him to make long-distance phone calls for free. Who is he?

But questions 5.1 and 5.2 above show that less complex, more realistic examples can easily be found of factoid questions that require joining multiple pieces of evidence with respect to a corpus.

6.3.2 The Evaluation Setting

The document collection used for the evaluation is the same AQUAINT corpus, which I also used for TREC question evaluations in Chapter 5. The evaluation has two purposes. The first is to see how well the sub-question strategy specified in Section 6.2 works, as a baseline for future development of sub-question approaches. The second is to see how well a sub-question strategy that can be carried out in the standard IR+AE systems compared to a Topic Indexing and Retrieval based one. Toward that end, the target system on which this method is tested would ideally be a conventional question answering system that relies on local passage based answer extraction method adhering to the IR+AE principle. Additionally, it would have the added ability to divide a question into sub-questions and to combine the answers of the sub-questions to produce the answer to the whole question as explained in the earlier section.

As this was not available, I used as a base QA engine the same Nexus system for this evaluation. As presented in Section 5.6.1, Nexus is not a conventional question answering system and its answer extraction method is not based on local passages. However, it has good performances on the more usual type of questions as demonstrated in Chapter 5 (MRR of 0.306 for TREC questions (TOQA-A)). Thus, for generating answers for the sub-questions of a MEQ (which are SEQs) it can be used as an ersatz conventional QA system without prejudicing the experiment. Thus, the experiment *simulates* a conventional question answering system with a modification specifically to accommodate MEQs by way of sub-question strategy. A similar evaluation could be carried out with any other QA system, since the point is to compare a system's ability to answer a MEQ based on the standard whole-question IR strategy (Section 6.1) with its ability to do so based on factoring the original question into sub-questions and then "joining" the results as described in Section 6.2.

QID	SQ	SQA	QID	SQ	SQA
1	2	1	**22**	2	1
2	4	0	**23**	3	3
3	4	1	**24**	2	1
4	3	0	**25**	3	3
5	3	0	**26**	5	3
6	3	1	**27**	2	2
7	4	1	**28**	2	1
8	3	1	**29**	2	2
9	5	2	**30**	3	3
10	4	1	**31**	2	2
11	4	0	**32**	2	2
12	3	1	**33**	5	0
13	2	2	**34**	4	0
14	2	0	**35**	5	0
15	3	0	**36**	9	1
16	3	1	**37**	2	1
17	2	2	**38**	8	1
18	4	1	**39**	6	0
19	2	0	**40**	9	3
20	2	2	**41**	9	0
21	3	2			

Table 6.1: Raw Results

The evaluation procedure ran as follows. First, a MEQ was factored into a set of sub-questions manually. Then each sub-question was fed into the QA engine to produce a set of 100 answer candidates. The resulting sets of answers were assessed by an Answer Selection/Ranking module that uses the algorithm described in Section 6.2.2 to produce a final set of ranked answer candidates.

6.3.3 Results and Observations

Table 6.1 contains the raw results for the 41 questions (QID), showing how many sub-questions (SQ) each was factored into, and how many of these had at least one correct

QID	SQ	MaxV	ACI	Rank	QID	SQ	MaxV	ACI	Rank
1	2	1	0	0	**23**	3	3	1	1
3	4	1	0	0	**24**	2	1	0	0
6	3	1	0	0	**25**	3	3	1	1
7	4	1	0	0	**26**	5	3	1	1
8	3	1	0	0	**27**	2	2	23	1
9	5	2	6	3	**28**	2	1	0	0
10	4	1	0	0	**29**	2	2	86	1
12	3	1	0	0	**30**	3	3	3	1
13	2	2	2	1	**31**	2	2	9	5
16	3	1	0	0	**32**	2	2	1	1
17	2	2	1	1	**36**	9	1	0	0
18	4	1	0	0	**37**	2	1	0	0
20	2	2	4	1	**38**	8	1	0	0
21	3	2	7	1	**40**	9	3	1	1
22	2	1	0	0					

Table 6.2: Questions with answer candidates identified for $\geq$1 sub-questions.

answer within its answer candidates (SQA). For 12 of the evaluation questions, the QA engine found no correct answer candidates (SQA=0), so these questions are ignored in the rest of the evaluation. (That is, for these, the sub-question strategy works no better than the standard whole question strategy.) For the remaining 29 questions, Table 6.2 shows the value of ranking (i.e., Step 5 above) when more than one answer candidate shares the same number of largest Votes. Such answer candidates need to be ranked in order to pick the best answer.

In Table 6.2, column **MaxV** indicates the largest number of Votes received by an answer candidate across the set of sub-questions. (This turns out to be the same as SQA in Table 6.1, indicating that, for this set of questions, no wrong answer has more Votes than a correct answer.) Column **ACI** indicates the number of answer candidates with this number of Votes. For example, MEQ 9 has 5 sub-questions. Its **MaxV** of 2 indicates that only the intersection of the answer candidate sets for two sub-questions (out of maximum 5) produced a non-empty set (of 6 members according to **ACI**). These 6 members are the final answer candidates that will be ranked.

The column labelled **Rank** indicates the ranking of final answer candidates by mean ranking with respect to the sub-questions they answer and identifies where the correct answer lies within that ranking. So for Question 9, the correct answer was third in the final ranking. Fifteen (15) questions had no answer candidates common to any set of sub-questions (i.e., **ACI**=0). Of the 14 remaining questions, six (6) had only a single answer candidate, so ranking was not relevant. Of the final eight (8) questions with $\geq$1 final answer candidates, Table 6.2 shows that ranking them according to the mean of their original rankings led to the correct answer being ranked first in six (6) of them. In the most extreme case, starting with 86 ties between answers for two sub-questions (all of which have a set of 100 answers), ranking reduces this to the correct answer being ranked first. Table 6.2 also shows the importance of the proportion of sub-questions that contain the correct overall answer. For MEQs with 2 sub-questions, in every case, both sub-questions needed to have contained the correct overall answer in order for that MEQ to have been answered correctly. (If only one sub-question contains the correct overall answer, my algorithm has not selected the correct overall answer.) For MEQs with 3 sub-questions, at least two of the 3 sub-questions need to have contained the correct overall answers in order for the MEQ to be answered correctly by this strategy. This seems to be less the case as the number of sub-questions increases. However, the strategy does seem to require at least two sub-questions to agree on a correct overall answer in order for a MEQ to be correctly answered. It is

possible that another sub-question ranking strategy could do better.

A@N	SQ-Combined
1	0.317:13
2	0.317:13
3	0.341:14
4	0.366:15
5	0.366:15
6	0.366:15
7	0.366:15
8	0.366:15
9	0.366:15
10	0.366:15
15	0.366:15
20	0.366:15
ACC	0.317
MRR	0.341
ARC	1.333

Table 6.3: Results for Sub-question Strategy

In summary, starting with 41 complex factoid questions that could not be answered in the AQUAINT corpus without pulling together multiple pieces of information (and hence could not be answered using a standard "complete question" strategy), the "sub-question" strategy found 14 with at least one "intersective" answer candidate. Of those 14, the top-ranked candidate (or, in 6 cases, the only "intersective" candidate) was the correct answer in 12 cases. Hence the "sub-question" strategy has succeeded in 12 of 41 cases that would be totally lost to any "complete question" strategy. Table 6.3 shows the final scores with respect to A@N, accuracy, MRR and ARC.

6.3.4 Discussion

I have shown that the strategy based on sub-question decomposition and ranking can be used for answering Multi-Evidence Questions, which cannot be answered using standard factoid QA on the question as a whole. But I also need to consider (1) the cost of adopting this strategy as a practical method for doing real time Question Answering

and (2) possibly more effective strategies for answering MEQs. These two issues are naturally related.

With respect to cost, clearly multiplying the number of questions to be answered by decomposing a question into sub-questions can make the overall task even more resource-intensive than it is already. Pre-caching answers to simple, frequent questions, as in Chu-Carroll *et al.* (2002), which reduces them to database look-up, may help in some cases. Another compatible strategy would be to weight the sub-questions, so as to place more emphasis on more *important* ones or to first consider ones that can be answered more quickly (as in database *query optimisation*). This would avoid, or at least postpone, very general sub-questions such as "Who was born in 1950?" with a large number of candidate answers and thus expensive to process. One might rather consider more highly weighted and specific sub-questions first. If the system has high confidence in the answers to these questions (so as to render other candidates unlikely), then less important questions may not have to be processed at all, saving significant computing resources. An appropriate method for weighting sub-questions is an issue that needs further investigation, but even before that there is a need for a method that can reliably factor a MEQ into simple sub-questions automatically. These, however, would not be trivial to develop.

Topic Indexing and Retrieval for QA offers an alternative that does not require the factoring of a question in the first place, which is the subject of the next section.

6.4 Evaluation of MEQ Performance for Topic Indexing and Retrieval Method

This section presents an evaluation of how Topic Indexing and Retrieval Method performs on MEQ questions. The purpose of the evaluation is to see how well this method can deal with this type of questions, particularly as compared to simulated IR+AE system with a sub-question strategy as presented in the previous section.

Topic Indexing and Retrieval enables a direct approach to Multi-Evidence Questions: There is no need for any special strategy of for question factoring. With respect to a topic document, a possible answer is already associated with all the distributed pieces of evidence about it in the corpus: In other words, MEQs are exactly the same as SEQs.

The Figure 6.1, illustrates the differece between the Topic Indexing and Retrieval

based approach and the more conventional Document Retrieval based approach with the special sub-question strategy. Here, to the question, "Which US senator is a former astronaut?" the conventional approach requires different set of textual evidence (documents) to be assembled based on the decomposition of the question, whereas in the Topic Indexing and Retrieval approach, only one topic document needs to be located.

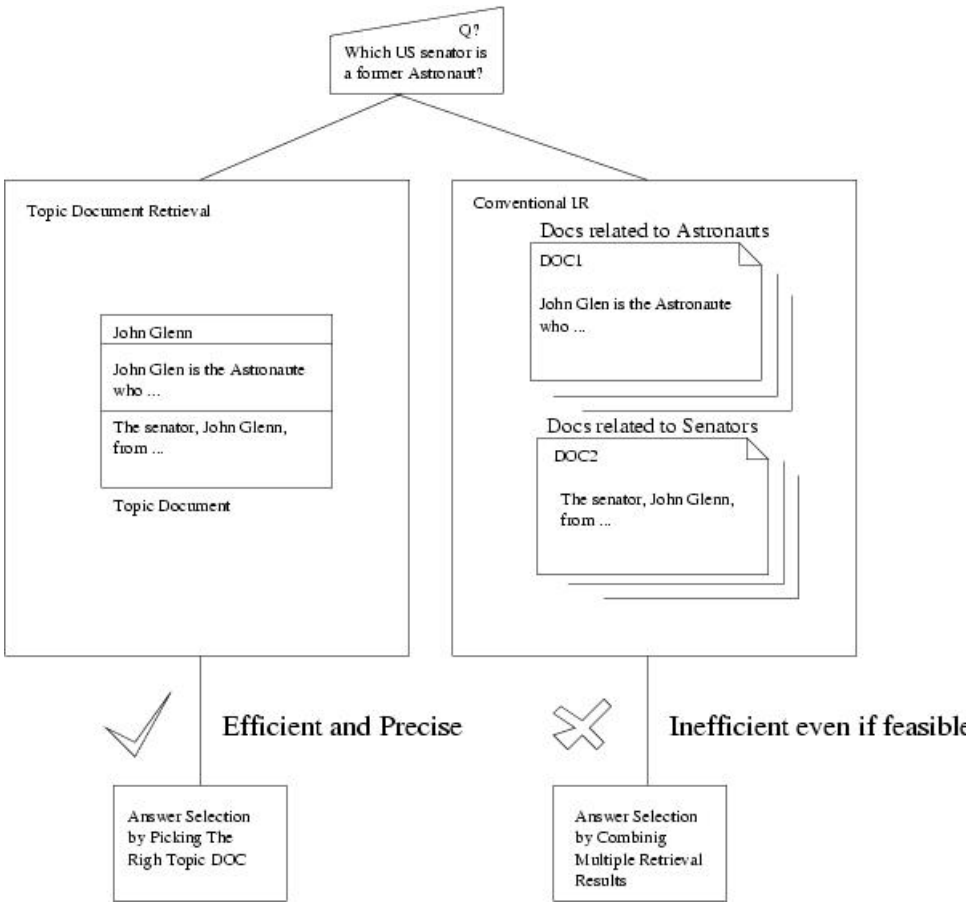

Figure 6.1: Topic Indexing and Retrieval vs IR+AE for MEQs

Thus there are three advantages for using Topic Indexing and Retrieval method over the conventional IR with the special sub-question strategy:

- No multiplication of questions.
- No need for question factoring.
- No need to combine the answers of the sub-questions.

6.4.1 The Evaluation Setting

The base question answering engine is the same, Nexus, as with the evaluation of the sub-question strategy but there are no question factoring pre-processing or complex answer candidate post-processing. The particular configuration that I used for my test is the same as the TOQA-A as in Section 5.4.2.[6]

The system simply retrieves answer candidates and the top 20 answers are taken for the score assessment. The questions used for evaluation are naturally the same as for the evaluation in the previous section.

6.4.2 Results and Comparison to IR+AE

A@N	SubQ-St	TopicInd
1	0.317:13	0.317:13
2	0.317:13	0.512:21
3	0.341:14	0.585:24
4	0.366:15	0.634:26
5	0.366:15	0.659:27
6	0.366:15	0.659:27
7	0.366:15	0.659:27
8	0.366:15	0.659:27
9	0.366:15	0.659:27
10	0.366:15	0.659:27
15	0.366:15	0.659:27
20	0.366:15	0.659:27
ACC	0.317	0.317
MRR	0.341	0.456
ARC	1.333	1.889

Table 6.4: Comparison of the Scores

Table 6.4 summarises the results of the evaluation for the Topic Indexing and Retrieval system (TopicInd) compared to the Sub-question Strategy based system (SubQ-St).

[6]The reason that TOQA-B is not used is that the fine-answer type identification is too complicated with complex questions.

The TopicInd system has returned more correct answers than the SubQ-St system (in all cut-off points except having tied in number 1 rank). Also all the correct answers that were found by the SubQ-St system were also found by the TopicInd system irrespective of the ranks. Twelve correct answers that were found by TopicInd (at A@5) were missed by the SubQ-St system. Among those answers that were found by both systems, SubQ-St produced better rank in 5 cases whereas in only one case the TopicInd produced a better rank. However, Table 6.4 shows that TopicInd in general found more correct answers (27 vs 15) in total, and more answers in higher rank (e.g. top 2) than SubQ-St system. The difference is statistically significant according to Wilcoxon Matched Signed Rank Test: W+ = 17.50, W- = 172.50, N = 19, p = 0.0007896. Hence, it is possible to conclude that Topic Indexing and Retrieval method is superior to the simulated IR+AE method with special sub-question strategy for this type of questions.

6.4.3 Discussion

The results of the evaluation shows not only that the performance of the Topic Indexing and Retrieval method is superior to the IR+AE based method that uses a special strategy, but also that it is simpler: it doesn't need any special strategy or tricky operation such as question factoring. Further more, there is no efficiency penalty: a MEQ is treated as one question rather than several sub-questions.

One final note, the fact that Topic Indexing and Retrieval Method has superior performance is no surprise considering that the more the evidence from the question for an answer, more information is available for the method in matching the relevant topic document of a candidate answer to the question. This is in contrast to the IR+AE based systems, which, without a special strategy such as the sub-question strategy discussed here, require that whatever evidence exist in a question must be found within a short passage in the corpus due to the locality constraint.

6.5 Summary and Conclusion

This chapter investigates whether the problem of the locality constraint mentioned in Chapter 4 does really pose a problem for the conventional QA architecture adhering to the IR+AE pipeline. Multi-Evidence Questions are introduced as the kind of questions for which the locality constraint bear out its constraining effect. A special strategy is devised in order to overcome this constraint with the conventional QA architecture.

This involves partitioning a question into a set of simpler questions. The results show that the strategy is successful to a degree in that some questions are indeed correctly answered but it also comes with a cost: A MEQ is multiplied into several vague questions via factoring, which is not trivial, and a special method is needed to combine the answers to sub-questions.

On the other hand, Topic Indexing and Retrieval method does not need any special strategy or tricky question factoring in dealing with MEQs. There is no performance penalty for MEQs: The method is efficient in dealing with these questions as MEQs are treated as just any questions. Against the baseline of the IR+AE with the special strategy, it was superior in overall correctness of answers. All in all, this chapter clearly shows the advantage of Topic Indexing and Retrieval method with respect to questions requiring distributed evidence in the corpus: The locality constraint does not apply to Topic Indexing and Retrieval method.

As there were no real MEQs in TREC, there is no evaluation data available from other systems for this type of questions. However, most systems that have participated in TREC have more or less conventional IR+AE architecture, and therefore it can be inferred that they would have the same difficulties for dealing with MEQs as the baseline system presented in this chapter. TOQA mehod is therefore more advantagous for MEQs.

Chapter 7

Conclusion

This chapter concludes this thesis by revisiting the main claims regarding the central method, *Topic Indexing and Retrieval for QA*, verifying them with respect to the experimental results obtained. The satisfaction of these claims indicates that the method is not without advantages, for the method affords a very efficient Question Answering for questions with proper name answers as well as a better use of the textual evidence present in a corpus.

However, the method, as it is currently realised, is not without limitations, some of which are discussed as the possible areas for Future Work. Finally, the thesis is concluded with a final remark on the contributions of this work to the field of Question Answering in general.

7.1 Reviewing the Claims of the Thesis

The main claims of the thesis as laid out in Chapter 1 regarding the thesis' central method, *Topic Indexing and Retrieval*, are the following:

1. Topic Indexing and Retrieval enables direct retrieval of answers using Information Retrieval technique for questions with proper name answers.

2. This method can handle questions which require distributed evidence more effectively as compared to the conventional QA architecture based method.

In order to verify the first of these claims, I have performed evaluations on TREC questions using *Nexus* QA system that implements the method as presented in Chapter 5. The evaluation results of 0.340 in Accuracy and 0.395 in Mean Reciprocal Rank

(TOQA-R) demonstrate that the method has a reasonably good performance on the questions with proper name answers as compared to a baseline system that is based on a more conventional IR+AE architecture with some of the more advanced IR features. (0.186 in accuracy and 0.245 in MRR for ANS-RR). The speed advantage over the baseline system also shows that the method is indeed far more efficient (less than 1 second per question vs. more than 2 minutes). These results hence substantiate the point that proper name factoid QA can be turned into a very fine-grained (proper name) information retrieval with the efficiency benefits that it provides.

Regarding the second claim, in Chapter 6 I have assembled 41 Multi-Evidence Questions (MEQs), and these were used to evaluate and compare the performances between a simulated conventional QA system with a MEQ tailored strategy and the Topic Indexing and Retrieval method based system without any modification. The superior results of the latter (MRR of 0.454 against 0.341) demonstrates that the method is indeed effective for MEQs: While none of the tricky operations that the former needed viz the sub-question strategy involving question factoring, answer combination, etc. were required, the accuracy and core efficiency of the method were not compromised as a MEQ was, under this method, not distinct to any other questions. Hence, it can be concluded that the two claims that I have made regarding the method are justified. This means that Topic Indexing and Retrieval for QA has some definite advantages over the more conventional IR+AE pipeline based Question Answering method. These advantages are due to the following factors.

First, by deriving the domain of all possible answers (topics) from a corpus beforehand, it is possible to pre-gather and associate all the relevant information about every potential answer contained in a corpus during pre-processing. This not only saves time during the on-line QA operations, but it also enables the examination of each relevant answer candidate with respect to the whole evidence present in the corpus. The more conventional QA system has to extract the eventual answer from a piece of text, a process that depends on retrieving the right piece of evidence in the first place and then extracting an answer from that evidence with further operations. Since the evidence consists of individual passages, each of which serves as an independent evidence by default, the *locality* constraint is inherent in this kind of Question Answering methodology. The overcoming of this locality constraint for MEQs is possible only with laboured efforts as exemplified by the sub-question strategy that involved tricky question factoring, answer sets combination and multiplication of one QA task into several subtask (each sub-question requiring one QA operation) as shown in Chapter 6.

Second, because potential answers (topics) are extracted off-line, a comprehensive named entity type information for each topic can be gathered (from an external source such as YAGO and WordNet) and stored with the full ISA hierarchy beforehand, which can simply be looked up via the topic-type hash table during on-line processing. In Chapter 5, it was shown how the fine answer-type based reranking improved the accuracy of the answers. This pre-identification of named entity type of each topic also enables the building of separate indices (for base topic types) further increasing the retrieval efficiency.

Third, not only is all the evidence about each topic gathered into a topic document but also every variant expression expressing the same fact about this topic is entered into it. This explains why the method performs as well as it does without any sophisticated syntactic or semantic operations (such as paraphrasing, textual entailment inference, etc).

Finally, document retrieval is a mature technology that is very scalable and efficient. By putting the evidence in the form of documents (topic documents) and using IR technique to index and retrieve these topic documents, the QA task becomes a Document Information Retrieval task resulting in efficient answer retrieval.

7.2 Future Work and Development

Despite the strength of the method outlined thus far, it is not without weaknesses and limitations. Some of theses have to do with the current implementation of the systems, and others have to do with limitations of the method itself. In this section, these issues are investigated as to what these limitations are and whether the method can be extended to deal with these limitations and how.

7.2.1 Textual Support

The current implementation of the method, Nexus, does not provide any textual support (i.e. justification) to the set of answer candidates it generates for a question. Since QA systems are not perfect, and the human user would want to verify the answers against the textual basis on which they were selected by the system (Lin *et al.*, 2003), it is desirable to present such textual support to each answer candidate. For example, to the question “Where was Einstein born?”, a text snippet such as “Einstein was born in Ulm.” would verify whether the answer candidate, “Ulm”, is in fact a good answer for

the user.

This can be done via an information retrieval operation on the source corpus after the answer generation phase is complete, using the candidate answer with the question together as the query. This is often done in TREC, where a QA system actually relies on the web (with its vastly greater answer redundancy) for the actual answer generation, but uses the AQUAINT corpus for the supporting document generation as required by TREC (Banko *et al.*, 2002). So in a similar fashion, Nexus can use the topic document indices for answer generation but use the original AQUAINT corpus for supporting document generation. Another possibility is to use the topic documents themselves. The topic document corresponding to an answer candidate topic can be tiled on-line in a similar fashion to the snippet generation process for web search engines. This tiling operation would look for the piece of text (a sentence) that contains the most keywords from the question and presents it as the support. For MEQs, a multiple sentences would be selected each containing some partial evidence.

7.2.2 Non-Proper Name Answer Types

The method of Topic Indexing and Retrieval cannot deal with questions that require answers consisting of something other than simple phrase such as those that require a whole sentence to answer. This is a limitation of the method itself, since a sentence or longer cannot be topics within the framework of the method. However, there are also some simple phrase answers that the method does not at present handle. These are dates, numerals (e.g. temperatures, quantities of things etc.), adjectives and common nouns such as the colour, ‘red’ or the quality of being ‘hot’, ‘spicy’ etc. The reason is that they are too generic, therefore having too much diverse information associated to them. For example, an adjective term lacks the essential topicality, which is the prerequest for Topic Indexing (e.g. ”spicy” is a generic quality lacking any topical significance). However, there is a way to accommodate these entities, namely via relation indexing. Although ‘spicy’ cannot be a topic by itself, the relation, (‘Kimchi’, ‘Spicy’) can be. The (‘Kimchi’, ‘Spicy’) topic would then contain all the sentences that mention these two items together, and indeed this could be used to answer question such as “What is the defining taste of Kimchi?”.

One way of indexing relations within the Topic Indexing and Retrieval framework would be via, what I call, bi-topic indexing. So constructing a bi-topic document containing two related topics (topics found within the same sentences) in contrast to a

single topic document will enable the retrieval of bi-topic relations. An example of a bi-topic document is the following Figure 7.1, which represents the two topics (Dolly (sheep), Ian Wilmut).

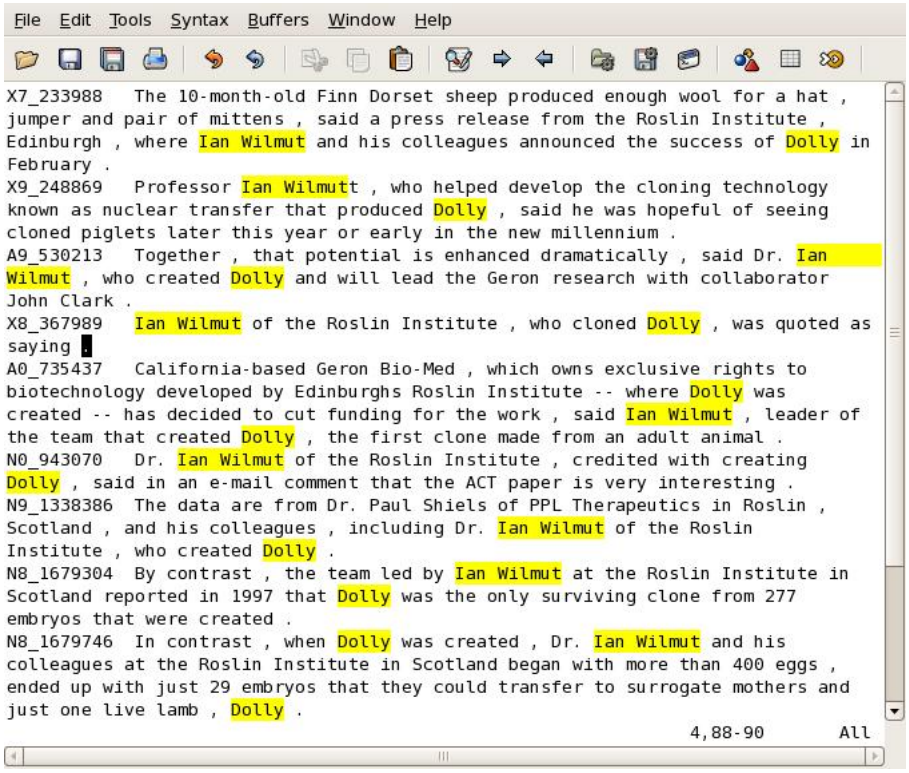

Figure 7.1: A Bi-Topic Document: (Dolly, Ian Wilmut)

Such a bi-topic document would represent the general relation between two topics via the context in which they co-occur, although the precise characterisation of the relation is not captured this way. The terms that more frequently appear in such document characterise the relation between the two topics in statistical fashion, and this document would be given a higher score for retrieval with respect to a question, if the question contains such a relatively frequently appearing term. For example, in scoring the bi-topic document pertaining to (Dolly, Ian Wilmut) bi-topic document with respect to the question, “Who created the fist cloned sheep, Dolly?”, the frequently appearing term in the document, ‘clone’ would give a very high mark for this document with respect to this question. One way of actually constructing the bi-topic document collection is to recursively apply the topic document method first on the original documents and then to the resulting topic documents. So given a single topic document, e.g. “Dolly” topic document, which itself is produced by applying topic document method to the original document collection, the same topic document generating process as described in Sec-

tion 4.2.2 is then applied to this topic document. Then a new set of topic documents will be generated that, in addition to having their own topics, e.g. "Ian Wilmut", will also contain the topic "Dolly" since the original topic document has the topic "Dolly" in its every sentence. The resulting bi-topic documents would be such as (Dolly, Ian Wilmut) bi-topic documents, (Dolly, Bonnie) bi-topic document, etc. Now all these topic documents are only with respect to the topic "Dolly" (hence bi-topic), which I call the *anchor topic*, and indexing these amounts to creating a "Dolly" (anchored) index. Separate indices for base types as in the case of the single topic documents need not be created since the number of bi-topic documents anchored to one topic is some magnitude smaller compared to the number of total single topic documents.

The chief problem of this approach would be the huge space requirement to accommodate all the bi-topic documents as the number of bi-topics would be quite great. One way of solving this problem is to construct the anchored topic index dynamically or even retrieving and scanning a single topic document (i.e. generating a bi-topic document dynamically) without creating an index (if the single topic document is small enough) online after a question has been entered and its question topic identified. This approach however will take up valuable online time. So for future work, an efficient way of accommodating bi-topic relations needs to be studied if Topic Indexing and Retrieval method is to be able to handle more generic types of answers.

7.2.3 Nil and List Questions

Nil answers (i.e. indicating that an answer isn't present in the corpus) are another type of answer that the method at present do not handle: It generates answer candidates no matter what. A simple thresholding with respect to the confidence of an answer candidate (the score of the answer candidate) to some particular value would be the most immediate way of handling nil questions. The problem with this is how to set such a threshold value. I have observed that the score of an answer candidate that is generated by the InQuery retrieval engine does not seem to correlate directly with any sort of confidence or probabilities in the answer. (Only the ranking matters) For example, an answer candidate would display a very low score even though it is the correct answer to a particular question (and ranked top), while another answer candidate to a different question, although wrong, might be scored very high (but ranked third for example): It seems that the scores generated by the IR engine have only relative values. The problem is not simply the case of a lack of understanding of the scoring formula, but rather

it is the general obscurities of such scores (Saggion *et al.*, 2004) especially when the query is complex as it usually is for QA.[1] A solution to this problem is to attach an independent validation mechanism for an answer candidate such as the Web Validation method (Magnini *et al.*, 2002a). This offers the additional benefit of cross-checking the validity of the answer with respect to an independent means. Since a QA system is usually a heterogeneous mixture of different and largely independent components anyway, there is no reason that the QA system that implements the core method in this thesis should not be augmented with such an addition (and indeed this will be pursued).

List questions are another type of questions that the method at present does not deal with. The reason is similar as for the nill questions: No threshold value can be established to pick the top N correct answers. The problem is compounded by the fact that certain reranking might have to be performed as the confidence and precision of an answer candidate decreases inversely to its rank. The core method does not offer any such mechanism and as in nil questions, an extra and independent means is required for list questions. In so far as building a good performing QA system goes, such an addition is greatly desirable, but for the present purpose of assessing the core method of Topic Indexing and Retrieval for QA, it suffices to mention that the method has no inherent means of handling list questions and nil questions.

7.2.4 Topic Disambiguation

In Section 4.2.1, it was mentioned that topic disambiguation, which is the task of distinguishing entities referred by the same name such as 'George Bush', has not been performed. One consequence of this is to conflate facts resulting in a topic document containing information pertaining to different entities under one topic. The direct impact of this has not been revealed by the evaluation performed in this thesis as there were no questions with ambiguous answers in the test set, but it is not hard to speculate the negative consequence this would have with questions for which the answers represent conflated topics. With the sort of bi-topic relation indexing mentioned in Section 7.2.2, there could be an inherent mechanism for topic disambiguation. For example, the bi-topic relations, (Edinburgh,Scotland) and (Edinburgh, Indiana) would contain only relevant information pertaining to the respective Edinburgh. However, this kind of disambiguation would have its limit as, for example, the bi-topic relation (Edinburgh, Walmart) might very well contain Walmarts of both cities indiscriminately.

[1] It doesn't help that most IR systems in practice use likelihoods rather than absolute probabilities to rank the documents.

For a more substantial disambiguation, document clustering / classification techniques from the general information retrieval field can be applied (e.g. Slonim and Tishby (2000)) to topics since under the Topic Indexing and Retrieval method, topics are in the form documents (topic documents). This is one area that definitely needs to be investigated.

7.2.5 Anaphora Resolution

Because anaphora resolution has not been performed on the source corpus, the implemented system (Nexus) missed out on quite a lot of potentially useful evidence. The benefits of anaphora resolution for QA in general are clearly documented by Morton (1999), Vicedo and Ferrández (2000) and Mur and van der Plas (2007). Mur and van der Plas (2007), in particular, makes use information extraction technique to store facts off-line for question answering. Here, it is reported that anaphora resolution has resulted in the increase of some 50 per cent in extracted facts for the use in QA. Although the method of the thesis does not rely on information extraction as such, this work shows that the addition of anaphora resolution to the core method would result in more content to be added to each topic document. Also, by doing anaphora resolution, a passage consisting of several sentences rather than one single sentence (as is currently the case) linked up by co-reference of the target topic could be extracted and merged into a topic document as evidence.

The effect of not doing anaphora resolution in Nexus is the assemblage of sentences in a topic document that are in some sense more generally 'typical' about the topic that is the target. Whenever an entity is referred co-referentially, then it has already been introduced previously. Considering that the textual source consisted of news paper articles (AQUAINT), the first sentence that mentions the entity with the full proper name usually contains some introductory information about the entity in order to familiarise the reader with the topic. This is all the more the case for person names. For example, ".. Tony Blair, the British prime minister, ... " is a very oft-occurring expressions found within the topic document "Tony Blair". As only the more typical information is associated to the given topic without anaphora resolution, I believe, there is a certain benefit to it when the question is of general InQuery about a topic, "Who is the British prime minister?". On the other hand, the more specific questions would require more detailed information about the entity in question, and the performance would suffer. So considering the net-effect whether it is advantageous to do anaphora resolution given

a big corpus with respect to the method of Topic Indexing and Retrieval needs further study.

7.3 Contributions and Summary

Factoid Question Answering is an exciting area of Natural Language Engineering that has the potential to complement today's search engines as the prevalent means of information access. In this thesis, I introduce a new method of handling factoid questions whose answers are proper names. The method, *Topic Indexing and Retrieval for QA*, addresses two issues that prevent current factoid QA system from realising this potential: They can't satisfy users' demand for almost immediate answers, and they can't produce answers based on evidence distributed across a corpus.

The first issue arises because the architecture common to QA systems is not easily scaled to heavy use because so much of the work is done on-line: Text retrieved by information retrieval (IR) undergoes expensive and time-consuming answer extraction while the user awaits an answer. If QA systems are to become as heavily used as popular web search engines, this massive process bottle-neck must be overcome.

The second issue of how to make use of the distributed evidence in a corpus is relevant when no single passage in the corpus provides sufficient evidence that it contains an answer to a given question. QA systems commonly look for a text span that contains sufficient evidence to both locate and justify an answer. But this will fail in the case of questions that require evidence from more than one passage in the corpus.

These two problems have a common cause: the employment of Information Retrieval as document pre-fetch. Retrieving passages for on-line processing limits the range and the amount of work that can be put into the process. The method of Topic Indexing and Retrieval for QA addresses these problems by shifting much of the work to preprocessing in such a way that it enables direct retrieval of answers using IR technique based on a more comprehensive use of the evidence present in a corpus. While this method can only deal with factoid questions with proper names at present, I believe that it has shown a promising future development potential.

Bibliography

Ahn, K. and Webber, B. (2007). Nexus: A real time qa system. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA. ACM.

Ahn, K., Bos, J., Clark, S., Curran, J., Dalmas, T., Leidner, J., Smillie, M., and Webber, B. (2004). Question answering with qed and wee at trec-2004. In *Proceedings of TREC'04*.

Androutsopoulos, I., Ritchie, G., and Thanisch, P. (1995). Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, **1**(1), 29–81.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

Banko, M., Brill, E., Dumais, S., and Lin, J. (2002). AskMSR: Question answering using the Worldwide Web. In *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*.

Bouma, G., Mur, J., van Noord, G., van der Plas, L., and Tiedemann, J. (2005). Question answering for dutch using dependency relations. In C. Peters, F. C. Gey, J. Gonzalo, H. Müller, G. J. F. Jones, M. Kluck, B. Magnini, and M. de Rijke, editors, *CLEF*, volume 4022 of *Lecture Notes in Computer Science*, pages 370–379. Springer.

Brill, E., Lin, J., Banko, M., Dumais, S., and Ng, A. (2001). Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*.

Brill, E., Dumais, S., and Banko, M. (2002). Analysis of the askmsr question-answering system. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, pages 257–264, Philadelphia PA.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the Sixth International World Wide Web Conference (WWW6)*.

Callan, J. P. (1994). Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994)*.

Chu-Carroll, J., Prager, J., Welty, C., Czuba, K., and Ferrucci, D. (2002). A multi-strategy and multi-source approach to question answering. In *Proceedings of the 11^{th} Text Retrieval Conference (TREC 10)*, National Institute of Standards and Technology.

Chu-Carroll, J., Czuba, K., Prager, J., and Ittycheriah, A. (2003). In question answering, two heads are better than one. In *Proceedings of the 2003 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2003)*.

Clarke, C., Cormack, G., and Lynam, T. (2001). Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*.

Curran, J. and Clark, S. (2003). Language independent ner using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natual Language Learning (CoNLL-03)*, pages 164–167.

Dang, H. T. (2006). Overview of duc 2006. In *Document Understanding Workshop (HLT-NAACL 2006)*.

Fleischman, M., Hovy, E., and Echihabi, A. (2002). Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL-2003)*.

Green, B., Wolf, A., Chomsky, C., and Laughery, K. (1961). BASEBALL: An automatic question answerer. In *Proceedings of the Western Joint Computer Conference*.

Hamblin, C. (1958). Questions. *Australasian Journal of Philosophy*, **36:159-168**.

Harabagiu, S., Moldovan, D., C. Clark, M. B., Williams, J., and Bensley, J. (2003). Answer mining by combining extraction techniques with abductive reasoning. In *Proceeding of the Twelfth Text Retrieval Conference (TREC 2003)*, pages 375–389.

Hermjakob, U. (2001). Parsing and question classification for question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001) Workshop on Open-Domain Question Answering*.

Hermjakob, U., Echihabi, A., and Marcu, D. (2002). Natural language based reformulation resource and Web exploitation for question answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*.

H.R.Turtle (1991). *Inference Networks for Document Retrieval*. Ph.D. thesis, University of Massachusetts.

Jijkoun, V. and de Rijke, M. (2004). Answer selection in a multistream open domain question answering system.

Jijkoun, V., de Rijke, M., and Mur, J. (2004). Information extraction for question answering: improving recall through syntactic patterns. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1284, Morristown, NJ, USA. Association for Computational Linguistics.

Katz, B., Felshin, S., Yuret, D., Ibrahim, A., Lin, J., Marton, G., McFarland, A., and Temelkuran, B. (2002). Omnibase: Uniform access to heterogeneous data for question answering.

Katz, B., Borchardt, G., and Felshin, S. (2005). Syntactic and semantic decomposition strategies for question answer ing from multiple resources. In *In: Proceedings of AAAI-05, Workshop on Inference for Textual Q uestion Answering*.

Kelly, D. and Lin, J. (2007). Overview of the trec 2006 ciqa task. *SIGIR Forum*, **41**(1), 107–116.

Kim, H., Kim, K., Lee, G. G., and Seo, J. (2001). MAYA: A fast question-answering system based on a predictive answer indexer. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001) Workshop on Open-Domain Question Answering*.

Lee, M., Cimino, J., Zhu, H. R., Sable, C., Shanker, V., Ely, J., and Yu, H. (2006). Beyond information retrieval? medical question answering. *AMIA Annual Symposium Proceedings.*

Lehnert, W. (1978). *The Process of Question Answering: A Computer Simulation Of Cognition*. Lawrence Elbaurn Associates.

Leidner, J., Bos, J., Dalmas, T., Curran, J. R., Clark, S., Bannard, C. J., Steedman, M., and Webber, B. (2003). The qed open-domain answer retrieval system for trec 2003. In *Proceeding of the Twelfth Text Retrieval Conference (TREC 2003)*, pages 595–599.

Li, X. and Roth, D. (2002). Learning question classifiers. In *Proceeding of the 19th International Conference on Computational Linguistics (COLING'02)*.

Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., and Karger, D. R. (2003). What makes a good answer? the role of context in question answering. In *Proceedings of INTERACT 2003*.

Magnini, B., Negri, M., Prevete, R., and Tanev, H. (2002a). Is it the right answer? Exploiting Web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*.

Magnini, B., Negri, M., Prevete, R., and Tanev, H. (2002b). Towards automatic evaluation of Question/Answering systems. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*.

Miller, G. (1995). WordNet: A lexical database for English. *Communications of the ACM*, **38**(11), 49–51.

Mollá, D. and Vicedo, J. L. (2007). Question answering in restricted domains: An overview. *Comput. Linguist.*, **33**(1), 41–61.

Monz, C. (2003a). Document retrieval in the context of question answering. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03)*.

Monz, C. (2003b). Minimal span weighting retrieval for question answering. In *PhD Thesis, University of Amsterdam*.

Morton, T. S. (1999). Using coreference in question answering. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*.

Mur, J. and van der Plas, L. (2007). Anaphora resolution for off-line answer extraction using instances. In *Proceedings of WAR 2007, Workshop for Anaphora Resolution*.

Murdock, V. and Croft, W. B. (2005). A translation model for sentence retrieval. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 684–691, Morristown, NJ, USA. Association for Computational Linguistics.

Ogilvie, P. and Callan, J. (2002). Experiments using the lemur toolkit. In *Proceeding of the 2001 Text Retrieval Conference (TREC 2001)*, pages 103–108.

Phillips, A. (1960). A question-answering routine. Technical report.

Prager, J., Radev, D., Brown, E., Coden, A., and Samn, V. (1999). The use of predictive annotation for question answering in TREC8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*.

Roberts, I. and Gaizauskas, R. (2004). Evaluating passage retrieval approaches for question answering.

Roth, D., Cumby, C. M., Li, X., Morie, P., Nagarajan, R., Rizzolo, N., Small, K., and tau Yih, W. (2002). Question-answering via enhanced understanding of questions. In *TREC*.

Saggion, H., Gaizauskas, R., Hepple, M., Roberts, I., and and, M. A. G. (2004). Exploring the performance of boolean retrieval strategies for open domain question answering. In *Proceedings of the 23rd Annual International ACM SIGIR Workshop on Question Answering (SIGIR 2004)*.

Saquete, E., Martínez-Barco, P., Muñoz, R., and González, J. L. V. (2004). Splitting complex temporal questions for question answering systems. In *ACL*, pages 566–573.

Schank, R. C. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, **3**, 552–631.

Soricut, R. and Brill, E. (2004). Automatic question answering using the web: Beyond the factoid. *Special Issue on Web Information Retrieval*.

Strohman, T., Metzler, D., Turtle, H., and Croft, W. B. (2005). Indri: a language-model based search engine for complex queries. Technical report, in Proceedings of the International Conference on Intelligent Analysis.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. In C. L. Williamson, M. E. Zurko, and P. J. Patel-Schneider, Peter F. Shenoy, editors, *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, Banff, Canada. ACM.

Thorne, J. (1962). Automated language analysis. Technical report.

Tiedemann, J. and Mur, J. (2008). Simple is best: Experiments with different document segmentation st rategies for passage retrieval. In *In: Proceedings of the 2nd IR4QA Workshop, Manchester*.

Vicedo, J. L. and Ferrández, A. (2000). Importance of pronominal anaphora resolution in question answering systems. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*.

Voorhees, E. (1999). The trec-8 question answering track report.

Voorhees, E. M. (2001). Overview of the trec 2001 question answering track. In *TREC*.

Voorhees, E. M. (2004). Overview of the TREC 2004 question answering track. In *TREC*.

Voorhees, E. M. (2005). Overview of the TREC 2005 question answering track. In *TREC*.

White, K. and Sutcliffe, R. F. E. (2004). Seeking an upper bound to sentence level retrieval in question answering. In *Proceedings of the 23rd Annual International ACM SIGIR Workshop on Question Answering (SIGIR 2004)*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, (1), 80–83.

Witten, I. H., Moffat, A., and Bell, T. C. (1999). *Managing Gigabytes; 2nd edition*. Morgan Kaufmann.

Woods, W. A., Kaplan, R. M., and Nash-Webber, B. L. (1972). The lunar sciences natural language information system: Final report. Technical Report 2378, BBN.

Zhang, D. and Lee, W. (2003). Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (SIGIR 2003)*.