ANSWERING *LIST* AND *OTHER* QUESTIONS

Majid Razmara

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

August 2008

Your file Votre référence
ISBN: 978-0-494-45712-2
Our file Notre référence
ISBN: 978-0-494-45712-2

Abstract

Answering *List* and *Other* Questions

Majid Razmara

The importance of Question Answering is growing with the expansion of information and text documents on the web. Techniques in Question Answering have significantly improved during the last decade especially after the introduction of TREC Question Answering track. Most work in this field has been done on answering *Factoid* questions. In this thesis, however, we present and evaluate two approaches to answering *List* and *Other* types of questions which are as important but have not been investigated as much as *Factoid* questions.

Although answering *List* questions is not a new research area, answering them automatically still remains a challenge. The median F-score of systems that participated at the TREC-2007 Question Answering track is still very low (0.085) while 74% of the questions had a median F-score of 0. In this thesis, we propose a novel approach to answering *List* questions. This approach is based on the hypothesis that the answer instances to a *List* question co-occur within sentences of the documents related to the question and the topic. We use a clustering method to group the candidate answers that co-occur more often. To pinpoint the right cluster, we use the target and the question keywords as *spies*. Using this approach, our system placed fourth among 21 teams in the TREC-2007 QA track with F-score 0.145.

Other questions have been introduced in the TREC-QA track to retrieve other interesting facts about a topic. In our thesis, *Other* questions are answered using the notion of *interest marking* terms. To answer this type of questions, our system

extracts, from Wikipedia articles, a list of *interest marking* terms related to the topic and uses them to extract and score sentences from the document collection where the answer should be found. Sentences are then re-ranked using universal interest-markers that are not specific to the topic. The top sentences are then returned as possible answers. To evaluate our approach, we participated in the TREC-2006 and TREC-2007 QA tracks. Using this approach, our system placed third in both years with F-score 0.199 and 0.281 respectively.

Acknowledgments

It would not have been possible to write this master thesis without the help and support of the kind people around me. First and foremost, I would like to take this opportunity to express my deepest gratitude toward my wonderful supervisor, Dr. Leila Kosseim, for her supervision, advice and guidance from the early stage of this research. I am also indebted to her for all the support, encouragement and invaluable comments she provided to me.

I would like to thank my beloved wife, Maryam, for her personal support and great patience at all times. I would not have finished my thesis without her backing. Many thanks also to my parents who have given me their unequivocal support throughout.

I would also like to thank my colleagues and friends at the CLaC lab, especially Osama El Demerdash with his invaluable helps for which I am extremely grateful.

Last but not least, my deepest thanks to Dr. Rene Witte, Dr. Joey Paquet and Dr. Todd Eavis for their constructive comments on my thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The amount of information provided on the Internet is increasing enormously every day. Thus, finding desired information in this huge knowledge repository has become an important research area in Computer Science. Information Retrieval (IR) techniques are able to analyze a query and return a ranked list of potentially relevant documents. Although IR systems and techniques have recently advanced significantly, the problem is not fully answered: IR systems do not directly locate the appropriate information; instead, they return a ranked list of potentially relevant documents. Therefore, the user has to go through the documents one by one and scan them until he finds the pertinent information. Another issue is forming the query; not everyone using IR systems are experienced users who know how to form a query that best represents the information they are seeking for. Question Answering (QA) takes the next step beyond Information Retrieval to tackle these two problems. In contrast to IR, QA systems are given questions in natural language and they attempt to respond to the questions by providing the exact answers.

Question Answering is an interesting area for NLP researchers because it provides a unique challenge where Information Retrieval and Information Extraction (i.e. automatically extracting structured information) techniques are used together. [Lin, 2007] showed the superiority of QA systems over traditional IR engines using a novel evaluation framework.

Although the first QA systems appeared during 1960s, Question Answering drew researchers' attention only after 1999 when TREC[1] introduced the first Question Answering (QA) track in TREC-8. It was the first large-scale evaluation of open domain question answering systems. In the first QA track, systems were asked to return a small snippet of text that contains the answer to a question while in the recent QA tracks, systems are to return the exact answer without any extra words (which otherwise, it is judged inexact). *Factoid* questions (i.e. questions with fact-based, short answers) were the only type of questions that were given to the participants in TREC-8 [Voorhees and Tice, 1999]. Two other types of questions, *List* and *Other*, were added to the QA track in later years.

1.2 TREC Question Answering

In the recent TREC QA competitions, the main task contains three different types of questions: namely *Factoid*, *List* and *Other*. The questions are organized into series of questions, each associated with a topic called the "*target*" which specifies the context of the questions in the series. Each question series consists of several *Factoid* questions, one or two *List* questions and ends with exactly one *Other* question. For example, question series # 69 of TREC-2005 is:

[1] Text REtrieval Conference

69		Target: France wins World Cup in soccer
69.1	FACTOID	When did France win the World Cup?
69.2	FACTOID	Who did France beat for the World Cup?
69.3	FACTOID	What was the final score?
69.4	FACTOID	What was the nickname for the French team?
69.5	FACTOID	At what stadium was the game played?
69.6	FACTOID	Who was the coach of the French team?
69.7	LIST	Name players on the French team.
69.8	OTHER	Other

The questions are developed as if they were asked by an adult, native speaker of English who is an average reader of U.S. newspapers [Dang *et al.*, 2007]. The target could be a PERSON, ORGANIZATION, THING, or EVENT.

Answers to TREC-2007 questions are to be drawn from a document collection comprising the BLOG06[2] corpus [Macdonald and Ounis, 2006] and the AQUAINT-2 Corpus of English News Text. Blog06 consists of a crawl of 100K RSS and Atom feeds (two main web feed formats). The AQUAINT-2 collection comprises approximately 2.5 GB of text (about 907K documents). Articles come from a several sources including Agence France Presse, Central News Agency (Taiwan), Xinhua News Agency, Los Angeles Times-Washington Post News Service, New York Times, and The Associated Press [Dang *et al.*, 2007].

Answers to TREC-2006 and prior, however, are to be extracted from the AQUAINT document collection. AQUAINT consists of about 1 million documents taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires [Dang *et al.*, 2006].

Each question type has its own evaluation measure. *Factoid* questions are evaluated based on accuracy, while the evaluation metric for the list and other components

[2] `http://ir.dcs.gla.ac.uk/test_collections/blog06info.html`

is the F-score. Hence, an aggregate score is computed for each series using the average of the component scores of questions in that series, and the final score is computed as the average of all per-series scores [Dang *et al.*, 2007].

Most focus in Question Answering evaluation forums such as TREC[3], CLEF[4] (Cross-Language Evaluation Forum), and NTCIR[5] (National institute of informatics Test Collection for Information Retrieval) has been on answering *Factoid* questions. In this thesis, however, we present and evaluate two approaches to answering *List* and *Other* types of questions, which are as important as *Factoid* questions but are not investigated as much. Recently, INEX[6] (INitiative for the Evaluation of XML retrieval) has introduced the *Entity Ranking* track to foster research on *List* questions.

1.3 List Questions

List questions require a list of correct answer instances to be extracted from multiple documents. This type of question was initiated in 2001 at the TREC Question Answering track [Voorhees, 2001].

Answering *List* questions is more difficult compared to answering *Factoid* questions because it requires a system to acquire the answer instances from different sources (answer fusion). In addition, the number of expected answer instances is not known beforehand. Indeed, the factoid type can be seen as a simplified version of the list type, in which the number of expected answers is provided and has value one.

For example, one of the *List* questions at the TREC QA-2007 is *"Which airlines use Dulles?"*, for which the target is *"Dulles Airport"*. Twenty airlines are found in the corpora to be instances of the answer to this question. The answers to this question were extracted from 37 different documents in the corpora by the participants of the

[3] `http://trec.nist.gov`
[4] `http://www.clef-campaign.org`
[5] `http://research.nii.ac.jp/ntcir/workshop`
[6] `http://inex.is.informatik.uni-duisburg.de/`

TREC-2007.

To evaluate the answers, TREC assessors, volunteer humans, assign exactly one of the following five judgments to each response [Dang *et al.*, 2007].

1. **Incorrect:** the answer string does not contain a correct answer.

2. **Not Supported:** the answer string contains a correct answer but the document returned does not actually answer the question.

3. **Not Exact:** the answer string contains a correct answer and the document supports that answer, but the string contains more than just the answer or is missing bits of the answer.

4. **Locally Correct:** the answer string consists of exactly a correct answer that is supported by the document returned, but the document collection contains a contradictory answer that the assessor believes is better.

5. **Globally Correct:** the answer string consists of exactly the correct answer, that answer is supported by the document returned, and the document collection does not contain a contradictory answer that the assessor believes is better.

Only globally correct answers are considered correct in the evaluation. After judging the answers to a *List* question, the response to that question is evaluated using F-score (also known as F-measure), precision, and recall (for the definitions, refer to Section 5.1).

Different instances of the answer to a *List* question have a special relation to one another. Beside the fact that they are all of the same entity class (e.g. country, people, book, position, ...), they either co-occur within sentences, or occur in different sentences having lexical similarities or occur in different sentences that are partially or totally semantically equivalent. The latter case requires a system to

Target 232: "Dulles Airport" *Question 232.6: "Which airlines use Dulles?"*

Source	Sentence Containing Answers
LTW-ENG 20050712.0032	**Delta**, Northwest, **American**, **British Airways** and **KLM** share four screening machines in the basement.
	United, which operates a hub at Dulles, has six luggage screening machines in its basement and several upstairs in the ticket counter area.
TTW-ENG 20060102.0106	**Flyi** suffered from rising jet fuel costs and the aggressive response of competitors, led by **United** and **US Airways**.
	Independence said its last flight Thursday will leave White Plains, N.Y., bound for Dulles Airport.
NYT-ENG 20050426.0151	**Delta** recently added more than 80 flights from its Atlanta hub and capped its business fares, while money-losing **Independence Air**, based at Washington's Dulles Airport, is making ....
WIKIPEDIA (web)	At its peak of 600 flights daily, **Independence**, combined with service from **JetBlue** and AirTran, briefly made Dulles the largest low-cost hub in the United States.
NEW YORK TIMES (web)	**Continental Airlines** sued **United Airlines** and the committee that oversees operations at Washington Dulles International Airport yesterday, ...

Figure 1: Answers tend to co-occur with one another and with the target and question keywords (Acceptable answers are shown in bold face and the target and question keywords are underlined)

deal with the difficulties of semantic analysis, which is not our concern. Using these features, one can extract or expand the potential candidate answers. For example, [Ahn *et al.*, 2005b] use lexical similarities among sentences to expand the initial candidate lists. It actually exploits the idea that sentences containing answer instances share similar words.

In this thesis, we hypothesized that the instances of the answer to a *List* question co-occur within the sentences of the documents related to the target and the question.

In addition, these instances also tend to co-occur with the target and the question keywords.

Figure 1 shows a few sample sentences from AQUAINT-2, one of the TREC corpora, and from the Web related to question 232.6 *"Which airlines use Dulles?"* (Target: *"Dulles Airport"*). As the figure shows, the sentences contain instances of answers along with the target and question keywords.

Our approach is able to select the most likely candidate answers among initial answers, while exploiting only statistical techniques. In this approach, after creating a list of initial candidate answers from a small number of documents relevant to the target and the question, the information regarding the co-occurrence of the candidates in a larger number of related documents are extracted. A method attempts to group candidates that co-occurred more often. Finally, the candidate in the most likely cluster to have the answers is returned as the final candidates. Using this approach, our system placed fourth among 21 teams in the TREC-2007 QA track with F-score (see Section 5.1) of 0.145.

1.4 Other Questions

Since 2004, the TREC Question Answering track has introduced a new type of challenge: answering *Other* questions [Voorhees, 2004].

The answer to an *Other* question is meant to be interesting information about the target that is not covered by the preceding questions in the series, and should consist of a snippet of text, called a *nugget*, extracted from the corpora. This type of question is to be interpreted as "Tell me other interesting things about this target; I don't know enough to ask directly" [Dang *et al.*, 2007].

To evaluate the answers to an *Other* question, NIST assessors create a list of acceptable information nuggets about the target. Some of the nuggets are deemed

vital, some are *okay* and others are *uninteresting* (or unrelated at all).

Systems are then evaluated based on precision and recall of the nuggets, and ultimately the F-score with $\beta = 3$ [7] (for the definitions, refer to Section 6.1). *Vital* and *okay* nuggets are evaluated differently: the number of *vital* nuggets are used to compute both recall and precision; while *okay* nuggets are used for precision only.

For example, target 201 of TREC-2006 requires interesting information (not asked before) about "William Shakespeare." Table 1 shows the vital and okay prototypical snippets for this question.

Judgement	Prototypical Snippet
Okay	From 1611 to 1616 lived in Stratford and wrote nothing
Vital	Chosen Britist Person of the millennium by BBC listeners
Okay	"Shakespeare in Love" based on the life of William Shakespeare
Okay	Shakespeare usually spelled his last name Shakspere
Okay	Anthony Holden wrote biography of Shakespeare called William Shakespeare: the man behind the genius
Okay	Britain's Globe Theater will stage the largest exhibit dedicated to Shakespeare

Table 1: Vital and Okay Nuggets for Target 201 of TREC-2006

Any system response that conveys the meaning of any of the prototypical snippets is judged the same as that prototypical snippet.

In TREC-2007, a *Nugget Pyramids* method [Lin and Demner-Fushman, 2006] was used for evaluating *Other* questions, in which multiple assessors provide judgments on whether a nugget is vital or okay. Then, a weight is assigned to each nugget based on the number of assessors who marked it as vital [Dang *et al.*, 2006].

[7] which means that recall is three times as important as precision

Answering *Other* questions is a difficult task because we do not really know what we are looking for. There is no exact definition of what constitutes a *vital* and an *okay* answer and humans themselves may have different opinions about how interesting a nugget is. In fact, at TREC-2005, the University of Maryland submitted a manual run for the *Other* questions where a human had identified manually what he considered to be interesting nuggets for each questions [Voorhees and Dang, 2005]. This manual run was then submitted for judging along with automatic runs and received an $F(\beta = 3)$ score of 0.299. This low score seems to show that humans do not agree easily on what constitutes an interesting (*vital* or *okay*) piece of information.

To answer an *Other* question, our system extracts from Wikipedia articles a list of *interest-marking* terms related to the topic and uses them to extract and score sentences from the corpora where the answers should be found. Sentences are then re-ranked using universal interest-markers that are not specific to any topic. The top sentences are then returned as possible answers.

We participated in the TREC-2006 and TREC-2007 QA tracks. Using this approach, our system placed third in both years with F-score 0.199 and 0.281.

1.5 Outline of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 outlines previous research on QA systems, with an emphasis on the *List* and *Other* types of questions. In Chapter 3, we discuss our approach to answering *List* questions. Answering *Other* questions is explained in details in Chapter 4. Chapter 5, then, presents the evaluation and the official results of our system in answering *List* questions at the TREC-2007 QA track. Chapter 6 includes the evaluation metric and the official results of our approach to answering *Other* questions at TREC-2006 and 2007. Finally, Chapter 7 presents a summary of our approaches and future directions.

Chapter 2

Literature Review

In this chapter, we will review the previous work done on answering *List* questions (Section 2.1) and then the related work on answering *Other* questions (Section 2.2).

2.1 Related Work on List Questions

This section presents some of the background work in answering *List* questions. The first part gives some basic background on the different approaches exploited by QA researchers. The second and third part cover prior and related work that use co-occurrence information and clustering.

2.1.1 Approaches to Answering List Questions

Several approaches have been used to answer *List* questions. Some systems, for example [Gaizauskas *et al.*, 2005], [Whittaker *et al.*, 2006] and [Zhou *et al.*, 2006], treat *List* questions as an expanded version of *Factoid* questions, which requires a list of distinct answers rather than only one answer. These systems answer a *List* question by simply returning the top N candidate answers found by the *Factoid* question answering system.

However, a variety of approaches have been developed by many automatic QA systems to answer *List* questions specifically. These techniques include logic inference, syntactic relation analysis, information extraction, using a pre-compiled knowledge base and external online knowledge resource [Wu and Strzalkowski, 2006].

The LCC system [Moldovan *et al.*, 2002, Moldovan *et al.*, 2004] combines several NLP tools such as Named Entity Recognition, syntactic parsing, ontologies, semantic relation extraction, advanced inference, coreference resolution, temporal contexts, and lexical chains. The system makes use of a natural language logic prover (COGEX) to re-rank the candidate answers based on the semantic entailment between the passage containing the candidate answer and the question. LCC's system was the best system from TREC-2000 to TREC-2007 and has been developed over the past 10 years.

Answering questions using syntactic relations has been investigated by some researchers, e.g. [Katz and Lin, 2003], [Cui *et al.*, 2004] and [Shen and Klakow, 2006]. Using a parser, syntactic relations from both the candidate answers and the question are extracted and matched against each other to ensure that the proper relations existed in both.

Pattern matching using regular expressions is an efficient solution for some questions, although it is relatively simple. Patterns have been used in quite a few systems, for example [Ravichandran and Hovy, 2001], [Soubbotin, 2001], [Jijkoun *et al.*, 2004] and [Hao *et al.*, 2006]. Patterns can be developed at different levels: lexical, syntactic or semantic; or a combination of them.

Alternatively, several statistical approaches have been proposed, for example [Brill *et al.*, 2002], [Echihabi and Marcu, 2003], [Ravichandran *et al.*, 2003], [Soricut and Brill, 2004], and [Whittaker *et al.*, 2005].

[Whittaker *et al.*, 2005] take a statistical, noisy-channel approach and consider question answering as a classification problem. A mathematical model for answer retrieval, answer classification and answer length prediction is derived. Word tokens

and web data are used extensively but no explicit linguistic knowledge is incorporated. The authors claim that this model can be used for many languages without the need for highly tuned linguistic modules.

Redundancy-based approaches, which are mostly exploited for *Factoid* questions, are used for *List* questions as well. They exploit the idea that the correct answer should appear in many places in large corpora, especially on the Web. In a redundancy-based system, for example [Clarke *et al.*, 2001], [Brill *et al.*, 2001] and [Kosseim *et al.*, 2006], the weight of each candidate answer is computed as a function of its frequency within the relevant documents.

On the other hand, in a number of systems, the relationship between question and answer candidates is taken into account. In our work and in a few others, however, the relationship between answer candidates and/or the relationships between answer candidates and question keywords are exploited.

The relationship between candidate answers and/or question keywords can be defined in several ways. In our work, we use co-occurrence information. However, other methods, e.g. using thesauri, WordNet[1] semantic similarity, conceptual similarity via ontologies, etc., can be exploited as well. For example, [Ahn *et al.*, 2005b] and [Kor, 2005] propose an approach that identifies the common context shared by two or more candidate answers and use this common context to identify more candidates that were previously not found by the original question answering system. The words that frequently co-occur with two ore more answer candidates are used as the common context.

[Dalmas and Webber, 2007] experimented with two approaches: 1) Considering candidates as competitors and scoring them as individuals; 2) Considering them as potential allies and including additional scoring based on their mutual relationships.

[1] A semantic lexicon for the English language `http://wordnet.princeton.edu/`

They use the Model-View-Controller[2] design pattern and generate an answer model from a question and a list of answer candidates. [Dalmas and Webber, 2007] showed that the second approach is more promising and claimed that such relations holds not only between correct answers but also between correct candidates and incorrect ones that relate to an answer topic. These incorrect, but related, candidates help by linking correct answers to each other and also by providing background information that can be used to explain answer multiplicity.

2.1.2 Utilizing Co-occurrence Information

Co-occurrence is an indicator of semantic similarity in *Statistical Semantics*. Generally, terms that co-occur more frequently tend to be related. Co-occurrence information has been used in several applications. In particular, [Cui *et al.*, 2005] use the co-occurrence of question keywords to extract and rank relevant passages to the question.

[Matsuo and Ishizuka, 2004] propose a keyword extraction algorithm that uses word co-occurrence statistical information within a single document. The algorithm extracts frequent terms in the document and counts the co-occurrences of each term in the document with these extracted frequent terms. Terms that appear frequently with a subset of terms are likely to have important meaning. The χ^2 test is used to measure the degree of bias of the co-occurrence distribution. The results of this algorithm is comparable to the *tf-idf* [3] measure, which is a popular, high performance measure for keyword extraction. The *tf-idf* measure extracts terms which are frequent in a document, but do not appear frequently in the rest of the corpus. Therefore using co-occurrence information, [Matsuo and Ishizuka, 2004] extract keywords in a document without requiring the use of a corpus.

[2] An architectural pattern used in Software Engineering to isolate business logic from user interface.

[3] Term frequency-inverse document frequency, a term-weighting method in information retrieval

[Li, 2002] uses co-occurrence data to address the problem of clustering words (or constructing a thesaurus). His work, then, makes use of the acquired word classes in resolving prepositional-phrase-attachment disambiguation. This approach represents co-occurrence patterns over word pairs using a probability model and uses an estimation algorithm based on the *Minimum Description Length*[4] (*MDL*) principle.

[Turney, 2001] in his widely cited paper shows how a simple algorithm based on statistical data acquired by querying a Web search engine can be used to identify synonyms of a word. The algorithm, called *PMI-IR*, uses *Pointwise Mutual Information* (*PMI*) and *Information Retrieval* (*IR*) to measure the similarity of pairs of words. The algorithm is, then, evaluated using *TOEFL*[5] tests. The algorithm scores almost 10% higher than *Latent Semantic Analysis* (*LSA*), a well known algorithm whose performance on the *TOEFL* tests has been often cited as evidence for the value of *LSA*.

[Wu and Strzalkowski, 2006] use co-occurrence information to answer a series of *Factoid* questions on a specific topic in a batch mode. The authors hypothesize that the answers to many *Factoid* questions in a series have a high degree of co-occurrence within relevant document passages. The approach uses the interdependence (co-occurrence) between the answers to different questions of a same topic to help to filter out the noisy information and pinpoint the correct answer for each question in the series. For this purpose, [Wu and Strzalkowski, 2006] cluster questions of a series based on a number of attributes. Answer co-occurrence maximization tries to retrieve a set of individual candidate answers to each question which has the maximum *Pointwise Mutual Information*.

To validate and identify relevant answers from a list of extracted candidates,

[4] An important concept in information theory which states that the best hypothesis for a given set of data is the one that leads to the largest compression of the data.

[5] Test of English as a Foreign Language

several approaches have been used. Some systems use WordNet, gazetteers and ontologies for this purpose (e.g. [Moldovan *et al.*, 2003] and [Xu *et al.*, 2002]) while [Ko *et al.*, 2007] uses a probabilistic graphical model. As we will discuss in Section 3.5.1, our approach attempts to cluster relevant candidates based on co-occurrence data and then chooses the cluster which is the most likely cluster to contain the answers.

2.1.3 Using Clustering in Answering List Questions

Clustering has also been used in the context of answering *List* questions. For example, [Yang and Chua, 2004] use Web pages to extract as many answers as possible. Using a list of features[6], their approach then classifies the Web pages into four categories: *Collection Pages*, those that contain a list of answers, *Topic Pages*, pages that are about an answer, *Relevant Pages*, which are relevant to a single answer instance and *Irrelevant Pages* which are not relevant to the topic. Having extracted the initial candidate answers from the collection pages, their approach clusters relevant pages into related groups using the topic pages as cluster seeds and extracts additional candidates from each cluster.

[Jijkoun *et al.*, 2006] use a divisive algorithm to cluster candidate answers at CLEF-QA 2006 [Magnini *et al.*, 2006]. A graph-based algorithm is used in order to capture the non-transitive nature of answer similarity. The similarity score is an inverse exponential function of the edit distance between the strings, normalized by the sum of the string lengths. The algorithm starts by putting all the answers in one cluster, then it recursively splits clusters according to spectral analysis of the similarity matrix. Splitting stops when the similarity degree exceeds a certain threshold.

[6] Based on *Known* and *Unknown elements*, *Answer Target elements*, *Hyperlinks*, *URL*, *HTML structure*, *Anchor*, *List* and *Named Entities*

[Li, 2002] employs co-occurrence data to address the thesaurus construction problem and uses a Hierarchical Agglomerative algorithm[7] to cluster the words. For each pair of clusters, the approach calculates the reduction of mutual information which would result from merging them. Iteratively, a pair with the least reduction in mutual information is chosen to be merged. The iterations stop when the least mutual information reduction found is greater than a threshold.

Some attempts have been made to reduce the difficulty of answering *List* questions by defining some subproblems and focusing on them. For instance, *Entity Ranking* track was proposed by INEX 2007 [8]. This track contains an *Entity Ranking* task, to return entities that satisfy a topic in natural language and also a *List Completion* task, to complete a partial list of answers, given a topic text and a number of examples. The *List Completion* task is inspired by *Google Sets*[9]. [Ghahramani and Heller, 2005] and [Adafre *et al.*, 2007] use several approaches based on these tasks to expand and validate the candidate list.

As we have seen in the previous paragraphs, many approaches are proposed to answer *List* questions. However, to our knowledge, none have used the idea of clustering candidate answers based on co-occurrence information. In Chapter 3, we will explain how we approached this problem as a clustering problem.

2.2 Related Work on Other Questions

Previous approaches to answering *Other* questions have mainly been addressed within the TREC confines, and only since 2004 [Voorhees, 2004][Voorhees and Dang, 2005]. The most widely used approaches are based on patterns, keywords and question generation techniques.

[7] A bottom up algorithm that builds a hierarchy of clusters

[8] "INitiative for the Evaluation of XML retrieval" `http://inex.is.informatik.uni-duisburg.de/2007/`

[9] `http://labs.google.com/sets`

In the pattern-based approach, a set of predefined patterns that seem to present interesting information are extracted from the answers of the previous years' TREC *Other* questions. Then the target is applied to the patterns to generate a potentially interesting string that is searched in the document collection.

[Harabagiu *et al.*, 2005] use a variety of strategies including the use of definition-patterns. For example, the pattern "*TARGET, which ...* " is used to identify nuggets that define the target, and hence is deemed to contain interesting information.

[Wu *et al.*, 2005b] also use patterns for extracting useful information and some semantic features to score sentences. These semantic features include comparative adjectives, digits, topic related verbs and topic phrases. [Ferres *et al.*, 2005] use patterns and a summarizer based on lexical chains to extract a sentence as a summary of a passage.

On the other hand, keywords are also used to find the answers to *Other* questions. [Ahn *et al.*, 2005a], for example, use syntactic information to identify interesting nuggets in the AQUAINT collection. They identify sentences where the target appears in the subject or object position, then use a list of interest-marking keywords (similar to our approach) to rank these sentences. [Ahn *et al.*, 2005a] also use the Wikipedia online encyclopedia to re-rank the sentences. However, they do not analyze the article per se to find interesting terms, but rather the corresponding XML file to look for the meta-data on the target and identify the categories the article belongs to. These categories are then used as keywords to re-rank the nuggets. As opposed to their work, we further re-rank the nuggets using interest markers.

[Chen *et al.*, 2005] identify sentences that contain more than 50% of the words in the targets as candidate sentences. In ranking the sentences, those having more overlap with the target are given higher scores.

Finally, [Roussinov *et al.*, 2005] use statistics about word triplet co-occurrences from the documents related to each target then, extract snippets corresponding to

the most frequent word triplets.

The third main approach used in answering *Other* questions can be qualified as question generation. This approach attempts to answer *Other* questions using *Factoid* or *List* question answering approaches. [Harabagiu *et al.*, 2005], for example, first classify the targets according to their type, then create a list of potential questions for each type of target. For example, if the target is of the type *musician-person*, a set of questions such as "*What is the name of the band of TARGET*" or "*What kind of singer is TARGET*" are generated. Using their factoid module, the answers to these typically interesting questions are extracted.

Some question answering systems use both pattern-based and keyword-based approaches. In [Wu *et al.*, 2005a], a web knowledge acquisition module determines which kind of knowledge base should be searched based on the target type. Then, the basic score of a candidate sentence is assigned either by searching the definitions about the target from online knowledge bases or by keywords and their frequencies. Finally, based on the target type, a set of structured patterns is used to re-rank the candidate sentences. [Gaizauskas *et al.*, 2005] use a list of terms related to each target extracted from the Web pages, Wikipedia and Britannica pages. Then two types of patterns were used: lexical patterns (e.g. "*X which is*", "*like X*") and part-of-speech and named entity patterns (e.g. "*TARGET, WD VBD*").

Other less popular approaches have also been proposed. In [Katz *et al.*, 2005], for example, three strategies are exploited: a nugget can be extracted either by searching a database of definitional contexts, searching the corpus for a nugget including many keywords from the Websters Dictionary definition, or extracting all sentences from the top documents and using Wikipedia synonyms of the target. [Harabagiu *et al.*, 2005] also try to locate specific named entities in the nuggets corresponding to the target types. For example, if the target is a person, then nuggets containing dates, quantities and locations are deemed more interesting.

As Section 2.2 shows, several approaches have been used to answer *Other* questions. Our approach uses the notion of *interest marking* terms. In Chapter 4, we will explain our approach in more details.

Chapter 3

Answering List Questions

This chapter describes our approach to answering *List* questions. Our approach is based on the *Distributional Hypothesis*, which states that words occurring in the same contexts tend to have similar meanings [Harris, 1954]. *Distributional Hypothesis* is the basis of *Statistical Semantics* defined as the study of "how the statistical patterns of human word usage can be used to figure out what people mean, at least to a level sufficient for information access" [Furnas, 2008]. Following this view, we hypothesized that:

1. The instances of the answer to a *List* question have the same semantic entity class;

2. The instances of the answer tend to co-occur within the sentences of the documents related to the target and the question;

3. The sentences containing the answers share similar context.

We use the target and the question keywords as the representatives of context. In other words, we hypothesize that the instances of an answer to a *List* question tend to co-occur together and also tend to co-occur with the target and the question keywords within the sentences of the relevant documents. Co-occurrence can be an

indicator of semantic similarities; generally, terms that co-occur more frequently tend to be related.

A *List* question is answered in the following steps: First, the answer type of the question is determined. Then, two different queries for searching the Web and the corpora (AQUAINT-2 and BLOG06) are generated using the target and the question text. We use these two queries to create a collection of documents in which we look for the answers to the question. We extract from the collection all terms that comply with the answer type; this constitutes the initial candidate list. A similarity value is then computed for each pair of candidate answers based on their co-occurrence within sentences. Having clustered the candidates and determined the most likely cluster, the final candidate answers are selected.

Figure 2 shows the architecture of the system using this approach. In the figure, 1 and 2 refer to the possible methods we have in candidate selection. In the following sections, we will discuss these processes in greater detail.

3.1 Answer Type Recognition

The first step to answering a question is question analysis and answer type recognition. This information is crucial for the next module, *Candidate Answers Extraction*, which uses the answer type to extract all possible terms with that answer type. Each question is associated to one of the nine semantic entity classes: PERSON, COUNTRY, ORGANIZATION, JOB, MOVIE, NATIONALITY, CITY, STATE, and OTHER. The class OTHER comprises all answer types which do not fall into any of the other categories and hence a subtype is defined for it. The answer type recognition module consists of three tiers:

1. Lexical Patterns

2. Syntactic Patterns

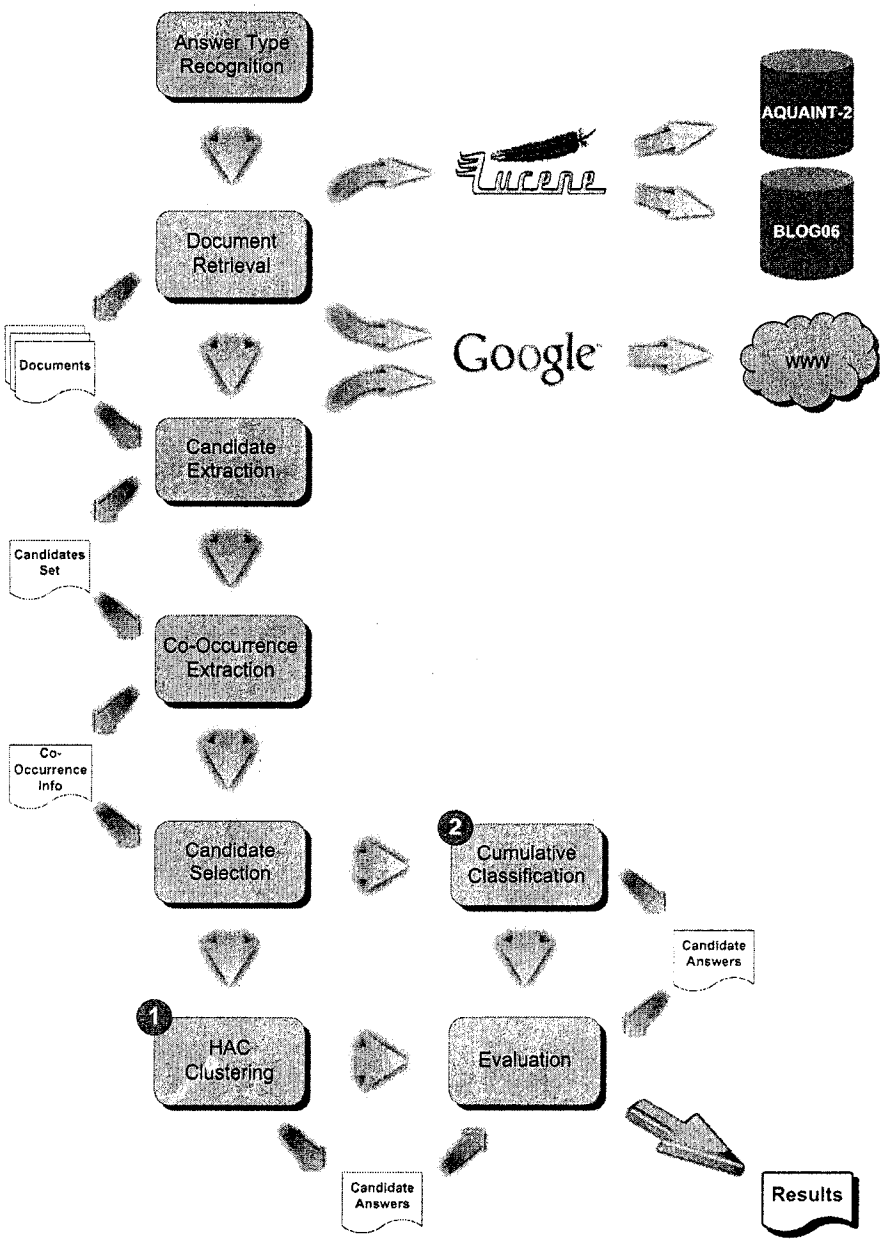

Figure 2: List Answering System Architecture

3. Type Resolving

Given a question, each tier attempts to predict the answer type using a different method. If it fails, the question is passed to the next tier.

3.1.1 Lexical Patterns

The type of the answer to a *List* question is first tried to be predicted using lexical patterns. A set of lexical patterns have been developed by analyzing the *List* questions of the TREC QA Track in 2005 and 2006. Table 2 shows some examples of lexical patterns. Patterns are written in regular expressions, which provide an efficient and easy way of matching. In Table 2, carets (^) indicate the beginning of an expression (a question, here) and vertical bars (|) separate alternatives.

LEXICAL PATTERN	TYPE
INIT = (Name | List | What | Which)	
^ (Who | Whom)	PERSON
^ *INIT* (persons | people | men | women | players | contestants | artists | graduates | opponents | students)	PERSON
^ *INIT* (countries | nations)	COUNTRY
^ *INIT* (organizations | companies | sponsors | manufacturers | corporations | institutions)	ORGAN
^ *INIT* (movies | films | tv shows | film titles | titles of movies)	MOVIE
^ *INIT* (cities | towns)	CITY
^ *INIT* (occupations | positions | posts)	JOB
(Who | Whom)	PERSON
INIT (persons | people | men | women | players | contestants | artists | graduates | opponents | students)	PERSON
INIT (countries | nations)	COUNTRY
INIT (cities | towns)	CITY

Table 2: Extracts of Lexical Patterns

In Table 2, *INIT* refers to (*Name*|*List*|*What*|*Which*); so it can match any of these four words. For example, ^*INIT*(*cities*|*towns*) matches a question that starts with one of the mentioned words (i.e. *name*, ...), either *cities* or *towns*, and possibly followed by other words. However, this pattern fails to match a question that does not start with *INIT*, e.g. "In what cities ...". To deal with this kind of questions, the twin of each rule is added without a caret (^). These rules, however, have a lower priority than their counterparts and are placed at the end of the lexical patterns list.

Table 2 contains extracts of the lexical patterns used in the *Answer Type Recognition* module. However, the patterns are not meant to be exhaustive. For example, the second pattern contains a list of hyponyms of PERSON extracted from the questions of the training set (TREC-2004 to 2006). Answer types of questions with other hyponyms of person (e.g. teacher, professor, actor, doctor, ...) will be recognized using the next two modules (see Sections 3.1.2 and 3.1.3).

These extracted lexical patterns are specific to *List* questions and can not be applied as is to other types of questions. For example, using the current lexical patterns, the answer type of "Who is John Wayne?" will be recognized as PERSON, which obviously is not the right answer type. Answer type recognition of non-list questions goes beyond the scope of our work.

If any of the lexical patterns match part of a question, the type of the pattern will be assigned to the question as its answer type. Table 3 lists four questions, taken from the test set of TREC-2006 QA track, whose types are identified using lexical patterns.

If none of the lexical patterns matches the question, the type of the answer is assumed to be OTHER, and the process passes to the next tier.

Q #	Question Text	Type
157.6	Who has served as Secretary-General of the U.N.?	PERSON
159.3	List companies that have filed suits against Wal-Mart.	ORGAN
170.5	What artists has John Prine done duets with?	PERSON
178.6	In what cities or towns have illegal methamphetamine labs been found?	CITY

Table 3: Examples of Questions Whose Answer Types are Recognized using Lexical Patterns

3.1.2 Syntactic Patterns

Lexical patterns are meant to cover regular cases; they are rather accurate, but do not cover all cases (i.e. high precision, low recall). For example, although "Name people ..." is recognized by the lexical patterns, they fail to match questions with a slight variation such as "Name all people ...". For handling these cases as well as questions with the OTHER type, we have developed *Syntactic Patterns.*

Assuming that the answer type is OTHER, this tier attempts to find out the answer subtype which is usually the head of a noun phrase after the Wh-determiner (What, Which, ...) or Verb (List, Name, ...) in the question. These patterns are sequences of part-of-speech tags written in regular expressions. For example, in the question 259.7 of TREC-2007 ,"Name the World Snooker tournaments", the answer type is OTHER and the subtype is *Tournament.*

Table 4 shows three syntactic patterns which are used to identify answer subtypes. The patterns are developed based on the Penn Treebank[1] part-of-speech tag set and regular expressions. Here, □ refers to word delimiters, '+' denotes one or more occurrences and '*' means zero or more occurrences of its previous clause.

[1] `http://www.cis.upenn.edu/~treebank/`

Syntactic Pattern
^(WDT|WP|VB|NN)□(DT|JJ)*□(NNS|NNP|NN|JJ|□)* □(NNS|NNP|NN|NNPS)□ (VBN|VBD|VBZ|WP|$)
^(WDT|WP|VB|NN)□(VBD|VBP)□ (DT|JJ|JJR|PRP$|IN)*□(NNS|NNP|NN|□)*□ (NNS|NNP|NN)
(WDT|WP|VB|NN)□(DT|JJ|JJR|PRP$\|VBG\|CC\|NN)*□(NNS\|NNP\|NN\|NNPS)*□ (NNS\|NNP\|NN\|NNPS)□(VBN\|VBD\|VBZ\|VBP\|WP\|WP$|WDT|$)

Table 4: Extracts of Syntactic Patterns based on the Penn Treebank

For example, (WDT|WP|VB|NN) should be interpreted as either of Wh-determiner (WDT), Wh-pronoun (WP), base form of verb (VB) or singular noun (NN).

The question is tagged using Stanford's POS tagger [Toutanova *et al.*, 2003] and its sequence of tags are compared to the syntactic patterns. If the matching succeeds, the term containing the tag that is underlined is assigned to the question subtype.

Table 5 shows examples of questions, taken from the training set of TREC-2006 QA track, whose subtypes are identified using syntactic patterns.

Q #	Question Text	Type	Sub Type
144.6	In what conflicts has the division participated?	OTHER	Conflict
151.2	What races are part of the Winston Cup series?	OTHER	Race
158.6	Name the schools of Tufts University.	OTHER	School
212.7	List the songs he recorded.	OTHER	Song

Table 5: Examples of Questions Whose Answer Subtypes are Recognized using Syntactic Patterns

3.1.3 Subtype Resolving

Sometimes the subtype extracted using syntactic patterns belongs to one of the main answer types. For example, for the question "List previous conductors of the Boston Pops"(TREC-2007, question 236.2), the answer type recognizer, after using lexical and syntactic patterns, determines OTHER for the answer type and *conductor* as its subtype, while the question, in fact, belongs to the PERSON type. This kind of problem is unavoidable since lexical patterns cannot contain an exhaustive list of all words that could refer to a person or any other type.

We take only three types (PERSON, ORGANIZATION, and JOB) into account for subtype resolving because the other types do not face this problem; for example, there are not many common words, as synonyms or hyponyms, which refer to COUNTRY and we can mention them all in the lexical patterns.

In this step, we try to resolve the subtype to one of the answer type categories, if possible. For this purpose, we exploit WordNet's hypernym hierarchy feature. The hypernym chain of the answer subtype is examined to see whether it contains any of the main type categories. Figure 3 illustrates the inherited hypernyms of *conductor* extracted from WordNet[2].

Because a word can have several senses, there could be more than one hypernym chain. To avoid introducing noise, we only consider the most frequent sense for hypernym checking. In WordNet, the different senses of a word are ordered from most to least frequently used [Fellbaum, 1998], so the hypernyms of the first sense is expanded and checked as to whether it contains any of the main types.

If the subtype identified is "name", "title", "type", etc. we need to do another round of type recognition because these types are too general. For example, in the question 167.6 of TREC-2006, "List the names of other millennium structures in England.", the identified subtype is "name" rather than the actual subtype which

[2] `http://wordnet.princeton.edu/perl/webwn3.0?s=conductor`

Noun

- S: (n) **conductor**, music director, director
  - *direct hyponym* / *full hyponym*
  - *has instance*
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) musician
      - S: (n) artist, creative person
        - S: (n) creator
          - S: (n) person, individual, someone, somebody, mortal, soul
            - S: (n) organism, being
              - S: (n) living thing, animate thing
                - S: (n) whole, unit
                - S: (n) object, physical object
                - S: (n) physical entity
                - S: (n) entity
            - S: (n) causal agent, cause, causal agency
              - S: (n) physical entity
              - S: (n) entity
  - *derivationally related form*
- S: (n) **conductor**
- S: (n) **conductor**
- S: (n) **conductor**

Figure 3: Inherited Hypernyms of the First Sense of *conductor* in WordNet

is "structure". To deal with this problem, such expressions as "names of", " titles of", "types of", "a list of", etc. are removed and the text is re-sent to the *Answer Type Recognition* module. Hence, "List the names of other millennium structures in England." is changed to "List the other millennium structures in England." on which the type recognition is applied one more time.

3.2 Document Retrieval

In this section, documents that are relevant to the target and the question are retrieved from the corpora and the Web. For this purpose, we need to generate appropriate queries and use a search engine tool.

Document retrieval is very important for Question Answering systems because it, along with *candidate answer extraction*, defines an upper bound on the recall we can achieve.

In the following sections, we will discuss how two different queries are generated, the query refinement method and the choice of search engines.

3.2.1 Query Generation

Query generation constitutes an extremely important phase of Question Answering because only an appropriate query can get appropriate answers.

We use both the corpora given and the Web as document collection sources. Two different search engines are exploited: *Lucene* to search on the corpora and *Google* to search on the Web. Both IR systems use different syntax for forming queries and they provide different capabilities and features, therefore two different query generation techniques are used to maximize the quality of their output.

Although the query generation techniques are different, they share the same basic methodology. We use the question and the target of the series to which the *List* question belongs for creating the queries. The question and the target are first processed separately. The process is the same for both the question and the target:

1. Stop words are removed from the text using a stop-list;
2. Terms surrounded by double quotations are left unchanged;
3. Consecutive capitalized words are quoted together as a single phrase. We quote consecutive capitalized words to form a phrase because they are very likely to refer to a proper noun. Proper nouns are perhaps the most important content words in a query;
4. The text within a pair of parentheses are processed recursively and treated as

an optional sub-query (i.e. the existence of its keywords is not necessary for a document to be retrieved, but it boosts the score of the document).

3.2.1.1 Lucene Query Generation

In order to search through the documents in the corpora, we use *Lucene*[3], an open source text search engine API written in Java (it provides several ports for other programming languages as well).

One of the useful facilities of Lucene is supporting fuzzy searches based on the Levenshtein Distance, or Edit Distance algorithm [Hatcher and Gospodnetic, 2004]. In the Edit Distance algorithm, the distance between two strings is measured as the minimum number of character deletions, insertions or substitutions needed to transform one string to the other. Using this capability, we can search for derived and inflected morphological forms of that word. For example, having the word *appeared*, we can retrieve documents containing such words as *appears*, *appear*, *appearing* and *appearance*. The similarity parameter can be changed in the *Query Refinement* module to allow a more lenient matching of the query keywords and the words in documents.

Proximity searches are also available in Lucene. Using this feature, words of a quoted phrase do not have to appear consecutively in a document in order for the document to be retrieved; instead, it allows to match words which are within a specific distance away [Hatcher and Gospodnetic, 2004]. For example, in searching for "USS Abraham Lincoln" (Q231.6 TREC-2007), documents containing phrases such as "USS aircraft group Abraham Lincoln", "USS Aircraft Carrier Abraham Lincoln" and "USS CVN-72 Abraham Lincoln" are retrieved. This feature also has a parameter indicating how far the words can appear in a document in order to be retrieved.

Using *Lucene's required operator* (i.e. '+'), we can indicate quoted phrases and non-quoted words as required and the terms derived from the text inside parentheses

[3] `http://lucene.apache.org`

as optional terms. The queries generated from the question and the target are, then, concatenated to constitute a single query.

3.2.1.2 Web Query Generation

The second IR tool that we use is the *Google*™ search API [4] to retrieve Web documents relevant to the target and the question.

The Google search API offers a feature for query expansion. The Google synonym operator, ~ (tilde), expands the query to include the matches of the keywords, synonyms and also words that are somehow semantically related to the query words. For example, searching Google for "~ car", we can retrieve documents containing *car, cars, motor, auto, vehicle, automobile, automotive* etc. [Dornfest *et al.*, 2006]. We use this feature to widen the search criteria for single words in the query.

The Google API provides very useful capabilities to search the Web. However, it imposes some limitations on developers, which might affect the way a query is composed. For example, the length of the search request should not exceed 1028 bytes and the maximum number of words in a query is 10 [Long *et al.*, 2004]. Therefore, the Google API simply ignores any extra words at the end of a query.

The order in which the keywords appear in the query matters for Google. It favors the keywords in order of their occurrences; so the first keyword is the most important and the last one is the least important keyword [Miller, 2006]. This and also the limitation by which the keywords after the tenth are ignored indicate that the terms should appear based on their importance. Since quoted phrases are very likely to be proper nouns, the query starts with them, followed by single words, and optional sub-queries which, in turn, appear in the same order: i.e. optional quoted phrases and optional single words.

[4] `http://code.google.com/apis/soapsearch/`

3.2.2 Query Refinement

The queries generated using the question and the target might be too strict and result in only a few documents and thereby a low recall. At this stage, if the number of hits is below a certain threshold, the query is loosened. Of course, different thresholds and different techniques are used for the Google and Lucene query refinements.

3.2.2.1 Lucene Query Refinement

Since the number of documents in the corpora, compared to the Web, is low and the documents are rather small in size, different techniques are used to gradually loosen the Lucene query in order not to bring in remarkable noise. The search criterion is iteratively loosened until the number of hits is greater than a certain threshold. The steps of relaxing a query used in this module are:

1. Increasing the proximity parameter for quoted phrases (i.e. the distance allowable between words of a phrase);
2. Decreasing the similarity parameter for fuzzy search;
3. Breaking up a quoted phrase into separate words;
4. Changing a required term to be optional.

These techniques can be used separately or together, but they should be used in order of their effect. For example, because increasing the proximity parameter is less likely to bring noise than removing one word from the required words, it should be applied first.

Table 6 illustrates the query refinement steps of the question 232.6 "Which airlines use Dulles?". The number of hits obtained using each query is also shown. The iteration continues until a minimum number of documents are retrieved. As the figure shows in the third iteration, having made "use" optional and decreased the

fuzzy search factor of "airlines" by 0.1, we can notably increase the number of hits matching the query.

Iteration	Query	Hits
1	"dulles airport" +"dulles" +airlines~0.8 +use~0.8	9
2	"dulles airport"~5 +"dulles" +airlines~0.7 +use 0.7	40
3	"dulles airport"~5 +"dulles" +airlines~0.6 use	146

Table 6: Web Query Refinement Steps of the Question 232.6 "Which airlines use Dulles?"

3.2.2.2 Web Query Refinement

In the Web query refinement, however, because of the huge number of documents on the Web, fewer techniques for query refinement are used. They are :

1. Breaking up a quoted phrase into separate words;
2. Changing required terms to be optional.

These techniques are applied one by one until the number of hits is more than a certain threshold.

3.2.3 Document Retrieval

Once the queries have been created and refined, relevant documents are retrieved using the *Lucene* and *Google* search engines. Two document collection are created; *source document collection* and *domain document collection.*

The *source document collection* is used to extract candidate answers from. It consists of a small number of corpora documents and Web documents. On the other

hand, the *domain document collection* is used to extract relations between the candidate answers from. As shown in Figure 4, the *source document collection* is, indeed, a subset of *domain document collection*, which contains the most relevant documents from both the corpora and the Web.

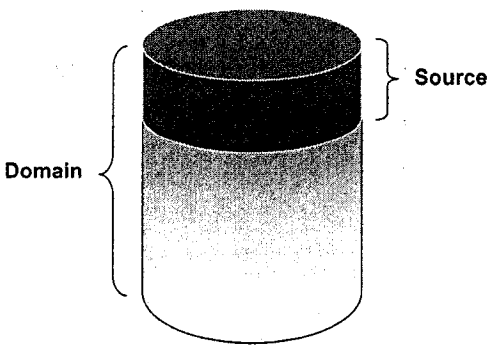

Figure 4: The Source Document Collection is a Subset of the Domain Document Collection

Although the proportion of the number of documents from the Web over the number of documents from the corpora in the domain collection affects the final results, this ratio is more influential in the source collection. Experiments show that more documents should be used from the corpora than from the Web to maximize the result. More details will be provided in Chapter 5.

We only use HTML files on the Web among all different types of files available (e.g. doc, pdf, ppt, ps, rtf, etc.). These files are downloaded and stored on the hard disk for further use. This saves time considerably in the training phase as it takes a few seconds to download each web page and a few minutes to download all required web pages for each question.

3.3 Candidate Answer Extraction

In this section, we extract from the source documents an initial list of candidate answers. The system will stick to this list from this step on. Therefore, the upper bound of recall is set at this phase.

In fact, the initial list of candidate answers can be imported from any source. In particular, we could use the candidate answers of the question generated by a *Factoid* question answerer. In Chapter 5, we will compare results of our approach with and without using an external source of candidate answers.

3.3.1 Term Extraction

For term extraction, we apply a simple method: All terms that conform to the answer type are extracted from the source document collection. Depending on the answer type, the candidate terms are extracted using different approaches. Because no NE tagger is perfect, we use three different Name Entity taggers: Alias-i's *LingPipe*[5] Named Entity tagger, *Stanford Named Entity Recognizer*[6], and Gate's *ANNIE*[7] Named Entity Recognizer [Cunningham *et al.*, 2002] [Cunningham *et al.*, 2007].

3.3.1.1 Person Type

Since the free softwares that we used for Named Entity Recognition (NER) were not adequately accurate, we considered to combine the NE taggers. The best result for questions of the type PERSON in the training set is acquired when two different NE taggers are used together. We use the *LingPipe* Named Entity tagger and the *Stanford Named Entity Recognizer* to pull out person entities from the source document collection. In this method, only entities detected by both NE taggers are extracted. Although this approach might lower the recall, it enhances the precision by avoiding

[5] http://www.alias-i.com/lingpipe
[6] http://nlp.stanford.edu/ner/
[7] http://gate.ac.uk/ie/annie.html

noise and it also reduces the time needed to answer the question (although time is not one of our quality criteria).

Figure 5 compares the F-score of seven different combination of the three NE taggers that we have. As shown, the best case is the combination of the Stanford and the LingPipe NE taggers.

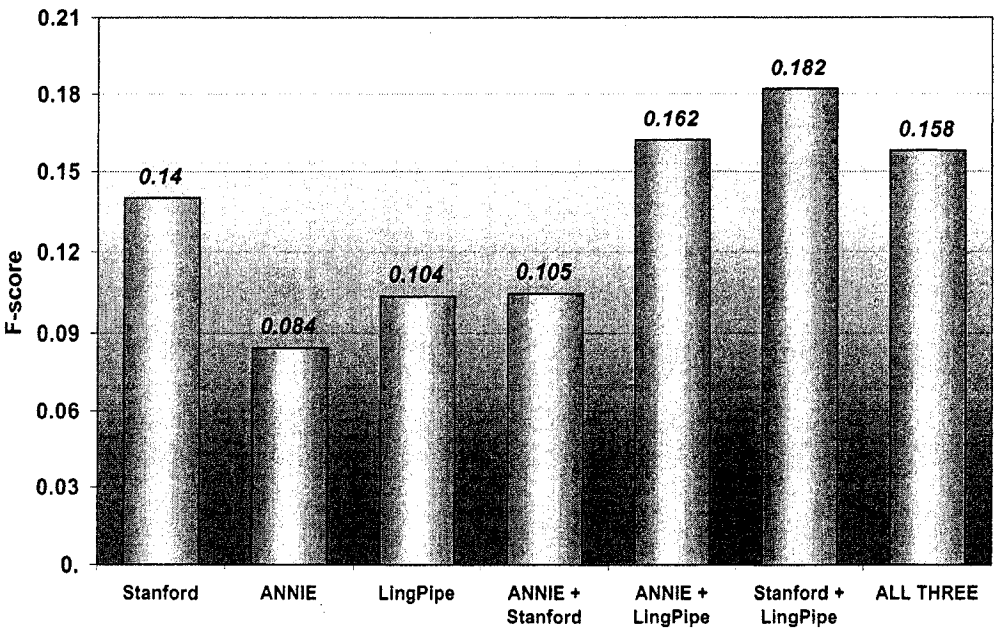

Figure 5: Results of Different Combinations of NE Taggers for PERSON

3.3.1.2 Organization Type

We use the same technique for extracting organization entities as we do for the PERSON type: we take the intersection of detected entities from two different NE taggers. As Figure 6 shows, combining the NE taggers increases the F-score of the ORGANIZATION questions in the training set. We take the intersection of the *Stanford* and the *ANNIE* NE taggers as it gives a slightly better result on the ORGANIZATION questions in the training set than other combinations.

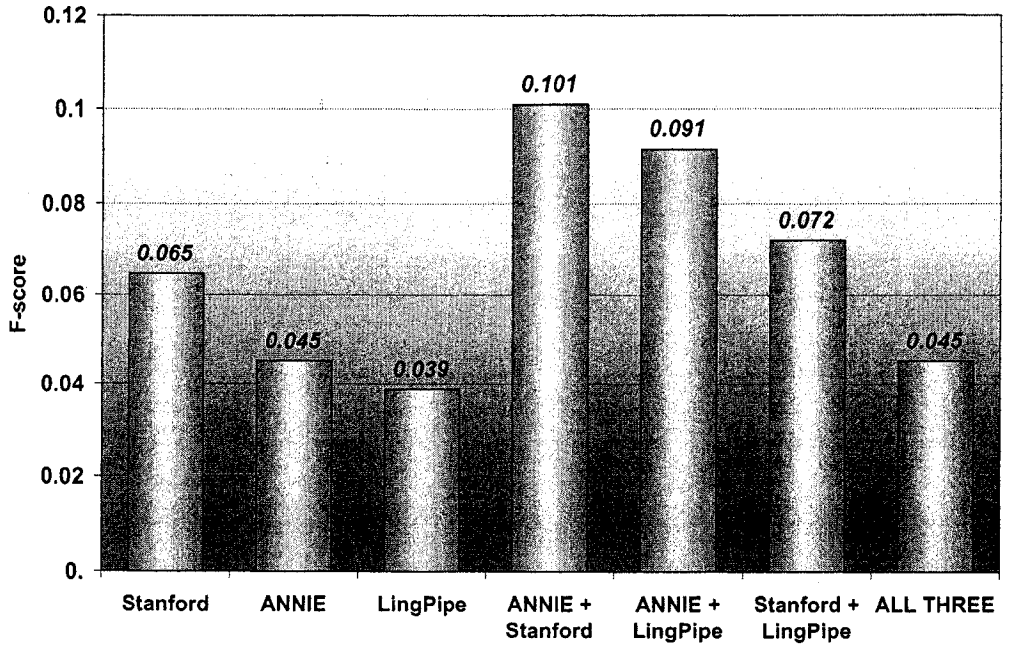

Figure 6: Results of Different Combinations of NE Taggers for ORGANIZATION

3.3.1.3 Job Type

The *ANNIE Named Entity Transducer* is exploited to annotate job titles using a gazetteer list. Although this NE tagger does not perform accurately, the other two NE taggers we use are not able to extract job titles.

3.3.1.4 Country Type

A gazetteer containing the names of all countries in the world is used to extract COUNTRY entities. This gazetteer, along with the country official names, possibly contains the initials or other known names for countries, which can be used for coreference resolution. For example, the country gazetteer includes the following record:

"United States of America = United States = The States = America = USA = U.S.A = U.S"

This record lists all common names of the United States and allows recognition of any such entity as a COUNTRY named entity that corefer to other occurrences. An

application of *Coreference Resolution* will be discussed in Section 3.3.3.

3.3.1.5 Nationality Type

NATIONALITY entity types are recognized using a gazetteer. It contains all nationalities corresponding to the countries in the country gazetteer.

3.3.1.6 State Type

Recognition of STATE entity types are also done using a gazetteer. The state gazetteer contains the names of all fifty states in the U.S. It also contains the states' abbreviations for coreference resolution.

3.3.1.7 City Type

CITY named entities are also detected using a gazetteer containing 8,000 cities around the world. This list does not contain all the cities in the world, but only medium and large-sized cities. The definition of a medium or a large-sized city is that there is an airport associated to the city [8].

3.3.1.8 Movie Type

Since there is no free NE tagger that annotates MOVIE titles (to our knowledge), we use two different methods to extract phrases that are likely to be movie titles.

First, all consecutive capitalized words and text in quotes are extracted from the source documents. Second, we use the IMDB (Internet Movie DataBase) website[9] as a source for extracting candidates. In this method, we search the IMDB website using the Google query generated and the Google API and we take the first page that satisfies the query. This IMDB page is downloaded and all the titles in the page that

[8] These cities are extracted from `http://www.world-airport-codes.com`

[9] `www.imdb.com`

have a link to a page under "*http://www.imdb.com/title*" subdomain are extracted. The query is relaxed if the number of hits is low.

The candidates extracted using these two methods are appended and sent to the *Term Verification* module (see Section 3.3.2).

3.3.1.9 Other Type

For the OTHER type of entities, we simply extract all consecutive capitalized words and also terms in quotations. This candidate list usually contains a huge number of terms, which requires the list to be narrowed down. In the *Term Verification* module, we use the frequency of each term on the Web as a way to shrink down the list.

3.3.2 Term Verification

After creating the initial candidate list, we need to verify whether or not a term complies with its entity class and also remove candidates that are not likely to be the answer. This is not needed for types for which a gazetteer is used because all verification can be applied on the gazetteer beforehand.

Running NE taggers on HTML pages and blog pages introduces some noise that can be easily avoided by removing stop words and terms that are too short or too long. This is why stop words are removed from the initial list. This increases the precision significantly with a negligible impact on recall. It can lower the recall only when a stop word is an homonym of an instance of the answer. For example, *she*, which usually is considered as a stop word, can refer to an ethnic group of China, a novel by H. Rider Haggard, a movie adapted from the novel, a music band, several music songs, several music albums, or the abbreviation of *Standard Hydrogen Electrode*. However, this is rarely the case and hence removing stop words is much more beneficial than keeping them.

In addition, the NE taggers return some false positives that suffer from their

length, resulted from using unstructured HTML pages, malfunctioning of HTML to text convertor, using blogs, extracting terms in quotes for MOVIE and OTHER types (which might include citation), etc. These too short or too long candidate answers can easily be filtered out to increase the precision and reduce the run time. Therefore, terms longer than a certain threshold or shorter than another threshold are removed from the initial list of candidates.

The *Term Verification* module, specifically targets the MOVIE and the OTHER types where all consecutive capitalized words and text in quotations are extracted as instances of MOVIE or OTHER classes with no syntactic or semantic clues for being such.

3.3.2.1 Movie Type Verification

Due to the way the initial list of MOVIE candidates is constituted, the number of terms in the list is large even after removing stop words and short and long terms. We still need to eliminate more unlikely candidates. For this purpose, a query is generated for each candidate term.

Using this query, IMDB webpages that satisfy the Google query (derived from the *Query Generation* module, see Section 3.2.1) and also contain the candidate term in its title are counted. We use the number of hits as a feature for how likely a term is the answer. More formally, this is defined as:

numberOfHits(*GoogleQuery* **intitle:***Term* **site:** www.imdb.com)

If the number of hits is below a certain threshold, the term is dropped from the candidate list. In fact, the content of the IMDB webpages containing the term does not matter, but the number of pages that satisfy the query constraints. The Google

API provides the number of hits for each search query.

3.3.2.2 Other Type Verification

The obstacle we faced for MOVIE entity type, the huge number of candidate answers, also applies for the OTHER type. However, we do not have such a website for filtering out the false positives.

Instead, we exploit a similar technique for validating the OTHER type of entities. We take the number of hits in which the term appears adjacent of its subtype normalized by how frequent the term is per se.

$$\frac{\textbf{numberOfHits}(\text{“}\textit{SubType Term}\text{”}\ \textbf{OR}\ \text{“}\textit{Term SubType}\text{”})}{\textbf{numberOfHits}(\text{“}\textit{Term}\text{”})}$$

The ratio must be higher than a threshold for a term to be considered as a potential candidate answer and remain in the list.

The idea is that, for a relatively high number of times, an entity should appear next to its type. For example, searching on the Web for the book title “After Dark”, we find a number of cases in which “After Dark” appears next to the word “book”. These cases usually happen in appositives (both restrictive and non-restrictive), inside parentheses, etc.

...After Dark (book)...

...After Dark, book 1...

...After Dark book series...

...the book After Dark...

...book: After Dark...

...new book, After Dark...

3.3.3 Coreference Resolution

Two types of coreferences are used in our system; terms that always corefer (global coreferences) and terms that corefer within the scope of the target (local coreferences). United States and U.S.A are examples of global coreferences. On the other hand, there are coreferences specific to the target and its context. For example, Question 143.6 of TREC-2006 is "Who have been "scholars" at the Institute?" (American Enterprise Institute). Among the acceptable answers, "Lawrence Lindsey", "Larry Lindsey", "Lawrence B. Lindsey", "Larry B. Lindsey" and "Lindsey" corefer to one another. Although the word "Lindsey" can refer to several places, institutions, fictional characters and many people with the same forename or surname, in the scope of the target, "American Enterprise Institute", we can assume, with a high confidence, that "Lindsey" only refers to "Lawrence B. Lindsey", a scholar in *American Enterprise Institute*, and not to any other ones.

As mentioned in Section 3.3.1, we have included all country-name coreferents and state abbreviations in their gazetteers. These are global types of coreferences. Local coreferences, in our system, are only used in PERSON type of entities. Generally, we make all person forenames and person surnames corefer to the full name (i.e. forename middle name and surname together). Thus, forenames can refer to surnames through the full name. This approach has a shortcoming when two different person entities appear within the relevant documents that happened to have the same first name or surname. However, this circumstance is too rare to be dealt with.

Local coreferences are extracted for question series to another, while global references are permanent and are kept for all questions.

Coreference resolution can improve the results. First, it improves the precision by avoiding the inclusion of coreferent answers in the final candidate list, in which

case, only one instance is deemed correct and other coreferents are regarded as unacceptable. Second, it can augment the recall since coreferents are counted for one another in the *Co-Occurrence Information Extraction* phase. This issue will be discussed more in Section 3.4.2. Therefore, coreference resolution can enhance the final results, F-score, by impacting both precision and recall.

3.4 Co-Occurrence Information Extraction

Once a list of initial candidate answers has been extracted, the similarity between each pair of candidates is computed based on how frequently they co-occur within the domain documents. This information is used by the *Candidate Answers Selection* module to select terms that tend to co-occur more often.

We use the co-occurrence information of a pair of terms as an indicator of their semantic similarity [Harris, 1954]. The correlation between co-occurrence and similarity has been shown by many researchers, for example [Spence and Owens, 1990] and [Miller and Charles, 1991].

Two terms $term_i$ and $term_j$ are said to have first order co-occurrence if they co-occur within a sentence. If $term_i$ co-occurs with $term_k$ in some sentences and $term_k$ co-occurs with $term_j$ in other sentences, $term_i$ and $term_j$ are said to share a second order co-occurrence via $term_k$ [Chakraborti *et al.*, 2007][Miller and Charles, 1991]. Higher orders of co-occurrence can similarly be defined. In our approach, we use first order co-occurrence as well as second order co-occurrence through target and question keywords.

3.4.1 Co-Occurrence Matrix

Co-occurrence of two candidate terms can be computed within documents, within paragraphs or even within sentences. Since there are not many documents about

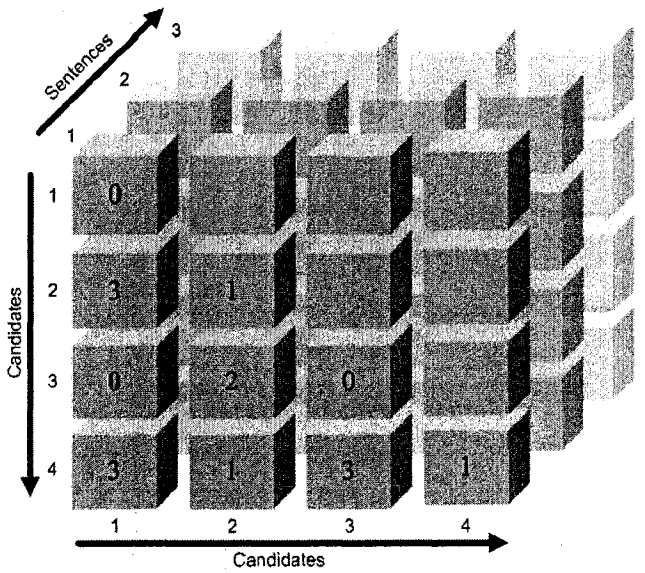

Figure 7: Example of a Co-Occurrence Matrix

each target (in the order of tens to a few hundreds), the co-occurrence information at the document level will not be statistically significant. The same reason also applies to paragraph-based co-occurrence information to some extend. We use co-occurrence at the sentence level since there are more instances to extract co-occurrence information from and hence it is more reliable.

The documents in the domain collection are split into sentences. These sentences are checked as to whether they include one or more candidate terms. The information regarding the occurrence and co-occurrence of the candidate terms are stored in a 3D co-occurrence matrix: sentence-candidate-candidate. Figure 7 illustrates an example of a co-occurrence matrix with four candidate terms and three sentences, in which the first sentence contains $term_2$ and $term_4$, but does not contain $term_1$ or $term_3$.

Each sentence S_k is represented using a 2D matrix. Since co-occurrence is a symmetric relation, we only use the lower triangular (aka left triangular) section of the matrix even though different values are assigned to each entry based on the appearance of the first and the second term.

$$M_i = \begin{bmatrix} l_{0,0} & & & \\ l_{1,0} & l_{1,1} & & \\ \vdots & \vdots & \ddots & \\ l_{n,0} & l_{n,1} & \dots & l_{n,n} \end{bmatrix}$$

The entries of this matrix are defined as below.

for $i < j$:

$$l_{i,j} = l_{j,i}$$

for $i > j$:

$$l_{i,j} = \begin{cases} 0 & term_i \text{ and } term_j \text{ did not occur within the sentence } S_k \\ 1 & term_i \text{ and } term_j \text{ co-occurred within the sentence } S_k \\ 2 & term_i \text{ occurred within the sentence } S_k \text{ but not } term_j \\ 3 & term_j \text{ occurred within the sentence } S_k \text{ but not } term_i \end{cases}$$

for $i = j$:

$$l_{i,i} = \begin{cases} 0 & term_i \text{ did not appear in the sentence } S_k \\ 1 & term_i \text{ appeared in the sentence } S_k \end{cases}$$

3.4.2 Dealing with Coreferences

The coreferences of each term, if any, is taken into account in co-occurrence information extraction when the term does not appear in a sentence. So, if a sentence does not contain a term, the coreferents of the term are checked, based on the idea that any term can be replaced with any of its coreferents. Considering coreferences in term co-occurrence can improve the recall since coreferents are counted for one another. In fact, coreferents are considered as one term and this will affect the original term's combined relation to other terms, thereby placing it in the top favorite candidates for the system.

3.5 Candidate Answer Selection

The purpose of *Candidate Answer Selection* is to narrow down the candidate list and select a subset of it that is most likely to contain the answer. This module is necessary since we have extracted all terms complying with the predicted answer type from the source documents and no further analysis has been made.

In this section, we present and investigate two different approaches to returning a subset of candidate answers based on the co-occurrence information extracted in the previous stage. First, a clustering method is used to group closely related terms. Candidates in the most appropriate cluster are returned as the final candidates. Second, we use a simple classification method to classify the candidates into two classes: *Relevant* and *Irrelevant*. Then, the candidates in the class *relevant* are returned. The evaluation of each method and also their comparison will be provided in Chapter 5.

3.5.1 Clustering

Clustering is the unsupervised classification of data instances into groups (clusters) of similar objects. In fact, by clustering the data into groups, we try to model

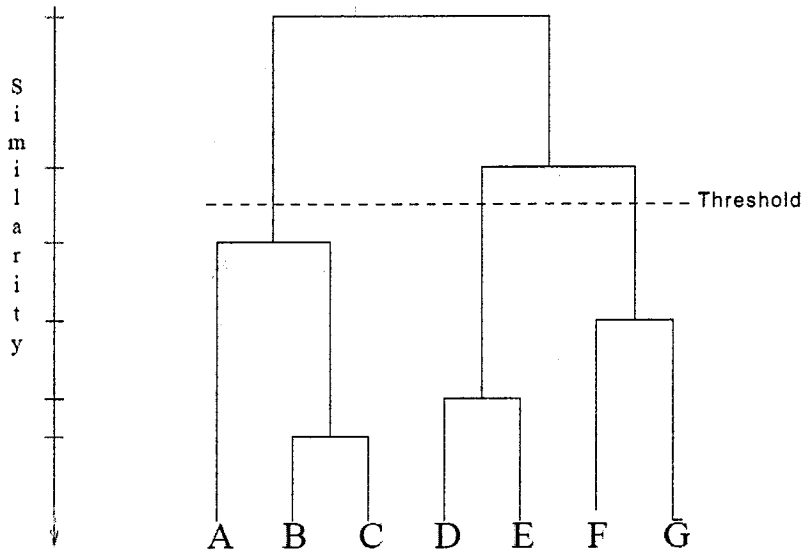

Figure 8: Example of a Dendrogram, Adapted from [Jain *et al.*, 1999]

the data. Although modeling the data by fewer clusters loses certain fine details, it brings simplification. Clustering methods are either *Partitional* or *Hierarchical* [Berkhin, 2002] [Jain *et al.*, 1999].

While Partitional clustering methods create a number of clusters directly, Hierarchical clustering builds a hierarchy of clusters. The tree created using this approach is called a *dendrogram.* Figure 8 shows an example of a dendrogram.

Hierarchical clustering can be further categorized into *agglomerative* or bottom-up and *divisive* or top-bottom. Agglomerative approaches start with singleton clusters and iteratively merges clusters into larger clusters until satisfying some criterion. Divisive clustering, on the other hand, starts with one cluster containing all data points and iteratively breaks a cluster into successively smaller clusters until a certain condition holds [Berkhin, 2002].

An important issue when choosing a clustering method is the *linkage metric*, which is used to define the (dis)similarity between two clusters. Major inter-cluster linkage metrics are *single linkage* (nearest neighbor), *complete linkage* (farthest neighbor) and

average linkage. The similarity between two clusters is the maximum similarity among all pairs of elements from the two clusters in single linkage, the minimum similarity among all pairs of elements in complete linkage and the average of all similarities in average linkage [Jain *et al.*, 1999].

3.5.1.1 Hierarchical Agglomerative Clustering (HAC)

Since there is no information to represent each candidate term as a data instance and the only information we have is the similarity between candidate terms, many clustering methods can not be applied. We use a Hierarchical Agglomerative clustering method which can be used in similar situations. In Chapter 5, results derived using the Hierarchical Agglomerative method with single linkage metric, complete linkage and also average linkage are compared.

The algorithm of HAC clustering is as follows:

1. Remove terms that appear less than a certain threshold.
2. Sort the candidate answers based on their frequency in the domain collection.
3. Put each candidate term t_i in a separate cluster C_i.
4. Compute the similarity between each pair of clusters.
   - In average-linkage clustering, the similarity between two clusters C_i and C_j is the average of all similarities between terms t_m in C_i and terms t_n in C_j.

$$\textbf{Similarity}(\mathbf{C_i}, \mathbf{C_j}) = \frac{1}{|C_i| \times |C_j|} \sum_{t_m \in C_i} \sum_{t_n \in C_j} Similarity(t_m, t_n)$$

   - In single linkage, the similarity between two clusters C_i and C_j is the

maximum of all similarities between terms t_m in C_i and terms t_n in C_j.

$$\textbf{Similarity}(\mathbf{C_i}, \mathbf{C_j}) = \max_{t_m \in C_i \; t_n \in C_j} (Similarity(t_m, t_n))$$

- In complete linkage, the similarity between two clusters C_i and C_j is the minimum of all similarities between terms t_m in C_i and terms t_n in C_j.

$$\textbf{Similarity}(\mathbf{C_i}, \mathbf{C_j}) = \min_{t_m \in C_i \; t_n \in C_j} (Similarity(t_m, t_n))$$

5. Merge two clusters that have the highest similarity between them.

6. Goto step 3 until there are only N clusters left or the highest similarity is less than a threshold.

In the first step, we remove candidates that do not occur frequently in the domain documents. This also considerably reduces the amount of time needed for clustering as the time complexity of agglomerative clustering algorithms is usually $O(n^2 \log n)$ to compute n^2 similarities and sort them (single linkage can be implemented in $O(n^2)$).

3.5.2 Similarity Measures

We measure the similarity between two candidate terms as a normalized value denoting how often they co-occur within sentences of the domain documents. This value is normalized by the frequencies of the terms. Once all the data regarding term appearances and co-occurrences is collected, the similarity between each pair of terms is computed.

To compute the similarity between two terms, $term_i$ and $term_j$, a 2×2 contingency table is used [Manning and Schütze, 1999].

	$term_i$	$\neg term_i$	Total
$term_j$	O_{11}	O_{21}	$O_{11} + O_{21}$
$\neg term_j$	O_{12}	O_{22}	$O_{12} + O_{22}$
Total	$O_{11} + O_{12}$	$O_{21} + O_{22}$	N

Table 7: 2 × 2 Contingency Table Containing Co-Occurrence Information of Two Terms

where:

O_{11} : Number of sentences in which $term_i$ and $term_j$ co-occurred;

$S_{11} = \{ S_k \mid S_k[i,j] = 1 \}, \qquad O_{11} = |S_{11}|$

O_{12} : Number of sentences in which $term_i$ appeared but $term_j$ did not appear;

$S_{12} = \{ S_k \mid S_k[i,j] = 2 \}, \qquad O_{12} = |S_{12}|$

O_{21} : Number of sentences in which $term_i$ did not appear but $term_j$ appeared;

$S_{21} = \{ S_k \mid S_k[i,j] = 3 \}, \qquad O_{21} = |S_{21}|$

O_{22} : Number of sentences containing neither $term_i$ nor $term_j$;

$S_{22} = \{ S_k \mid S_k[i,j] = 0 \}, \qquad O_{22} = |S_{22}|$

N : Total number of sentences in the domain collection.

$N = |S| \qquad or \qquad N = O_{11} + O_{12} + O_{21} + O_{22}$

3.5.2.1 Chi-Square Test

The Chi-square (χ^2) test is used to compare observed frequencies of terms with frequencies expected to hypothesize their independency [Manning and Schütze, 1999]. In our case, we would like to see if two terms have co-occurred by chance or if they are indeed related to each other. Among several Chi-square tests, Pearson's Chi-square is the original and the most widely-used test. The χ^2 statistic sums the difference

between observed and expected frequencies, scaled by the magnitude of the expected frequencies:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where:

$\mathbf{O_i}$: an observed frequency;

$\mathbf{E_i}$: an expected (theoretical) frequency;

$\mathbf{n}$: the number of possible outcomes of each event.

The χ^2 value for a 2×2 contingency table can be computed using the following formula [Manning and Schütze, 1999].

$$\chi^2 = \frac{N\,(O_{11}O_{22} - O_{12}O_{21})^2}{(O_{11} + O_{12})(O_{11} + O_{21})(O_{12} + O_{22})(O_{21} + O_{22})}$$

The similarity between each pair is computed and stored in a similarity matrix. This matrix is an $N \times N$ symmetric matrix denoting the relation value between each pair of terms. The similarity between each term and itself is zero by definition.

$$Similarity(term_i,\ term_i) = 0$$

$$Similarity = \begin{bmatrix} 0 & \chi^2_{1,0} & \cdots & \chi^2_{n,0} \\ \chi^2_{1,0} & 0 & & \chi^2_{n,1} \\ \vdots & \vdots & \ddots & \vdots \\ \chi^2_{n,0} & \chi^2_{n,1} & \cdots & 0 \end{bmatrix}$$

3.5.2.2 Yates' Chi-Square Test

The χ^2 statistic significantly favors the similarity of two low-frequent terms that happened to co-occur most of the times they appeared. However, since the frequencies of the terms in the relevant documents (domain documents) are used, we do not expect potential answers to have a low frequency.

Yates' chi-square test (aka Yates' correction for continuity) is used when at least one cell of the contingency table has an expected frequency less than 5 (10 in some sources [Adler, 1951]). [Yates, 1934] suggests to subtract 0.5 from the difference between each observed value and its expected value in the contingency table:

$$\chi^2_{\mathbf{yates}} = \sum_{i=1}^{n} \frac{(|O_i - E_i| - 0.5)^2}{E_i}$$

Therefore, the statistic for a 2×2 contingency table changes to:

$$\chi^2_{\mathbf{yates}} = \frac{N\left(|O_{11}O_{22} - O_{12}O_{21}| - N/2\right)^2}{(O_{11} + O_{12})(O_{11} + O_{21})(O_{12} + O_{22})(O_{21} + O_{22})}$$

In fact, Yates' correction statistic prevents overestimation of statistical significance for small data. However, it may tend to overcorrect in some cases. This is why there are many discussions about using Yates' correction in the scientific community. Some statisticians argue that it should be used when expected frequencies are less than 5, others use 10, and yet others argue that Yates' correction should be used in all 2×2 tables. Some sources also say that it should not be used for 2×2 contingency tables at all [Miettinen, 1974].

In our work, we used both similarity measures and the results using the Yates' correction and the Chi-square test are compared in Chapter 5.

3.5.2.3 Mutual Information

In addition to Chi-square and Yates' correction statistic, we have experimented with an alternative similarity measure based on the information theoretic concept of *Pointwise Mutual Information* (PMI). PMI tells us about the amount of information one point contains about the other, or in other words, the reduction in uncertainty of one point due to knowing about the other [Manning and Schütze, 1999].

According to [Fano, 1961], the pointwise mutual information between two points (words) x and y with probabilities $P(x)$ and $P(y)$ is defined as follows:

$$\mathbf{I(x,y)} = \log_2 \frac{P(x,y)}{P(x)\,P(y)}$$

Pointwise Mutual Information compares the joint probability of x and y with the probabilities of observing x and y independently (chance). If there is an association between x and y, then the joint probability is much greater than chance $P(x,y) \gg P(x)P(y)$, and consequently $I(x,y) \gg 0$ [Church and Hanks, 1990]. If x and y are not related, we expect $P(x,y) \approx P(x)P(y)$ and hence $I(x,y) \approx 0$. Similarly, if the

joint probability is much less than chance $P(x,y) \ll P(x)P(y)$, then $I(x,y) \ll 0$.

In our case, we are interested to see if two terms co-occurred by chance or if there is an association between them (i.e. the number of co-occurrences is significantly greater than it would be by chance). The interpretation when $I(term_i, term_j) \ll 0$ is that there is a dissociation between two terms rather than an association [Resnik, 2001]. It happens when the two terms are quite frequent in the corpus but unrelated or in other words, as [Church and Hanks, 1990] states, they are in complementary distribution.

We use maximum likelihood estimates to compute the probabilities. Therefore:

$$\mathbf{I(term_i, term_j)} = \log_2 \frac{P(term_i, term_j)}{P(term_i)\,P(term_j)}$$

$$= \log_2 \frac{\frac{F(term_i, term_j)}{N}}{\frac{F(term_i)}{N}\,\frac{F(term_j)}{N}}$$

$$= \log_2 \frac{N \times F(term_i, term_j)}{F(term_i)\,F(term_j)}$$

$$= \log_2 \frac{N \times O_{11}}{(O_{11}+O_{12})\,(O_{11}+O_{21})}$$

Where, $F(term_i)$ and $F(term_j)$ refer to the number of sentences in which $term_i$ and $term_j$ appear and N, O_{11}, O_{12} and O_{21} are values in the contingency table of them (discussed in Section 3.5.2).

[Manning and Schütze, 1999] discuss the shortcoming of Pointwise Mutual Information in dealing with sparseness with considering two extreme cases: perfect dependence of the occurrence of two words and perfect independence of occurrence of them. In case of perfect independence:

$$\begin{aligned}
\mathbf{I(term_i, term_j)} &= \log_2 \frac{P(term_i, term_j)}{P(term_i)\,P(term_j)} \\
&= \log_2 \frac{P(term_i)\,P(term_j)}{P(term_i)\,P(term_j)} \\
&= \log_2 1 = 0
\end{aligned}$$

for perfect dependence, we have:

$$\begin{aligned}
\mathbf{I(term_i, term_j)} &= \log_2 \frac{P(term_i, term_j)}{P(term_i)\,P(term_j)} \\
&= \log_2 \frac{P(term_i)}{P(term_i)\,P(term_j)} \\
&= \log_2 \frac{1}{P(term_j)} = \log_2 \frac{1}{P(term_i)}
\end{aligned}$$

So in perfect dependence of two terms, as the words get rarer, their mutual information increases. However, a suitable measure for us is one that assigns higher scores to more frequent terms as there is more evidence of them in the corpus. [Manning and Schütze, 1999] concludes that mutual information is a good measure of independence but not a proper measure of dependence because for dependence the

score depends on the frequency of the individual terms and it is biased towards infrequent terms. Two solutions to compensate for the bias of mutual information may be 1) considering terms with a frequency of higher than a threshold and 2) multiplying the statistic with a coefficient. An extension of Pointwise Mutual Information with a coefficient (referred to as Weighted Mutual Information (WMI) in our work) is defined as:

$$\mathbf{WMI}\,(\mathbf{term_i},\,\mathbf{term_j}) = O_{11} \times \log_2 \frac{N \times O_{11}}{(O_{11} + O_{12})\,(O_{11} + O_{21})}$$

In this definition, the original Mutual Information between two terms is weighed by the number of their co-occurrences (O_{11}). However, this definition of WMI is not appropriate either. Suppose, F_1 and F_2 are the number of sentences in which $term_i$ and $term_j$ appear within a certain collection ($F_1 = O_{11}+O_{12}$ and $F_2 = O_{11}+O_{21}$). For two specific words with fixed F_1 and F_2, we assume that the number of co-occurrences of the two terms is a variable value between 0 and $min(F_1, F_2)$, Figure 9 shows the correlation between the number of co-occurrences and Mutual Information using the new definition.

As the figure illustrates, WMI is zero where the number of co-occurrences is exactly the same as expected, $O_{11} = \frac{F_1 \times F_2}{N}$. Although the function is not defined at $O_{11} = 0$ (since $\log 0 = -\infty$), the Weighted Mutual Information value gets close to 0 as O_{11} approaches 0. As the number of co-occurrences (O_{11}) grows from 0, the WMI value decreases until the turning point, at which the slope equals zero. The behavior of the function is undesirable between these two points since the similarity value of two terms should increase as there are more co-occurrences between them. From the turning point onward, the value of WMI increases as we would expect from a suitable similarity measure.

A solution can be to change the definition of the function for points between

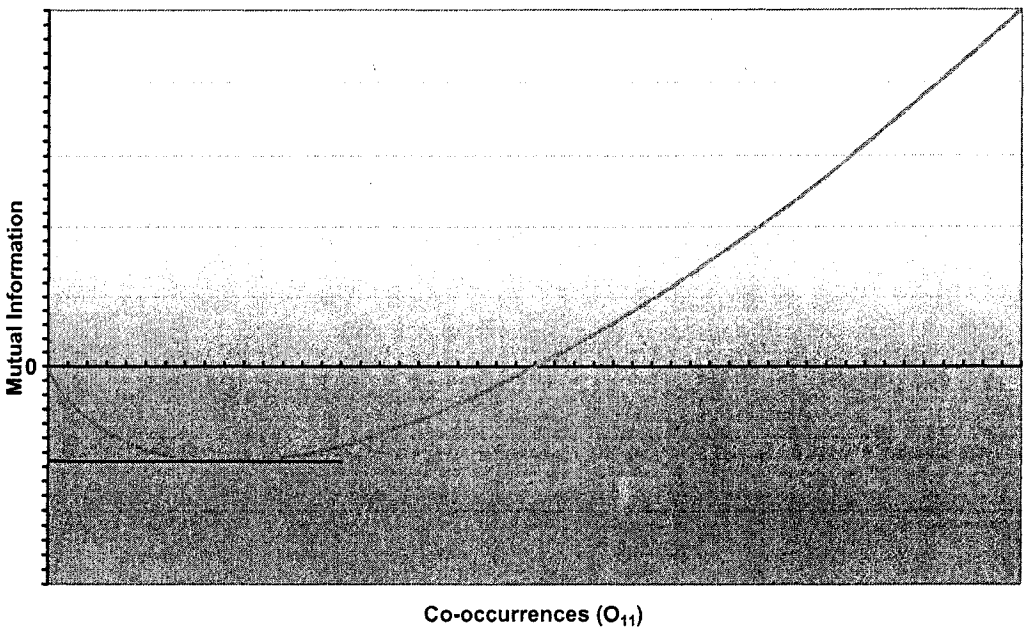

Figure 9: Typical Sketch of Weighted Mutual Information

$O_{11} = 0$ and the turning point as we want the WMI value increases while the number of co-occurrences increases. There are many ways to redefine the function for the mentioned subdomain. To preserve the MI original nature, we flip the subgraph vertically down around the tangent line at the turning point (see Figure 10). In fact, we change the turning point to an inflection point.

To find the turning point, we have:

$$\begin{aligned}
\mathbf{WMI}\,(\mathbf{term_i}, \mathbf{term_j}) &= O_{11} \times \log_2 \frac{N \times O_{11}}{(O_{11} + O_{12})\,(O_{11} + O_{21})} \\
&= O_{11} \times \log_2 \frac{N \times O_{11}}{F_1 \times F_2} \\
&= O_{11} \times \log_2 K O_{11} \qquad (K = \frac{N}{F_1 \times F_2})
\end{aligned}$$

where K is a constant for each pair of terms.

$$\begin{aligned}
\mathbf{WMI'} &= \log_2 KO_{11} + \frac{O_{11}}{O_{11} \times \ln 2} \\
&= \log_2 KO_{11} + \log_2 e \\
&= \log_2 KO_{11}e \\
&= 0
\end{aligned}$$

$$KO_{11}e = 1 \qquad \Rightarrow \qquad O_{11} = \frac{1}{Ke} = \frac{F_1 \times F_2}{N \times e}$$

The value of WMI at the turning point is:

$$\begin{aligned}
\mathbf{WMI}\,(\mathbf{O_{11}} = \frac{\mathbf{F_1} \times \mathbf{F_2}}{\mathbf{N} \times \mathbf{e}}) &= \frac{F_1 \times F_2}{N \times e} \times \log_2 \frac{N \times \dfrac{F_1 \times F_2}{N \times e}}{F_1 \times F_2} \\
&= \frac{F_1 \times F_2}{N \times e} \times \log_2 \frac{1}{e} \\
&= -\frac{F_1 \times F_2}{N} \times \frac{\log_2 e}{e}
\end{aligned}$$

Having found the coordinates of the turning point, we can define the modified Weighted Mutual Information statistic:

$$\mathbf{WMI}\,(\mathbf{term_i},\,\mathbf{term_j}) = \begin{cases} -2 \times \dfrac{F_1 \times F_2}{N} \times \dfrac{\log_2 e}{e} & \text{if } O_{11} = 0 \\ -2 \times \dfrac{F_1 \times F_2}{N} \times \dfrac{\log_2 e}{e} + O_{11} \times \log_2 \dfrac{N \times O_{11}}{F_1 \times F_2} & \text{if } O_{11} < \dfrac{F_1 \times F_2}{N \times e} \\ O_{11} \times \log_2 \dfrac{N \times O_{11}}{F_1 \times F_2} & \text{if } O_{11} \geq \dfrac{F_1 \times F_2}{N \times e} \end{cases}$$

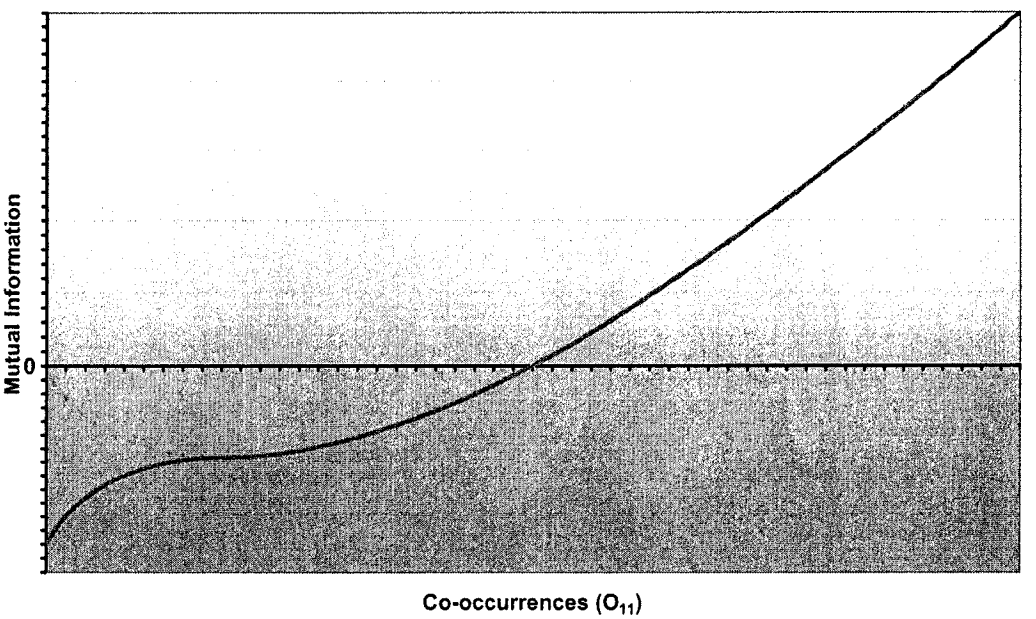

Figure 10: Typical Sketch of Modified Weighted Mutual Information

From this point on, this new definition of Mutual Information is referred to as *WMI* in our work. In Chapter 5, the empirical results of using mutual information, Chi-square and Yates' chi-square will be compared.

3.5.2.4 Example

The following example will illustrate how the similarity between a pair of candidates is computed using different similarity measures. Table 8 shows a contingency table containing the co-occurrence information for two candidate answers, *afghanistan* and *india*, of TREC-2007 Question 269.2, "Pakistan earthquakes of October 2005" with the target "What countries were affected by this earthquake?"

	afghanistan	¬ afghanistan	Total
india	85	397	**482**
¬ india	90	3379	**3469**
Total	**175**	**3776**	**3951**

Table 8: 2×2 Contingency Table for *afghanistan* and *india*

Based on the contingency table, the similarity between *afghanistan* and *india*, using Chi-square, is computed as:

$$\begin{aligned}
\chi^2(\text{afghanistan, india}) &= \frac{N\left(O_{11}O_{22} - O_{12}O_{21}\right)^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})} \\
&= \frac{3951\left(85 \times 3379 - 90 \times 397\right)^2}{(85+90)(85+397)(90+3379)(397+3379)} \\
&= 225.207
\end{aligned}$$

Similarly, we can define the similarity of the two candidates based on Yates' chi-square statistic:

$$\chi^2_{yates}(\text{afghanistan, india}) = \frac{N\left(|O_{11}O_{22} - O_{12}O_{21}| - N/2\right)^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})}$$

$$= \frac{3951\left(|85 \times 3379 - 90 \times 397| - 3951/2\right)^2}{(85+90)(85+397)(90+3379)(397+3379)}$$

$$= 222.619$$

Yates' score is slightly lower than the Chi-square score. The reason is that, as mentioned in Section 3.5.2.2, Yates' statistic tries to prevent overestimation of statistical significance. The WMI value is also computed as:

$$WMI(\text{afghanistan, india}) = O_{11} \times \log_2 \frac{N \times O_{11}}{(O_{11}+O_{12})\,(O_{11}+O_{21})}$$

$$= 85 \times \log_2 \frac{3591 \times 85}{(85+90)\,(85+397)}$$

$$= 169.429$$

3.5.3 Pinpointing the Right Cluster

After the clusters have been created, the main concern is how to select the right one. For this purpose, before clustering, the target and the question keywords are extracted, stemmed, and added to our candidate terms. Their responsibility is to *spy* on the candidate terms.

This strategy follows our hypothesis that the answers to a *List* question tend to

co-occur with one another and with the target and question keywords as well.

These spies are treated exactly like candidate terms; hence their co-occurrences with candidate terms and also other spies are computed, their similarities are calculated and finally they are clustered along with the candidate terms. These *spies* are used to pinpoint the cluster that is more likely to be the correct cluster; the cluster with the highest number of spies is selected and after removing the spies from the cluster, its members are returned as the final candidate answers. If the number of spies are equal for two or more clusters, then the largest cluster in terms of the number of members is returned.

3.5.3.1 Example

Here, through an example, we will illustrate the process of pinpointing the right cluster. The example shows how our approach uses spies to select the proper cluster on question 269.2 of TREC-2007. There are 33 candidate answers and 5 spies which are grouped into three clusters.

Target: "Pakistan earthquakes of October 2005"
Question: "What countries were affected by this earthquake?"
Keywords: Pakistan, earthquakes, October, 2005, affected
Spies: pakistan, earthquak, octob, 2005, affect

Clusters:

- **Cluster 1**: [hong kong, thailand, israel, united kingdom, spain, bangladesh, japan, indonesia, guatemala, germany, haiti, cambodia, nepal, china, congo, sweden, iran, north korea, mexico, vietnam, belgium, sri lanka, lebanon, iraq, sudan, russia, australia, mali, turkey]

- **Cluster 2**: [oman]
- **Cluster 3**: [*pakistan*, *2005*, afghanistan, *octob*, u.s, india, *affect*, *earthquak*]

Here, *Cluster 3* is selected as the correct cluster because it contains 5 spies, while *Cluster 1* and *Cluster 2* contain no spy.

The final candidate answers are therefore: [afghanistan, u.s, india]. Knowing the expected answers are [afghanistan, pakistan, india], we can now evaluate our answers. Among three candidates in *Cluster 3*, two of them are correct answers (afghanistan & india), which results in a precision of 0.667. Two of the three possible answers are returned; so recall is 0.667 and finally, the F-score takes the same value as precision and recall.

Recall [10] $= \frac{2}{3} = 0.667$
Precision [11] $= \frac{2}{3} = 0.667$
F-score [12] $= \frac{2}{3} = 0.667$

3.5.4 Cumulative Classification

Sections 3.5.1, 3.5.2, and 3.5.3 showed how a clustering method is used to group candidates and how the right cluster is pinpointed. As mentioned in Section 3.5, we also experimented with a different method: *Cumulative Classification*. In this method, candidate terms are simply classified into two classes: *Relevant* and *Irrelevant*. The classification is based on the same co-occurrence information used in clustering.

The idea is that candidate terms that generally co-occur more often with other candidate answers, hence with a relatively higher similarity value, are more likely

[10] Number of correct instances over number of expected instances, see 5.1 for more details
[11] Number of correct instances over number of final candidates, see 5.1 for more details
[12] The harmonic mean of precision and recall, see 5.1 for more details

to be the answer. Therefore, we use the sum of the similarities of each term to other terms as an indicator of how often each term co-occurs with all other terms. Candidates with a cumulative similarity greater than a certain threshold are labeled as *Relevant* and the rest are *Irrelevant.*

Table 9 illustrates how the cumulative similarity of each term to other terms is computed.

	term_1	term_2	...	term_n
term_1	0	$Sim_{2,1}$	...	$Sim_{n,1}$
term_2	$Sim_{1,2}$	0	...	$Sim_{n,2}$
⋮	⋮	⋮	0	⋮
term_n	$Sim_{1,n}$	$Sim_{2,n}$	...	0
Score	$\sum_{1 \le i \le n} Sim_{1,i}$	$\sum_{1 \le i \le n} Sim_{2,i}$	...	$\sum_{1 \le i \le n} Sim_{n,i}$

Table 9: Cumulative Similarity Table

Since the number of candidate answers and also the number of documents in the domain collection are different for each question, a fixed-value threshold is not appropriate. Instead, we exploit a variable value for the threshold. This value is related to candidate terms with the highest cumulative similarity. In this method, a certain fraction of the average of cumulative similarity of the top K candidates is set as the threshold. The following shows how this threshold is computed.

$$\textbf{Threshold} = F \times \frac{1}{K} \times \sum_{i=1}^{K} CumulativeSim(term_i)$$

where :

F : Fraction of the average of cumulative similarity $(0 < F \leq 1)$

K : Number of the top candidates used $(K \geq 1)$

Setting the threshold using this formula guarantees that at least K candidates will be returned if there are more than K initial candidates. Otherwise, no candidates are removed from the candidate list and all candidates are returned.

The steps to classify the candidate answers into *Relevant* and *Irrelevant* are as follows:

1. For each term in the candidate list, compute the sum of its similarities to all other candidate terms;
2. Sort the candidate terms based on their cumulative similarities;
3. Compute the threshold based on the above formula;
4. Retain only candidates whose cumulative scores are equal to or higher than the threshold;
5. Return the candidate terms as the final candidate terms.

We can use all three similarity measures that we defined in Section 3.5.2. In Chapter 5 the empirical results of Cumulative Classification using all three similarities will be shown.

3.6 Answer Projection

For the TREC competition, all instances of the answer to a *List* question returned by the system is required to be supported by a document from the corpora (i.e. AQUAINT-2 and BLOG06). Otherwise, the instance is judged as *Not Supported* by TREC and hence is not considered as a correct instance. The process of supporting an instance of an answer by a document is called *Answer Projection.*

We use a simple approach to answer projection. We search in the corpora using the Lucene query and the term to be supported. The top scored document returned by Lucene is taken as the supporting document for that term. The steps to find a supporting document are described below:

1. A query is generated to search in the corpora using Lucene:

$$\textbf{query} = \textit{LuceneQuery}\ \textbf{AND}\ \text{“}\textit{Term}\text{”}$$

2. The AQUAINT-2 corpus is searched using the query;

3. If there is at least one document satisfying the query, the top scored document is returned;

4. Otherwise, the BLOG06 corpus is searched and the top scored document from BLOG06 is returned;

5. If there is no document, neither on AQUAINT-2 nor on BLOG06, satisfying the query, the term is removed from the candidate list.

We use the BLOG06 corpus only when the system could not find a supporting document from AQUAINT-2 because of the noisy and unstructured text in BLOG06. The ungrammatical and sometimes non English text, HTML tags, advertisements, spam, etc. in the BLOG06 documents significantly reduces the quality of the index.

On the other hand, we did not have access to the entire corpus at the development phase, instead, we had 50 documents for each target (shared by all questions in the question series), which were provided by NIST. These documents are not guaranteed to contain all or any of the correct answers. Therefore, we decided to use AQUAINT-2 as the main corpus and use BLOG06 as an auxiliary corpus for answer projection.

If the module fails to support a term with a document, it drops the term from the candidate list because chances are that the term was extracted from a Web document and the term co-occurs with other candidates only in the documents retrieved from the Web. Besides, for TREC unsupported answers are as bad as wrong answers.

In this chapter, we have described how we answer *List* questions. This approach will be evaluated in Chapter 5, but for the moment we will discuss how we handle *Other* questions in the next chapter.

Chapter 4

Answering Other Questions

The answer to an *Other* question is meant to be interesting information about the target that is not covered by the preceding questions in the series. This type of question is to be interpreted as "Tell me other interesting things about this target; I don't know enough to ask directly" [Dang *et al.*, 2007].

Fundamentally, our approach to answering *Other* questions is based on the hypothesis that interesting sentences can be identified by:

1. **Target-Specific Interest Markers:** important and descriptive terms related to the target. For instance, for the target "Titanic", *White Star Line* (the owner company) and *15 April, 1912* (the date of sinking) are two target-specific interest markers.

2. **Universal Interest Markers:** terms that are intrinsically important regardless of the target. For instance, the word *first* (e.g. "first man on the moon"), and *died* (e.g. "thousands of people *died*") usually bear important meanings regardless of the target.

To identify these interest marking terms, we did not use the documents of the corpus (AQUAINT for TREC-2006 and prior and AQUAINT-2/BLOG for TREC-2007),

where the answer should be found. This is why the corpus consists of newspaper articles that do not necessarily present the highlights of a target. An article presents detailed facts regarding the target but not an overview. We believe that a rich resource to find interesting facts related to many targets is an encyclopedia. Many target types are described and the content of each article is a short summary that highlights the most interesting facts, which is precisely what we are looking for. To find target-specific interest markers, we therefore used the Wikipedia online encyclopedia[1]. Wikipedia contains more than 2 million encyclopedic entries in English for various topics ranging from famous persons, to current events, to scientific information. The chances of finding an article on the topic of an *Other* question is therefore high, and we can extract potentially interesting terms from these entries without much noise. These terms are then searched in the corpora to extract interesting sentences which are then re-ranked using universal interest marking terms. Sentences with the highest scores are finally presented as interesting nuggets.

In the following sections, we will discuss the process of extracting answers to *Other* questions in greater detail.

4.1 Finding the Wikipedia Article

The first stage to answering an *Other* question is to find the proper Wikipedia article. This process is shown in Figure 11.

First, we generate a Google query using the target of the question. The target is first parsed, stop words are removed, and consecutive capitalized words are quoted together as a single term. Because verbs in the targets are usually in the present tense (e.g. "Russian submarine Kursk sinks", "France wins World Cup in soccer") while in the Wikipedia articles, verbs are usually in the past tense (e.g. "It *sank* in the Barents Sea", "The tournament was *won* by France"), they are not included in

[1] `http://en.wikipedia.org`

the query. The remaining words and quoted terms are then ANDed and sent to the Google API to search the Wikipedia sub-domain.

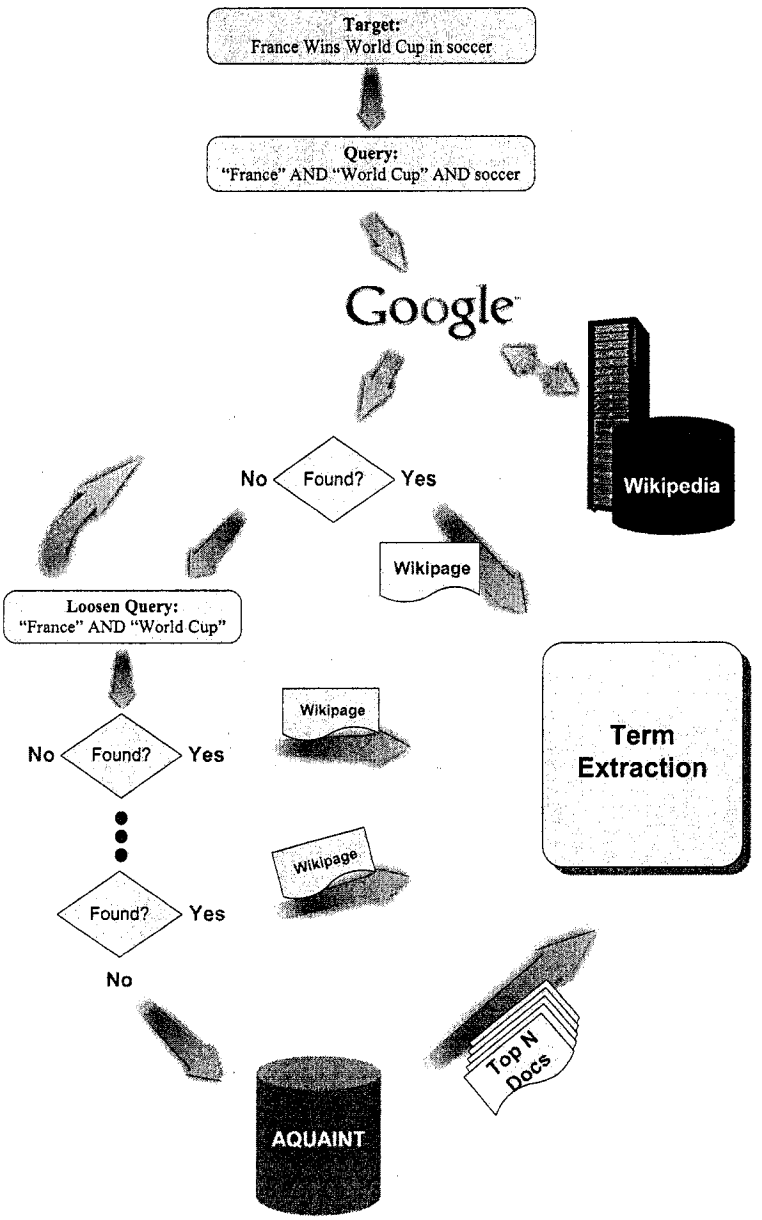

Figure 11: Finding the Relevant Wikipedia Article for the Target "France wins World Cup in soccer"

If several Wikipedia articles satisfy the query, the first one is taken. However, if no Wikipedia article satisfies the query, then we try to relax the query. Considering

that quoted terms often have a non-compositional meaning, we keep them as is but OR single words. If this is not sufficient, then we gradually remove the last single word from the end. Finally, if still no Wikipedia article is found, then we simply drop Wikipedia and take the top N documents (between 3 to 10 documents depending on whether the number of keywords is large enough) of the corpus collection using the original query.

4.2 Extracting Target-Specific Interest Markers

After the Wikipedia article or top N corpus documents are retrieved, interest-marking terms are extracted from the page (or pages). Because the Wikipedia entries consist of rather short documents (with an average of 400 words per article[2]), we only consider named entities as interesting terms. These are extracted using the GATE NE tagger[3]. If the number of terms in a specific semantic class (Date, Location, Person and Organization) is abnormally high (20 terms for each semantic class), then we assume that the page does not present a balanced overview of the highlights, but presents a specific point-of-view about the target and will therefore be biased towards that point-of-view. For example, if a Wikipedia article on an event (e.g. the *1998 World Cup*) contains a large number of person names, then we assume that the article is biased towards describing the people involved (e.g. the soccer players) as opposed to *Other* interesting information. To avoid this, we set a threshold on the number of terms for each semantic entity class that we keep. After removing terms occurring only once, the N most frequent terms are kept (in our case, 20).

We approximate co-reference resolution, by using word overlap. For example, in answering the target "Port Arthur Massacre", we may find in the Wikipedia article the terms "Port Arthur" and "Port Arthur Massacre". To consider both terms as

[2] `http://en.wikipedia.org/wiki/Wikipedia:Words_per_article`
[3] `http://gate.ac.uk`

single concepts, we separate longer terms that overlap with shorter ones into sub-terms (e.g. "Port Arthur" and "Massacre").

4.3 Finding Interesting Sentences

Once we have a set of interesting terms for each target, we search for the N most relevant documents in the corpus. These documents are retrieved by the *Lucene*[4] search engine using the same query generated for the target as in the Wikipedia search (see Section 4.1). If the appropriate Wikipedia article has been found then we also use a secondary query from the title of the Wikipedia article in order to get more documents related to the target. This secondary query is ORed to the Google query. For example, for the target "France wins World Cup in soccer" we have:

Google Query = "France" AND "World Cup" AND soccer

Title of Wikipedia Article = 1998 FIFA Word Cup

and we generate:

Lucene Query = (1998 AND "FIFA World Cup")

OR

("France" AND "World Cup" AND soccer)

If too many documents are returned through the Lucene search with this new query, then we add content words from the previous questions of that target (i.e.

[4] `http://lucene.apache.org`

factoid and *List* question) to the query with less priority in order to focus the search.

Within the documents chosen as the domain, the frequency of each interest marking term is then computed. For each term, we compute a weight as the logarithm of its frequency.

$$\mathbf{Weight(T_i)} = \log(Frequency(T_i))$$

This weight represents how interesting a term is as a function of its frequency in the related documents. The less frequent a term, the less interesting it is considered.

4.4 Ranking Interesting Sentences

All sentences from the domain documents are then scored according to how interesting they are. The score of each sentence is computed as the sum of the weight of the interesting terms it contains.

$$\mathbf{Score(S_i)} = \sum_{j=1}^{n} Weight(T_j) \quad | \quad T_j \in S_i \qquad \forall 1 \leq j \leq n$$

In order to increase the precision, we try to remove any extra characters on both ends of the sentence which do not contain much interesting content. Two kinds of information are removed: the source of the news at the beginning of sentences (e.g. WASHINGTON (AP) - ...) and markers of reported speech at the end of sentences (e.g. ..., *local newspaper Daily Telegraph reported*).

After scoring the sentences and throwing away those with a score of zero (i.e. no interesting terms in them), we try to remove paraphrases. In order not to remove false

paraphrases, we play it conservatively, and only remove lexically similar sentences. Either the sentences are almost equivalent to each other at the string level or they share similar words but not the same syntax. To compare sentences, we have used the *SecondString* package[5], an open-source Java-based package of approximate string-matching techniques [Cohen *et al.*, 2003]. For removing the first kind of similarity, the Jaccard algorithm was used and for the second kind, the Jensen-Shannon was used. Both algorithms compute similarity based on token distance.

Since long sentences decrease the precision (see Section 6.1), if the length of a sentence is more than 100 non-white-space characters, its score is decreased by a factor (20% of its score). Analogously, for sentences with length less than 100 non-white-space characters, the score is boosted.

4.5 Universal Interest Markers

Once the sentences are ranked based on the target-specific interesting terms, we boost the score of sentences that contain terms that generally mark interesting information regardless of the topic. Such markers were determined empirically by analyzing the previous TREC data.

We have defined three types of universal interest markers: superlatives, numbers and interest marking keywords.

4.5.1 Superlatives

We hypothesized that an interesting sentence would typically contain superlative adjectives and adverbs. People are interested in knowing about the *best*, the *first*, the *most wonderful*, and find normal or average facts uninteresting.

To verify this hypothesis, we computed the percentage of superlatives in *vital*, *okay*

[5] `http://secondstring.sourceforge.net`

Sentence Type	Corpus Size	Superlatives	Numerals
Vital	49,102 words	0.52 %	2.46 %
Okay	56,729 words	0.44 %	2.26 %
Uninteresting	2,002,525 words	0.26 %	1.68 %

Table 10: Ratio of Superlatives and Numerals in Each Type of Sentence

and *uninteresting* sentences from the 2004 data. For *vital* and *okay* sentences, we used the answers submitted by the TREC-2004 participants and judged by NIST assessors. For *uninteresting* sentences, we extracted sentences from the top 50 AQUAINT documents from the domain documents which do not contain *vital* or *okay* nuggets. The results, illustrated in Table 10, clearly show an increase in the use of superlatives in *vital* compared to *okay* and *uninteresting* sentences. When re-ranking nuggets, the score of sentences that contain superlatives are therefore given a bonus. Experimentally, we set this bonus to be 20% of the original sentence score per superlative it contains.

4.5.2 Numerals

We hypothesized that sentences containing numbers probably contain interesting information as well. For example, "Bollywood produces 800 to 900 films a year" or "Akira Kurosawa died at age 88". To verify this, we also compared the percentage of numerals in *vital*, *okay* and *uninteresting* sentences on the same corpus. The results, shown in Table 10, again indicate that numerals are used more often in *vital* and *okay* sentences as opposed to *uninteresting* sentences. To account for this, the score of sentences containing numerals is boosted by 20% for each numeral they contain. However, numerals that are part of a date expression such as *Sep 27, 2000* are excluded because we already considered them interesting from the Wikipedia entry.

4.5.3 Interest Marking Keywords

In addition to superlative and numerals, we also wondered if for specific target types, different terms are typically regarded as interesting. For example, information on someone's birth or death, the founders of an organization, the establishment of an entity ... would all be considered interesting. These terms do not fit any specific grammatical category, but just happen to be more frequent in interesting nuggets. This is similar to the work of [Ahn *et al.*, 2005b]. To identify these terms, we analyzed the data of the TREC-2004 *Other* questions. The data set consisted of:

1. The factoid and *List* questions of each target, because they mostly ask for interesting information.

2. The prototypical *vital* and *okay* answers to *Other* questions given by the TREC assessors[6].

3. The participants answers to *Other* questions which are judged *vital* or *okay* by the TREC assessors.

All these were stop-word removed and stemmed, then the frequency of each word was computed. The score of a keyword was computed as:

$$\mathbf{Score(K_i)} = Freq(K_i) \times Distrib(K_i)^2$$

where $Freq(K_i)$ is the frequency of a keyword and $Distrib(K_i)$ is the number of targets whose source contains the keyword. The intuition behind this scoring function is to favor keywords that are referred to in a high number of targets as opposed to keywords that appear frequently, but only for a few targets. Hence, a keyword K_i that

[6] Available at `http://trec.nist.gov/data/qa/2004_qadata/04.other_answers.txt`

occurs in a large number of targets is considered more important than a keyword K_j occurring more often (i.e. $Freq(K_j) > Freq(K_i)$) but in a smaller number of targets (i.e. $Distrib(K_j) < Distrib(K_i)$).

To identify terms that appear more often in interesting sentences as opposed to *uninteresting* sentences, we also built such a list of terms from the *uninteresting* answers submitted by the participants to the TREC-2004 QA track (i.e. answers not considered as either *vital* or *okay*). Then, we computed the ratio of their scores as:

$$\mathbf{ScoreRatio(K_i)} = \frac{Score_{int}(K_i)}{Score_{uni}(K_i)}$$

Where $Score_{int}(K_i)$ refers to the score of K_i in the *vital* and *okay* sentences and $Score_{uni}(K_i)$ refers to the score of K_i in the *uninteresting* sentences.

Table 11 shows the 15 top-ranking keywords that were extracted from each target type and from all combined using the TREC-2004 data. As the table shows, the ranking of *most* and *first* verifies the importance of boosting superlatives.

In order to make a specific list of interesting keywords for each target type, we did the same work for each category of questions (PERSON, ORGANIZATION and THING), which are also shown in Table 11.

Each sublist can be consulted according to the type of our target. For example, if the target is a PERSON, then we boost terms appearing in the PERSON sublist. However, instead of using the interest marking terms in this way, we re-constructed a global list from the concatenation of the top keywords of each sublist in order not to use a target-type classifier that would introduce errors in classification and hence in sentence re-ranking.

On the other hand, we do not make use of the initial all-target-type list shown in Table 11. There are two advantages to using the re-constructed list. First, it allows us to make sure that each target type equally contributes in creating the global list as

Rank	All Types	THING	PERSON	ORGANIZATION
1	found	kind	born	chang
2	die	fall	servic	publish
3	associ	public	serv	establish
4	life	found	become	first
5	begin	countri	film	leader
6	publish	offici	general	associ
7	first	field	old	larg
8	public	program	movi	found
9	servic	develop	chairman	releas
10	group	director	place	project
11	death	begin	receiv	group
12	see	discov	begin	lead
13	countri	particl	win	organ
14	old	power	life	begin
15	most	figur	intern	provid

Table 11: Interest-marking Keywords in all Target Types and for Each Type of Target in TREC-2004

there are equal number of questions for those target types in TREC QA. Therefore, high score terms of one target-type can not take over most of the list. In addition, a re-constructed global list prevents us from considering terms that do not have a particularly high score in any one sublist, but occurs in every sublist with an average score; therefore having a high overall score, but not a high score in any one sublist (e.g. "see").

Sentences containing terms that are in the final re-constructed list are given a bonus of 20% per term, except if the terms also appear in the previous questions of the target as they might be indicators of the same information asked in the previous questions of the question series.

This chapter presented how we answer *Other* questions. The approaches to answering *List* and Other questions will be evaluated in Chapter 5 & 6.

Chapter 5

List Questions: Evaluation and Results

In this chapter, our approach to answering *List* questions will be evaluated. Section 5.1 will explain the official evaluation metric used by NIST to assess answers to *List* questions. The experimental setup will be described in Section 5.2. In Section 5.3, each component of our system (i.e. Answering Type Recognition, Document Retrieval, Candidate Answer Extraction, Clustering, and Pinpointing the Right Cluster) will be evaluated individually. Section 5.4 will investigate using an external candidate resource in the Candidate Extraction method. The performance of our system in answering *List* questions will be illustrated for each question type in Section 5.5. Finally, the official results of our method, acquired from participating at the TREC-2007 QA track, will be discussed in Section 5.6.

5.1 Evaluation Measure

An answer to a *List* question is evaluated using precision and recall based on the complete list of known distinct globally-correct answers provided by the TREC assessors [Dang *et al.*, 2007]. For each question, instance precision is the percentage

of instances returned that are correct and instance recall is the percentage of the expected correct instances that are returned.

$$\textbf{\textit{IPrecision}} = \frac{\text{number of correct answers}}{\text{total number of returned answers}}$$

$$\textbf{\textit{IRecall}} = \frac{\text{number of correct answers}}{\text{total number of expected answers}}$$

To evaluate the quality of an answer, the instance F-score (also known as F-measure) which is a weighted harmonic mean of precision and recall is used.

$$\textbf{\textit{IF-score}} = \frac{(\beta^2 + 1) \times \mathit{IPrecision} \times \mathit{IRecall}}{\beta^2 \times \mathit{IPrecision} + \mathit{IRecall}}$$

Here, β is the relative weight of recall versus precision. The instance F-score used for evaluation *List* questions gives equal weight to precision and recall. So $\beta = 1$ is used:

$$\textbf{\textit{IF-score}}_{\beta=1} = \frac{2 \times \mathit{IPrecision} \times \mathit{IRecall}}{\mathit{IPrecision} + \mathit{IRecall}}$$

The F-score of an entire run (a set of question series) is the average instance F-score over all the questions in the run, which is used to evaluate a run. Therefore:

$$\textbf{\textit{F-score}} = \frac{\sum \mathit{IF\text{-}score}}{N}$$

Where N refers to the number of questions in the run. Similarity, the precision and recall of a run is computed as:

$$\textbf{\textit{Precision}} = \frac{\sum IPrecision}{N} \qquad \textbf{\textit{Recall}} = \frac{\sum IRecall}{N}$$

5.2 Experimental Setup

We have used the data from TREC to evaluate our approach as well as each component of our system. 237 *List* questions of TREC-2004 to 2006 were used for training our system (i.e. developing the lexical and syntactic patterns, setting various threshold values, etc.) and 85 *List* questions of TREC-2007 were used for testing our approach.

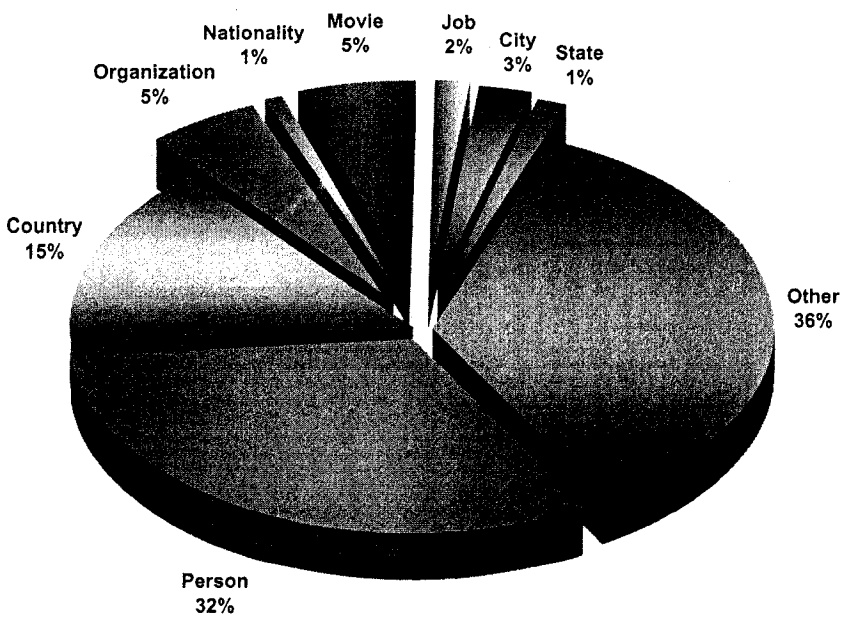

Figure 12: Percentage of Each Question Type in the Training Set

Figure 12 shows the distribution of the question types in the training set. OTHER, PERSON and COUNTRY account for the largest portions of the questions.

The percentage of each question type in the test set is shown in Figure 13. In this set, there is no NATIONALITY or CITY questions.

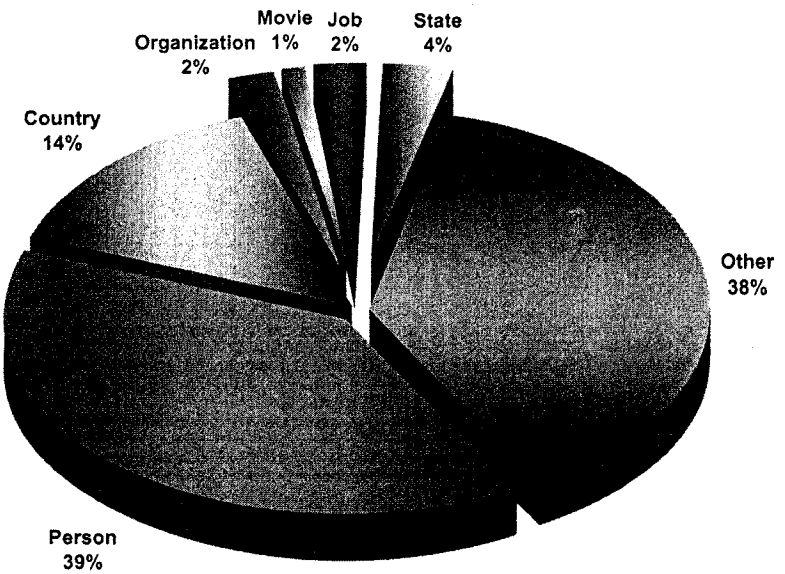

Figure 13: Percentage of Each Question Type in the Test Set

5.3 Component Evaluation

In this section, the performance of each component, namely Answer Type Recognition, Document Retrieval, Candidate Answer Extraction, Clustering and Pinpointing the Right Cluster, is evaluated individually. Different measures have been used to evaluate these components.

5.3.1 Evaluation of Answer Type Recognition

Table 12 shows the number of correctly classified and incorrectly classified questions in both the training and the test sets using the Answer Type Recognition techniques described in Section 3.1. Accuracy which is defined as the number of correctly classified questions over the total number of questions indicates that our Answer Type Recognition module works well in classifying the questions into the defined classes.

Set	Questions	Incorrect	Correct	Accuracy
TREC-2004 to 2006 (Training)	238	4	234	0.98
TREC-2007 (Test)	85	5	80	0.94

Table 12: Accuracy of the Answer Type Recognition Module

As Table 12 shows, the accuracy of our Answer Type Recognition method is 0.98 and 0.94 in the training set and the test set respectively.

Table 13 shows how much each tier contributes to the accuracy of the method in the training and the test sets.

Tier	Training	Test
LEXICAL PATTERNS	0.43	0.31
LEXICAL + SYNTACTIC PATTERNS	0.75	0.64
LEXICAL + SYNTACTIC PATTERNS + SUBTYPE RESOLVING	0.98	0.94

Table 13: Contribution of Each Tier in Answer Type Recognition in the Training and Test sets

5.3.2 Evaluation of Document Retrieval

The Document Retrieval module is evaluated in this section. Among the two document collections created (i.e. *source* and *domain*), evaluating the *domain* collection is not easy because it is used to extract co-occurrence information and evaluating the documents based on co-occurrence information is not practicable. Therefore, we only evaluate the *source* documents and a feasible way to do so is to evaluate the retrieved documents based on the proportion of answers they contain. The list of correct answers to the TREC questions are provided by NIST. Using this list, the retrieved documents from the Web and the corpora are evaluated in different configurations based on recall (for the definition of recall refer to Section 5.1).

Figure 14 illustrates the number of distinct answer instances found in different number of documents retrieved from the corpus and the Web and also the number of distinct answers in their combination in the training set (TREC-2004 to 2006). The total number of instances to be found is 2541. As the figure shows, the web documents are more informative than the corpus documents because they are generally larger in size. However, even though the web documents are more than four times bigger in size (the average size for Web documents is 17KB and for corpus documents is 4KB), they do not contain four times as many instances as the corpus documents do. Surprisingly, in 40 documents, the number of instances in the Web documents and in the corpus documents are very close. Obviously, the distinct instances found in the union of the Web documents and the corpus documents are fewer than the sum of instances found in them separately because some instances are shared between them.

Although the web curve grows faster at the beginning, it flattens out at 7 or 8 documents (the knee of the curve), which suggests that, assuming the precision decreases linearly with the increase in the size of the documents, the best performance (based on the F-score) of the Web documents is achieved when we use 7 or 8 documents from the Web. On the other hand, it seems that the corpus curve keeps growing nearly

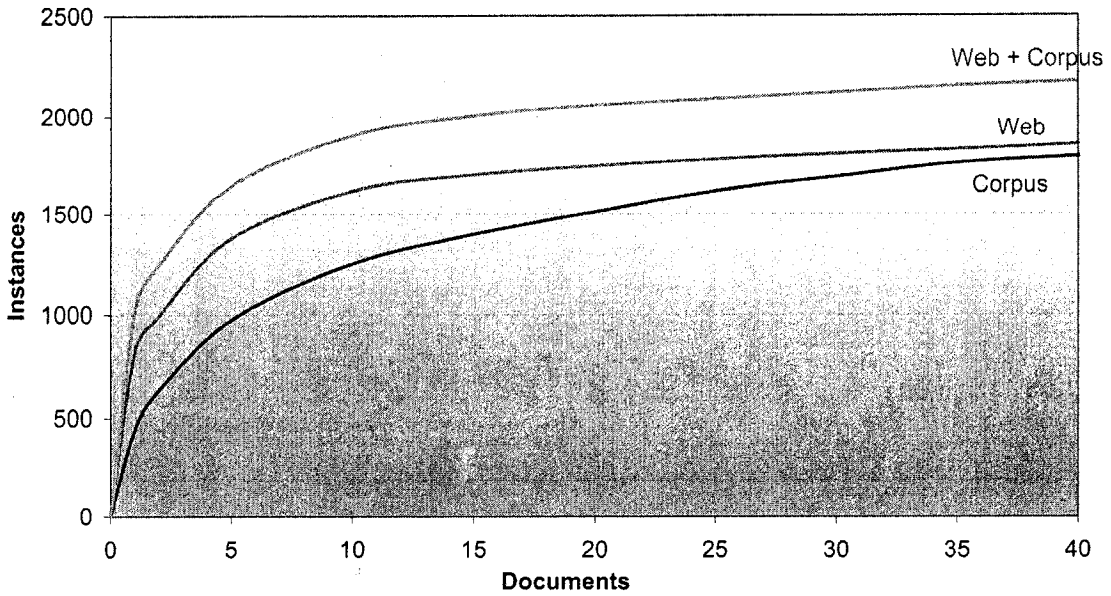

Figure 14: Answer Instances vs Number of Documents

constantly.

Figure 15 shows the correlation between the size of the documents and the number of distinct answers found. At any given size, the corpus text contains more instances than the Web text, suggesting that the number of corpus documents used for the source document collection should be higher than the number of Web documents. The figure also shows the combination of the corpus and the Web performs better than each individual corpus at any given size.

Again, assuming that there is a negative linear correlation between the precision and the size of documents, the knees of the curves suggest the approximate maximum values for the total size of documents to have the optimal F-score. For example, the curve for the combination bends where the size is about 230KB. From this point onward, much fewer instances are found for the same change in size. The similar values for the Web and the corpus curves are about 150KB. However, we previously

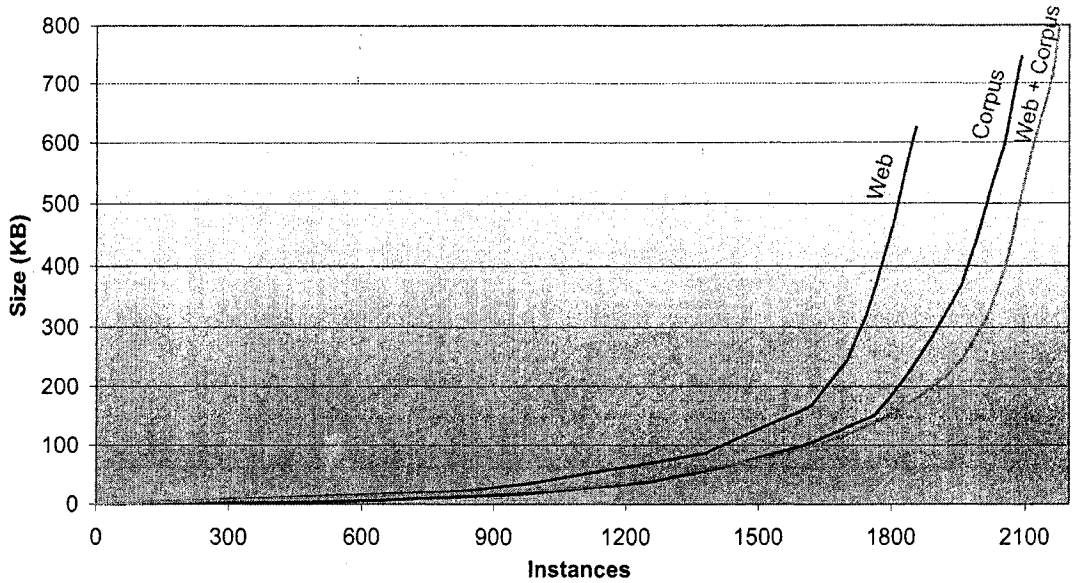

Figure 15: Answer Instances vs Size of Documents

stated that, using Figure 14, the optimal value for the number of web documents is 7 or 8 documents ($8 \times 17 = 136 \approx 150KB$), leaving roughly 80KB for the corpus documents which is about 20 ($80KB \div 4KB$) documents. However, since the F-score is a harmonic mean of precision and recall and the harmonic mean tends strongly toward the lowest element, the final choice for the number of Web documents and the number of corpus documents also depends on the overall precision and recall of our system since we may be willing to trade the precision for recall if the precision is very low or the other way around to increase the overall F-score.

In Section 3.2.2.1, we mentioned that the created queries from the target and the question are relaxed until the number of hits for that query exceeds a certain threshold (two different thresholds for the corpus and the Web). Now, we would like to approximate the proper values for these thresholds. Figure 16 shows the results when using three values for the corpus query based on the number of instances

found for different document sizes. The experiment is carried out on the training set (TREC-2004 to 2006) with 50, 100 and 200 for the minimum number of documents that satisfies the query in order not to relax the query anymore.

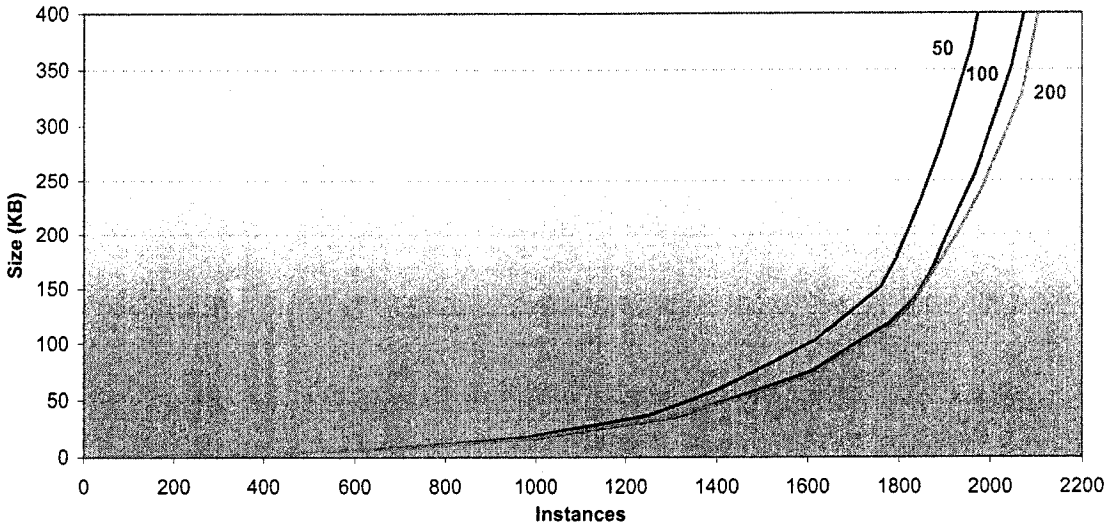

Figure 16: Evaluation of the Corpus Documents Retrieved with Different Thresholds for Query Refinement

As Figure 16 illustrates, using 50 for the threshold caused too strict queries. The difference between using 100 and 200 is not as significant as the difference between 50 and the other two. However, for the size we have concluded to use from the corpus for the source document collection (80KB), having 100 or 200 for the threshold does not make a notable difference, so we have used 100 as the value of this threshold in our final system.

A similar experiment is used to compare the performance of our Document Retrieval module in retrieving Web documents in the training set. Since the Web is a very big corpus compared to AQUAINT, we experimented with much bigger values for the threshold. We use 1000, 5000 and 20000 for this purpose. Figure 17 compares the number of instances found per KB having these three numbers as the threshold.

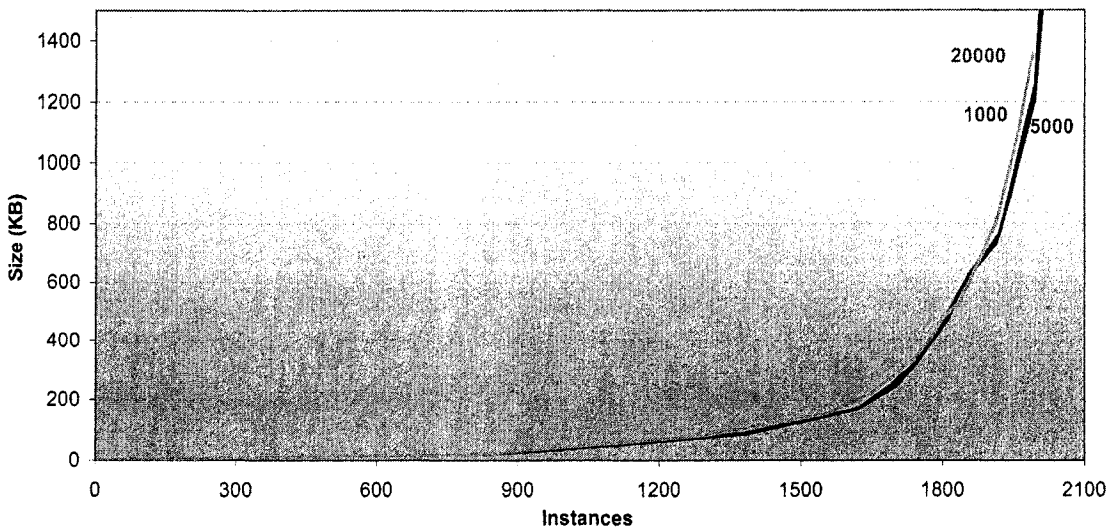

Figure 17: Evaluation of the Web Documents Retrieved with Different Thresholds for Query Refinement

In fact, the results are very much similar. However, using 5000 performs a bit better than the other two. Therefore, we have used 5000 for the threshold in our final system.

Having set the optimal values for these thresholds, we can define the performance of Document Retrieval. The performance of this module is defined as the number of instances appearing in the 8 most relevant Web documents and the 20 most relevant AQUAINT documents over the total number of expected instances for all questions. Using this configuration, the system finds 2027 instances out of 2541 instances; so the performance is 0.79 or, in other words, the Document Retrieval module is able to create a source collection that contains 79% of the correct instances in the training set.

5.3.3 Evaluation of Candidate Answer Extraction

In this section, the Candidate Answer Extraction module will be assessed. The evaluation can be based on the precision and recall of the initial candidate answers or based on the percentage of correct instances, within the source collection, that is extracted.

Using the estimated thresholds in Section 5.3.2 (taking 8 Web documents and 20 corpus documents for the source collection and using the thresholds acquired for the query refinement), candidate answers are extracted based on the techniques described in Section 3.3. These candidate answers are compared with the expected answers. Table 14 shows the number of correct instances, returned instances and expected instances for all the questions in the training set and for each question type as well. The table also shows the maximum number of instances that the Candidate Answer Extraction module can extract, which is the number of correct instances appearing in the source documents because the system in this point does not add any extra candidates.

Using the information in Table 14, we can define the overall recall of the extracted terms. It can be defined as the recall of a list consisting of all the initial candidate answers for all the questions (not average of instance recalls), which can be computed as the number of correct instances over the number of expected answers. Therefore the overall recall of this module is 0.492 ($1252 \div 2541$).

The reason why we are more interested in using overall precision and overall recall rather than precision and recall is that in overall precision and overall recall, all instances have equal weight while in precision and recall, each question is given equal weight and the weight is further divided into the answer instances of the question. Therefore, instances in questions with fewer answers have more impact than those in questions with more answers. The overall recall for each question type is illustrated in Figure 18.

	Correct	Returned	Expected	Maximum
Country	456	2361	549	467
Nationality	11	58	11	11
State	17	94	17	17
Movie	21	367	182	164
City	41	822	63	48
Person	499	12258	944	671
Organization	37	1594	91	81
Other	165	7558	666	553
Job	5	307	18	15
Total	1252	25419	2541	2027

Table 14: Evaluation of Candidate Answer Extraction for Each Answer Type

Not surprisingly, the recall of the answer types that use gazetteers (i.e. COUNTRY, CITY, NATIONALITY and STATE) is high. The recall of JOB, MOVIE AND OTHER is very low and requires more investigation because the recall in this module constitutes the upper bound for the recall that our system can eventually attain.

Overall precision can also be defined in a similar way. Therefore, overall precision can be computed as the number of correct instances over the number of returned answers. The overall precision of the Candidate Answer Extraction module, using the mentioned configurations, is 0.049 ($1252 \div 25419$). Although the overall precision is very low, this module focuses more on recall rather than on precision as the Candidate Selection module is supposed to filter less likely candidates out while trying not to lose any recall. Nevertheless, knowing the answer-based precision can be useful to compare the different techniques used in answer extraction for each answer type. Figure 19 illustrates the overall precision for each question type.

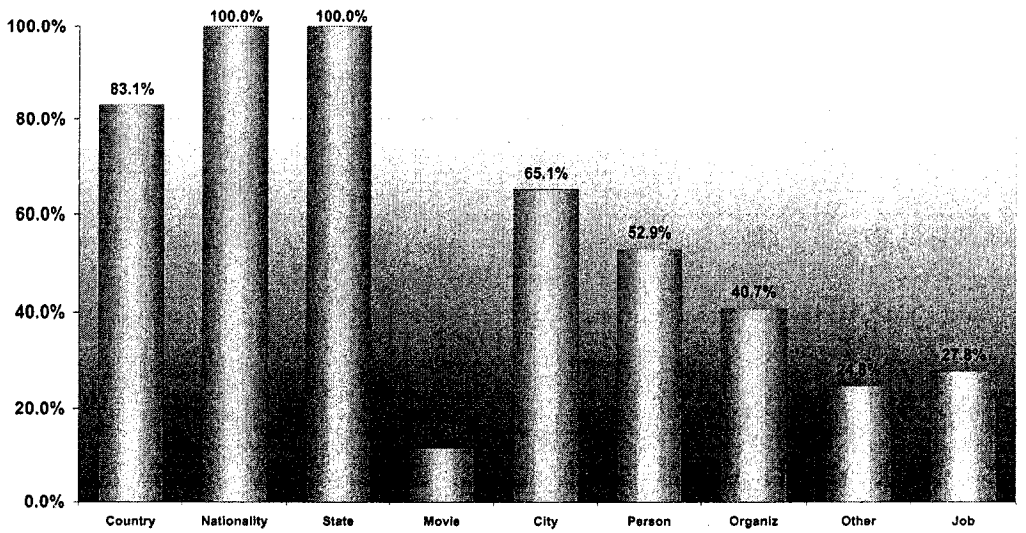

Figure 18: Overall Recall of Each Answer Type

As Figure 19 shows, the precision of COUNTRY, NATIONALITY and STATE is relatively high because these entities are extracted using gazetteers. However, the precision of CITY, although using a gazetteer, is not satisfying. Beside the fact that the city gazetteer contains a relatively high number of cities (approximately 8,000), many city names are homonymous (e.g. zero, david, george, mary, gore, clinton, van, eagle, rivers, parks, sale, ...), which results in significant noise. Although the overall precision of MOVIE, PERSON, ORGANIZATION, JOB and OTHER is low, the need to enhance the candidate answer extraction techniques used to extract them is to be judged based on the final precision of these types of questions as the Candidate Answer Selection module is supposed to improve the precision through clustering.

A more realistic method to evaluate the Candidate Answer Extraction module is to compute the percentage of correct instances, within the source documents, that are extracted. So, we define the performance of this module based on the number of correct instances over the maximum it can extract (i.e. total number of instances in the source documents). Using this definition, the performance of this module on the training set is 0.62 ($1252 \div 2027$). In other words, 62% of the correct instances

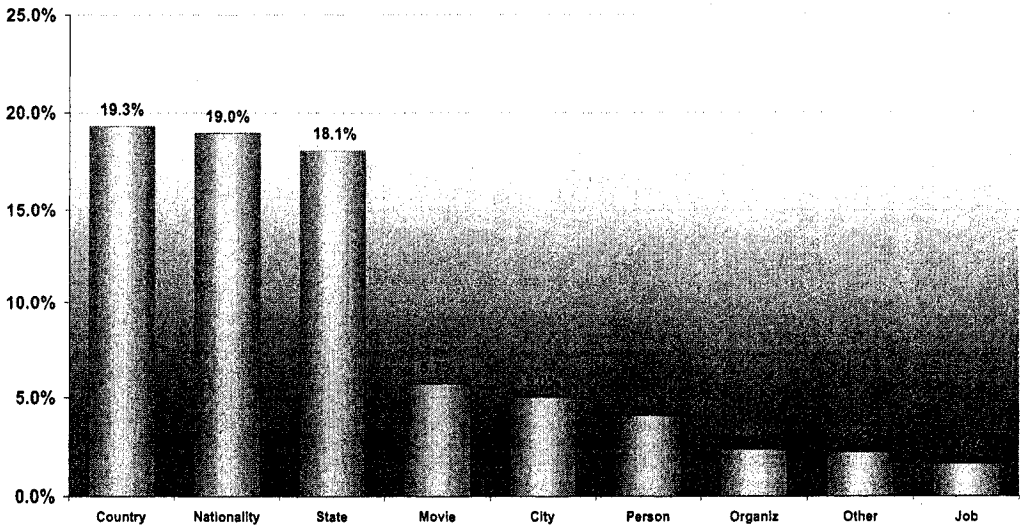

Figure 19: Overall Precision of Each Question Type

appearing in the source collection is extracted using Candidate Answer Extraction.

Figure 20 compares the performance of the module for each answer type. The performance of COUNTRY, NATIONALITY, STATE, CITY AND PERSON is satisfactory. On the other hand, ORGANIZATION and JOB, even though using NE taggers, miss respectively 54% and 66% of the correct instances that appear within the source documents. Although the performance of the module for OTHER is not very high, we do not expect much as there is no clue to extract such entities. Finally, the module is very weak in extracting MOVIE entities and needs more investigation.

In overall, the performance of the Candidate Answer Extraction module (i.e. 0.62) reveals that this module needs to be enhanced, especially in OTHER as it constitutes 36% of all the questions in the training set (see Figure 12) and any improvement in answering OTHER questions has a great effect on the final result of the system.

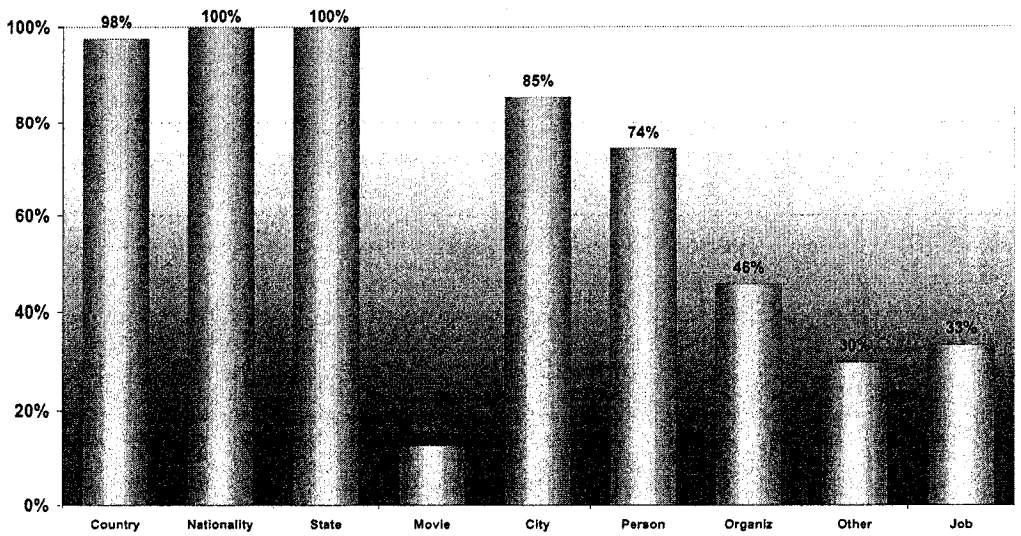

Figure 20: Performance of Candidate Answer Extraction

5.3.4 Evaluation of Clustering

To evaluate our clustering method, the input to the clustering method (i.e. the initial candidate list) and its output (i.e. the final candidate answers) as well as the maximum it can achieve using the clustering method are evaluated. Particularly, we are interested to see how much our method can improve the baseline and also how much capacity there is to enhance the clustering method.

- **Baseline:** Baseline represents the evaluation of the initial candidate list before clustering them.

- **System:** represents the evaluation of our clustering method. It actually consists of two sub-components: 1) Clustering and 2) Pinpointing the right cluster. In Section 5.3.5, we will assess the pinpointing method separately.

- **Theoretical Maximum:** The theoretical maximum is the best possible output of our clustering method using a certain initial candidate list. In other words,

	Set	Questions	Precision	Recall	F-score
Baseline	Training	237	0.071	0.490	0.109
System			**0.172**	**0.320**	**0.182**
Theoretical Max			1	0.490	0.568
Baseline	Test	85	0.075	0.448	0.110
System			**0.199**	**0.296**	**0.181**
Theoretical Max			1	0.448	0.547

Table 15: Results of the System Before and After Applying Our Approach.

in this case, all and only correct answers in the initial list are returned by the system. Therefore, the precision of the retrieved sublist is 1.0 and its recall is equal to the recall of the initial list (since our approach does not extract further candidates which are not in the initial list).

Table 15 illustrates the results obtained using this experimental setup in the training set (TREC-2004 to 2006) and the test set (TREC-2007). As it shows, our method is able to increase the F-score of the initial candidate list by 67% and 64% for the training set and the test set respectively.

The precision, recall and F-score of the system in Table 15 are computed using Mutual Information as the similarity measure and single linkage as the linkage metric. Figure 21 compares the F-score of each of the three similarity measures (i.e. Mutual Information, Chi-square and Yates' chi-square) with each of the three linkage metrics (i.e. single, average and complete) in the training set. In this bar chart, for each category the first bar represents the F-score of the system clustering the initial candidate list into two clusters and then pinpointing one. In case of Mutual Information, however, the clustering continues until there are only two clusters remaining or the highest similarity is less than zero (indicating dissociation rather than association), whichever occurs first. Therefore, this number is the minimum number of the final

clusters in case of Mutual Information. In a similar way, the second bar of each category represents the F-score of the system when clustering the initial candidate list into three (in case of Chi-square and Yates' chi-square) and at least three (in case of Mutual Information).

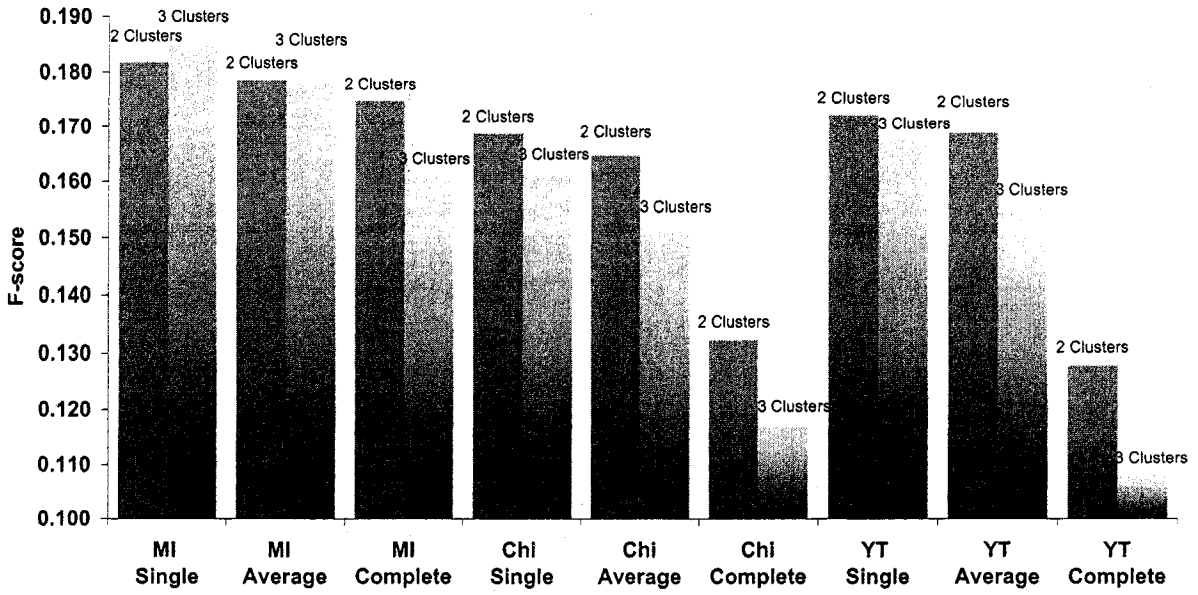

Figure 21: F-score of the System Using Different Linkage Metrics in the Training Set

As Figure 21 shows, clustering with single linkage performs slightly better than the average linkage and the average linkage is relatively much better than the complete linkage. In addition, Mutual Information is a more appropriate similarity measure than the other two. Yates' chi-square performs slightly better than Chi-square. Using two clusters results in a higher F-score than using three clusters. One reason could be the pinpointing method has more challenge to pinpoint the right cluster among three clusters, comparing to only two clusters.

Figure 22 shows the F-score of the system, on the training set, using all three similarity measures along with different number of clusters (i.e. 2, 3, 4 and 5 clusters) for each similarity measure. The single linkage is used in all clustering configurations.

The figure also illustrates the baseline (the F-score of the initial candidate list), the F-score of the candidates having filtered out infrequent terms, and the F-score of the *Cumulative Classification* method. As the figure shows, Cumulative Classification performs pretty well comparing to the clustering method.

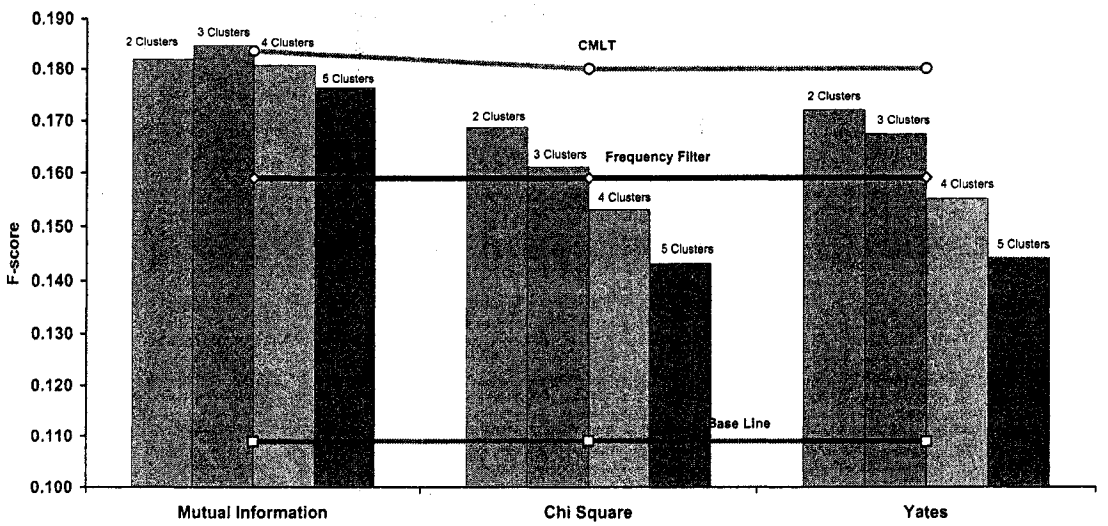

Figure 22: F-score of the System Using Different Similarity Measures in the Training Set

Similarly, Figure 23 compares the F-score when different similarity measures are used with different number of clusters in the test set.

We can define the performance of the clustering method as the F-score of the system over the theoretical maximum F-score the system can achieve. Therefore, the performance of the clustering method is 32% and 33% respectively.

5.3.5 Evaluation of Pinpointing

In this section, we would like to evaluate the performance of the Pinpointing method. Particularly, the quality of the pinpointed cluster is compared with the best cluster. Figure 24 compares the F-score of the system and the maximum F-score among the

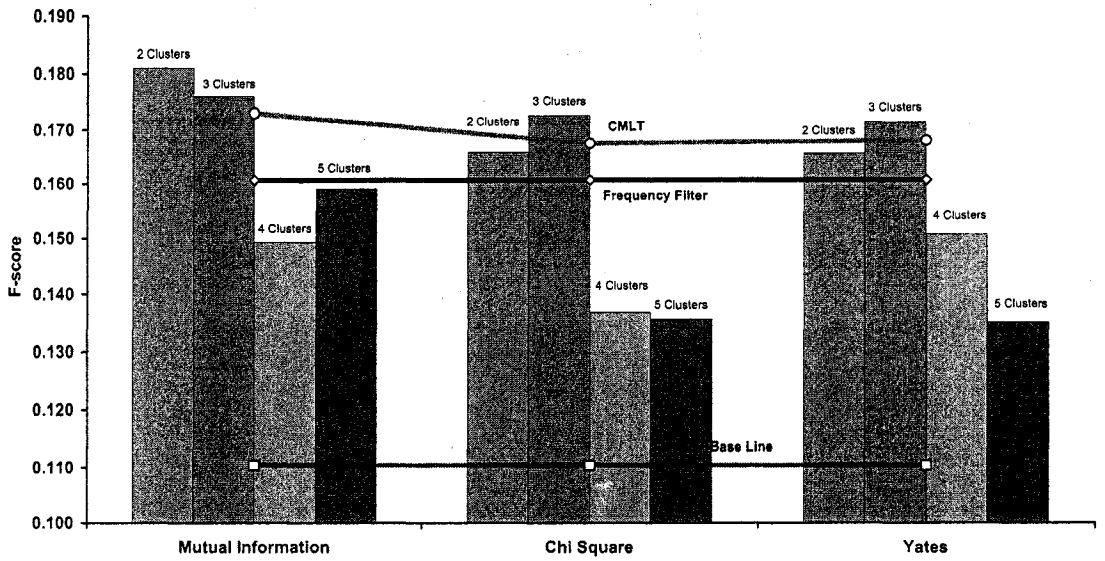

Figure 23: F-score of the System Using Different Similarity Measures in the Test Set

clusters created in the clustering method in the training set using Mutual Information as the similarity measure and single metric as the linkage metric. The difference between the F-score of the best cluster and the F-score of the pinpointed cluster is called the *pinpointing error* and is shown in Figure 24.

As expected, the F-score of the best cluster increases while the number of clusters increases, but the F-score of the system starts to fall down after 3 clusters. As the number of clusters increases the pinpointing error increases because the actual answers and the spies are divided between the clusters and finding the best cluster containing the highest number of actual answers is more tricky.

We can define the performance of the Pinpointing method as F-score of the system over F-score of the best cluster:

$$\textit{Performance} = \frac{\text{F-score of the pinpointed cluster}}{\text{Maximum F-score among clusters}}$$

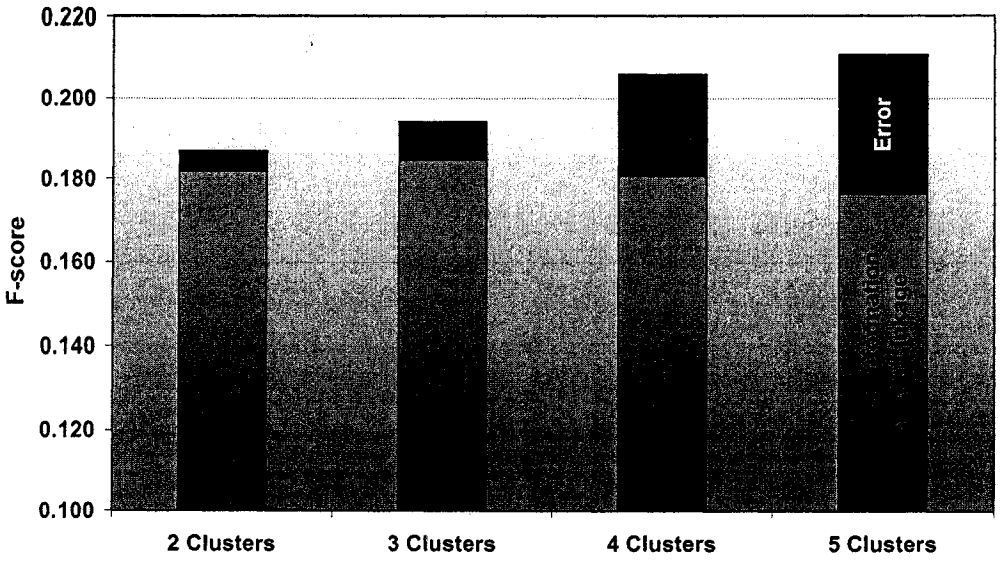

Figure 24: F-score of the Pinpointed Cluster in the Training Set

The performance of the pinpointing method for 2, 3, 4 and 5 clusters are therefore 0.97, 0.95, 0.87 and 0.83 respectively.

5.4 Importing External Candidate Resources

We mentioned in Section 3.3 that we can import the initial candidate list from any external source. In this section, we investigate the use of an external source of candidate answers. We consider using the candidate answers that our *Factoid Question Answering System*, an augmented version of Aranea [Lin and Katz, 2003], generates for *List* questions. The details of our Factoid QA system can be found in [Kosseim *et al.*, 2006] and [Razmara *et al.*, 2007].

For this purpose, the candidate answers of our Aranea system to the *List* questions are extracted and verified using our Term Verification module to remove unlikely candidates. Figure 25 illustrates the F-score obtained using Aranea's suggested candidate terms, using our Candidate Answer Extraction method (see Section 3.3) and using

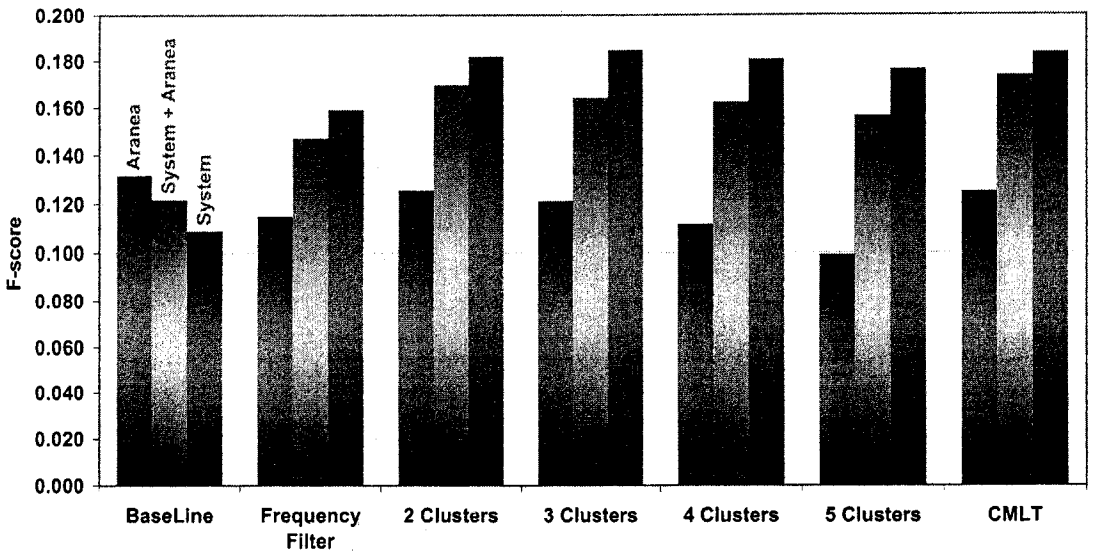

Figure 25: F-score of the System with or without Using an External Candidate Source in the Training set

the combination of the two candidate lists for different candidate selection methods.

As the figure shows, the baseline of the Aranea's candidates is higher than the baseline of the system or the combination. The reason is that the system creates too many initial candidates (113.1 terms per question on average) comparing to the Aranea's candidates (12.4 terms per question on average). Therefore, Aranea's list has a high precision and low recall while the system's list has a low precision and high recall. For the other configurations, using only Aranea's candidates results in a relatively low F-score because of the low number of candidates. An interesting result is that combining Aranea's candidates and the candidates extracted using our system produces slightly lower results than using only the system's candidates. This is because the number of true positives that Aranea will add to the system's list (and they are not already in the system's list) is less (or less effective) than noise it introduces. In other words, Aranea shares some true positive with the system's candidates, on the other hand, Aranea's false positives are mostly high-frequency

noise that may remain in the final candidate answers and hence resulting in a lower precision.

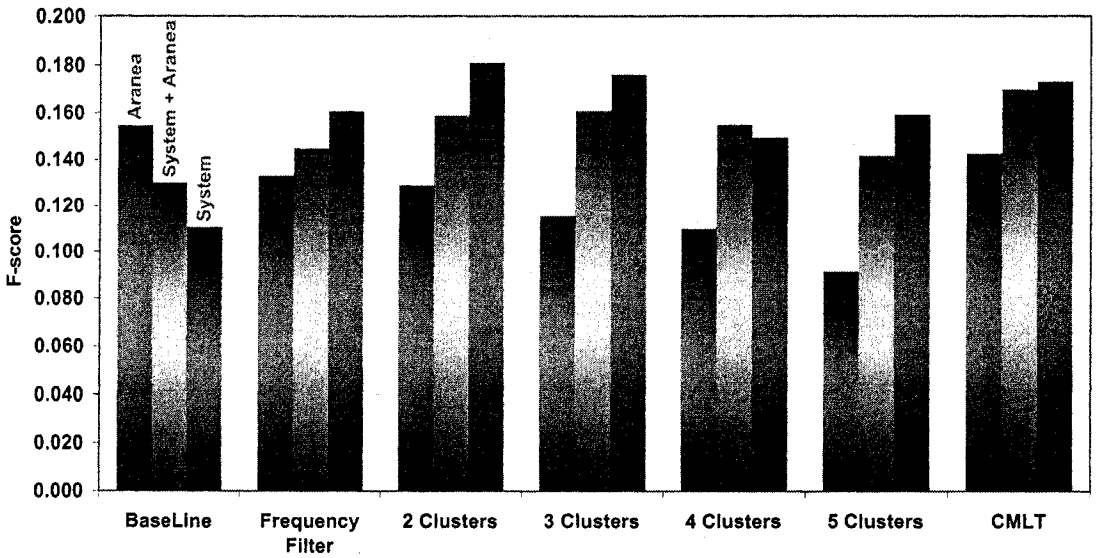

Figure 26: F-score of the System with or without Using an External Candidate Source in the Test Set

Similarly, Figure 26 shows the same results with the test set. In this Figure, Aranea contains a higher baseline as well. For other configuration, except for 4-Cluster, the same relations hold.

5.5 Evaluation of Each Question Type

Figure 27 illustrates precision, recall and F-score of each question type in the training set using Mutual Information as the similarity measure with single linkage.

Figure 28 similarly shows precision, recall and F-score of each question type in the test set. Note that in the test set there are no question of the types NATIONALITY or CITY.

To find out the impact of each question type in the final F-score, the percentage of each question type in the set should be considered (refer to Figures 12 and 13).

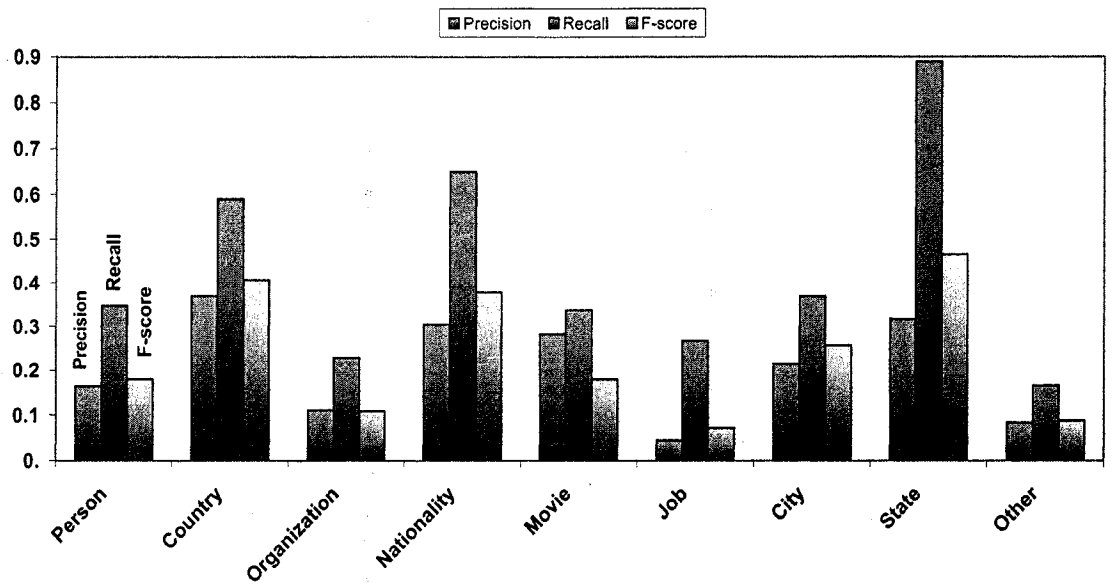

Figure 27: F-score of Each Question Type in the Training Set

5.6 TREC Official Results

In this section, we present the official results of our system submitted to TREC in 2007. In the TREC QA track, each participant is allowed to submit up to three runs. The precision, recall and F-score of our three runs are shown in Table 16.

Run-1 uses the HAC clustering algorithm with Chi-square as the similarity measure to cluster the initial candidate answers. For answer projection, only AQUAINT-2 are taken into account (and not the BLOG06 corpus). In Run-2, the Cumulative Classification method with Chi-square is used to select the final candidate terms and again only AQUAINT-2 is used as the source of supporting documents. However, RUN-3 uses both AQUAINT-2 and BLOG06 for answer projection. In this run, the candidates of our Factoid answering system is also added to the initial candidate list and the Cumulative Classification method (with Chi-square) is used to choose the final answers.

Table 17 shows the number of returned instances in each run and also the number

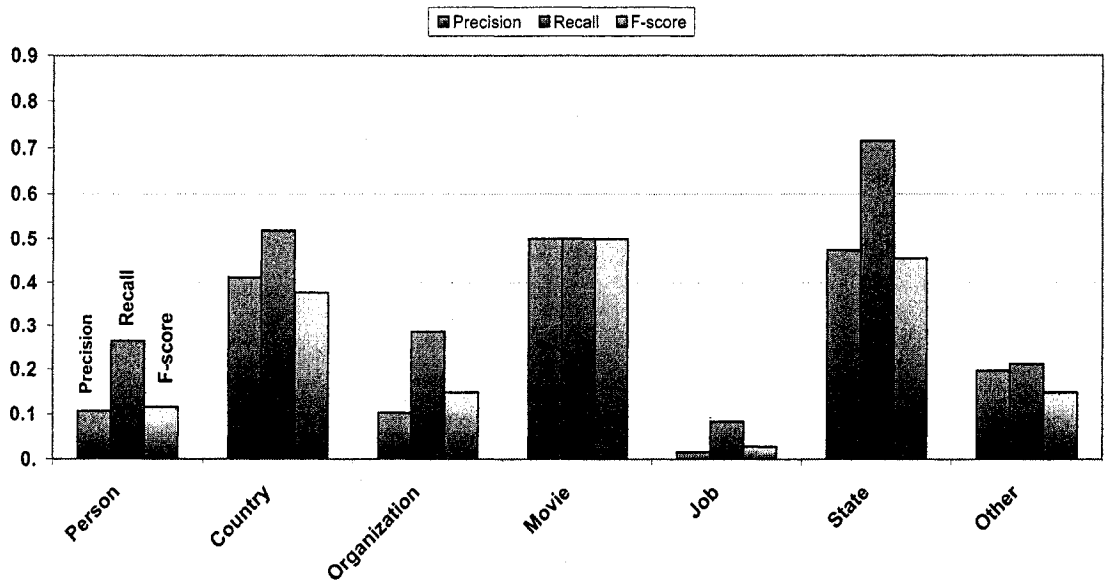

Figure 28: F-score of Each Question Type in the Test Set

	Precision	Recall	F-score
Run-1	0.191	0.129	0.128
Run-2	0.176	0.149	0.134
Run-3	0.175	0.163	0.145
Median	N/A	N/A	0.085

Table 16: Official Results of the 3 Runs Submitted to TREC-2007

and the percentage of *wrong*, *unsupported*, *inexact*, and *globally correct* instances returned (for the definition of these judgements, see Section 1.3). *Correct* refers to globally correct answers and there were no locally correct answers returned. *Distinct* answers are correct answers that are not duplicate and it is used to compute the precision and recall of a run.

There are 946 distinct answers in total, of which 239 distinct answers were found

	Returned	Wrong		Unsupported		Inexact		Correct		Distinct	
	#	#	%	#	%	#	%	#	%	#	%
Run-1	463	269	58.1	81	17.5	9	1.9	104	22.5	96	20.7
Run-2	674	455	67.5	89	13.2	8	1.2	122	18.1	110	16.3
Run-3	771	512	66.4	113	14.7	12	1.6	134	17.4	122	15.8

Table 17: Details of the TREC-2007 Results

only in the BLOG06 corpus [Dang *et al.*, 2007].

The percentage of distinct answers (referred to as overall precision) should not be confused with precision (described in Section 5.1). Percentage of distinct answers is the number of distinct correct answers for all questions over the total number of expected answers while precision is the average of instance precisions (i.e. precision of each question). Since the number of expected answers for different questions is not necessarily the same, precision and percentage of distinct answers may have different values. However, these quantities refer to the same concept and their values are close.

Figure 29 illustrates the results of all participants of the TREC-2007 QA track. The graph shows the F-score of each participant's best run. We (QASCU which stands for "Question Answering System of Concordia University") placed forth among 21 teams participated in TREC-2007 QA track. As the graph shows, there are only two teams with relatively high F-score and the results of the other teams are not very satisfying, which shows that answering *List* questions is still challenging.

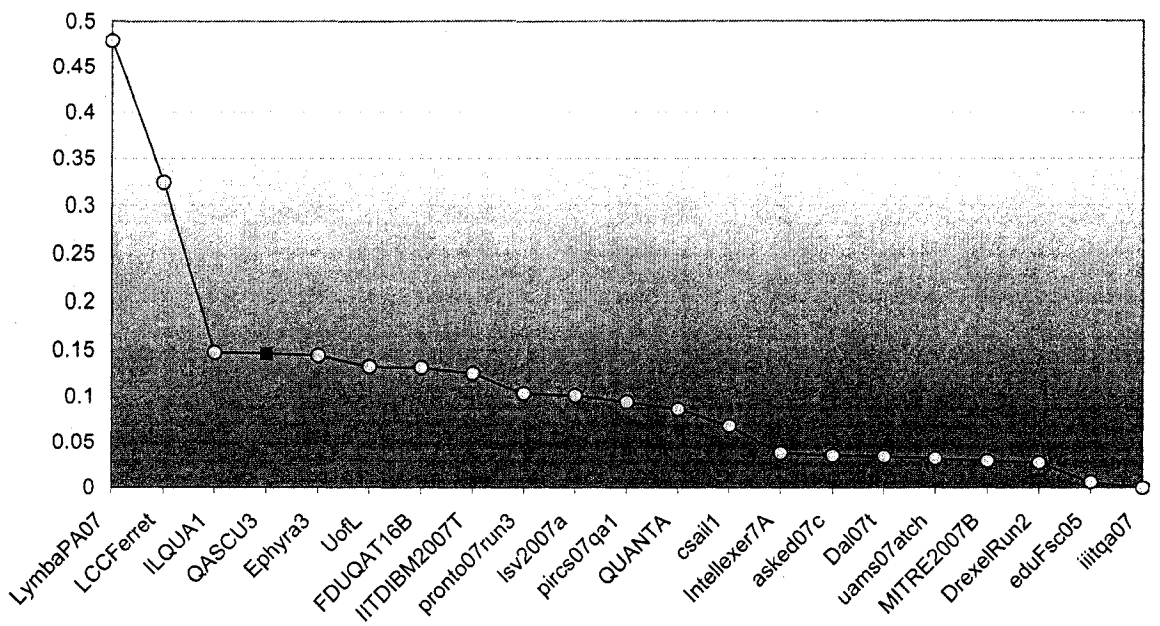

Figure 29: F-score of Best Runs of TREC-2007 Participants in *List* Subtask

Chapter 6

Other Questions: Evaluation and Results

In this chapter, our approach to answering *Other* questions will be evaluated. Section 6.1 will explain the official evaluation metric used by NIST to assess answers to *Other* questions. The section also explains our automatic evaluation method of the system responses. The development results will be described in Section 6.2. Section 6.3, finally, shows the TREC Official Results our system achieved in years 2006 and 2007.

6.1 Evaluation Measures

At TREC, the quality of a system in answering *Other* questions is judged based on precision and recall of the response. The snippets returned by the system are matched against a list of information nuggets developed by NIST assessors. An information nugget is a prototypical answer and is defined as an atomic piece of information about the target that is interesting (in the assessor's opinion) and is not part of an earlier question in the series or an answer to an earlier question in the series. An answer string is tagged as *vital* or *okay* if there is a conceptual match between the answer string and any of the nuggets. The definition of conceptual match is that the match is

independent of the particular wording used in either the nugget or the system output and they are semantically equivalent to one another [Dang *et al.*, 2006].

Given the nugget list and the set of nuggets matched in a system's response, the nugget recall of the response is the ratio of the number of matched vital nuggets to the total number of vital nuggets in the list [Dang *et al.*, 2006].

$$\textbf{\textit{Recall}} = \frac{\text{number of matched vital nuggets returned}}{\text{total number of vital nuggets}}$$

For nugget precision, since there is no effective way of enumerating all the concepts in a response, a length-based measure is used as an approximation to nugget precision. It allows a size of 100 non-white-space characters for each (vital or okay) nugget matched.

$$\textbf{\textit{Allowance}} = 100 * (\text{number of vital and okay nuggets returned})$$

$$\textbf{\textit{Length}} = \text{non-white-space characters in answer}$$

If the total system response is less than or equal to this number of characters, the precision is 1.0, otherwise the precision is the ratio of the allowance to the length of the response for that question [Dang *et al.*, 2006].

$$\textbf{\textit{Precision}} = \begin{cases} 1 & \text{if } \textit{Length} \leq \textit{Allowance} \\ \dfrac{\textit{Allowance}}{\textit{Length}} & \text{if } \textit{Length} > \textit{Allowance} \end{cases}$$

To evaluate the quality of an answer, the F-score (also known as the F-measure) which is a weighted harmonic mean of precision and recall is used.

$$\textbf{\textit{F-score}} = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall}$$

Here, β is the relative weight of recall versus precision. The F-score used for evaluating *Other* questions gives three times more weight to recall than precision; so $\beta = 3$ is used:

$$\textbf{\textit{F-score}}_{\beta=3} = \frac{10 * Precision * Recall}{9 * Precision + Recall}$$

6.1.1 Nugget Pyramids

The vital-okay evaluation has been criticized because only vital nuggets affect nugget recall and it is difficult for systems to achieve non-zero scores on topics with few vital nuggets in the answer key. Thus, assessor errors and other random variations in evaluation conditions highly affects the scores [Dang *et al.*, 2006].

The "*nugget pyramid*" extension of the nugget-based methodology has been proposed and used to evaluate the system responses since TREC-2007. In this method, multiple assessors provide judgments of whether a nugget is vital or okay. Nine different sets of vital-okay judgments were solicited from eight unique assessors (the original question developer votes twice). Using the pyramid procedure, a weight is assigned to each nugget based on the number of assessors who mark it as vital. Given the nugget list and the set of nuggets matched in a systems response, nugget recall was computed as the ratio of the sum of weights of matched nuggets to the sum of

weights of all nuggets in the list [Dang *et al.*, 2007].

$$\textit{Recall} = \frac{\text{sum of weights of matched nuggets returned}}{\text{sum of weights of all vital nuggets}}$$

The precision and F-score in nugget pyramid are measured in the same way as in the primary vital-okay evaluation explained in the previous section.

In TREC-2006, the pyramid extension was also computed for testing purposes along with the original F-score. For 16.4% of the questions, where the original single-assessor F-score of the all participants responses was zero, the nugget pyramid assigned a non-zero F-score. Thus, the F-scores from the nugget pyramids may be more useful since they are more discriminating [Dang *et al.*, 2006].

6.1.2 Evaluation of the System Responses

Since there exists no automatic standard scoring system to evaluate our system responses in the development phase, we automatically compared our responses to the prototypical answers given by NIST assessors and the actual answers submitted by all participants and judged by NIST assessors. If our sentence is identical to a vital or okay answer, we mark it as such (in case of nugget pyramid the weight of the nugget will be assigned to the sentence, instead). If our sentence is not identical but is a substring of a longer vital or okay nugget, then to determine whether it contains the required information, we compare it to the assessors' prototypical answers of that target (marked as vital or okay) using the token-based Jensen-Shannon similarity function[1]. If our sentence is closer to the assessor answer than the longer nugget is, then we consider our sentence as a correct one and mark it the same as the long answer is marked (vital or okay) or in case of nugget pyramid, the weight is assigned

[1] `http://secondstring.sourceforge.net`

Markers Used	F-score
All	0.265
All - Superlative Markers	0.255
All - Numeral Markers	0.257
All - Other Markers	0.266
Best	0.248
Median	0.156

Table 18: Test Results with the 2005 *Other* Questions

to the answer. Having a list of sentences marked as vital, okay or uninteresting, we can then evaluate the score of the question using the same F-score (with $\beta = 3$) as used at TREC.

6.2 Development Results

We participated in the TREC QA track for the first time in 2006. Since TREC *Other* questions was introduced in 2004, we did not have much data for training and testing. Therefore, we used the TREC-2004 questions (65 questions) as the training set and the TREC-2005 questions (75 questions) for the test set.

Table 18 shows the results of the test set using the overall approach along with the contribution of each type of universal marker. The figure marked *All* refers to the final score of the system when using all markers; while *All - X* refers to all markers except for X. *Best* and *Median* refer to the best and median score of all systems submitted to TREC-2005.

As the table shows, numeral and superlative markers increase the results somewhat; while, surprisingly, the keyword markers do not. We suspect that this is due

to two main reasons:

1. To extract the interest marking keywords, a small corpus was used. We only had sentences related to 65 targets of 2004, which were composed of approximately 132,000 words; 44,000 words, for each of the three targets.

2. The TREC 2004 question series do not include the EVENT target type; while this type of target accounts for 24% of the questions in 2005. Since we identified the keyword markers from the 2004 data, we have no specific markers for EVENT types of target. In fact, if we compare the results of the approach per target type (i.e. PERSON, ORGANIZATION, EVENT and THING) we can clearly see that the F-score is lower for the EVENT target type compared to the other target types (see Table 19).

Target Type	Targets	F-score
PERSON	19	0.300
THING	19	0.277
ORGANIZATION	19	0.268
EVENT	18	0.210
OVERALL	75	0.265

Table 19: Test Results with the 2005 Questions per Target Type

6.3 TREC Official Results

We participated in TREC QA track for two years in 2006 and 2007. The system developed in TREC-2006 had been used with no modification in TREC-2007. The next two sections describe the details of our participation in these two years.

6.3.1 TREC-2006

We submitted three runs in TREC-2006 QA track for *Other* questions. Since NIST also provides the output of the PRISE search engine with the target as query, in Run-1 we take the intersection of the top 25 documents returned by Lucene and the top 25 documents returned by PRISE. The idea is that if the two IR systems retrieve the same document using two different search engines and queries, then we should be more confident of its pertinence. Experimentally, we observed that taking the intersection of the two IR outputs increased the final F-score by 0.02 with our testing set. Run-2 uses the top 50 documents returned by Lucene for the domain collection. For the source collection, it adds the top two relevant documents from AQUAINT to the Wikipedia article for term extraction. In Run-3, the top 50 relevant documents returned by Lucene is used as the domain collection.

	F-score	F-score (pyramid)
Run-1	0.197	0.206
Run-2	0.180	0.194
Run-3	0.199	0.203
Median	0.129	0.139

Table 20: Official Results of the 3 Runs Submitted to TREC-2006

Table 20 summarizes the results we obtained using the mentioned configurations. As the table indicates, Run-1 and Run-3 which do not use AQUAINT for term extraction performed better than Run-2 for which in addition to a Wikipedia article, two documents from AQUAINT are used. This seems to confirm our intuition (discussed in Chapter 4) that AQUAINT should not be used for term extraction, and Wikipedia, or another source is more appropriate.

Figure 30 illustrates the results of participants of the TREC-2006 *Other* subtask. The graph shows the F-score of each participant's best run. We (QASCU) placed third among 27 teams participated in TREC-2006 QA track in *Other* questions.

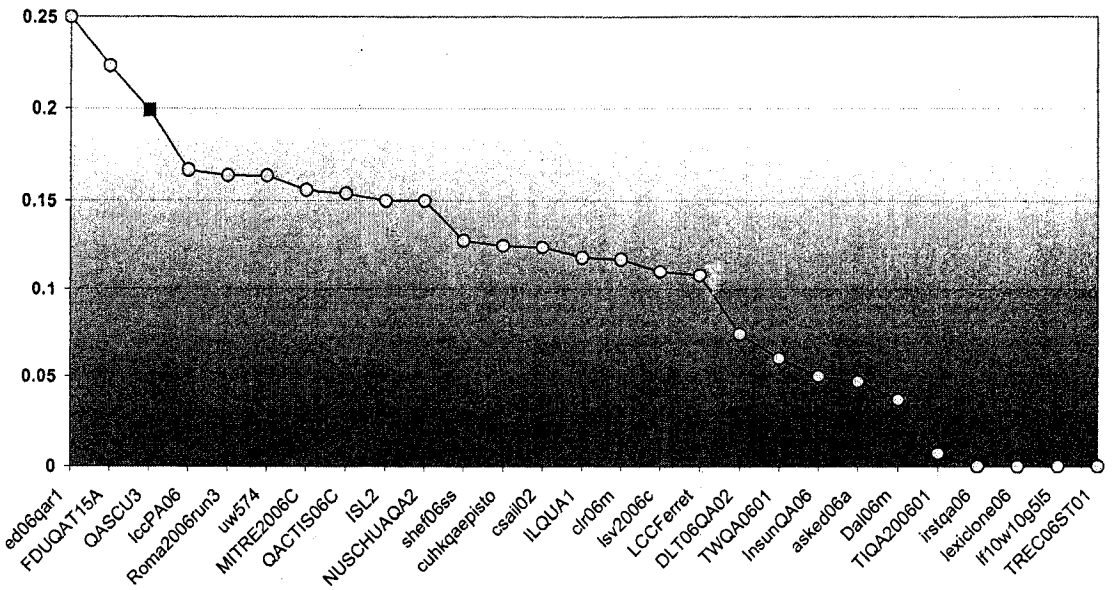

Figure 30: F-score of Best Runs of TREC-2006 Participants in *Other* subtask

6.3.2 TREC-2007

We also participated in TREC-2007 with two different configurations for *Other* questions. Table 21 summarizes our results in TREC-2007 QA track in *Other* subtask. Run-1 extracts the target-specific interest markers from Wikipedia while Run-2 uses Wikipedia and also two documents from AQUAINT-2 as the source of term extraction. Run-3 is the same as Run-1 although its F-score is slightly different. This is due to some minor changes in the ranking of the documents returned by Google in different times.

Although no modifications had been made from TREC-2006 to TREC-2007, the results are much higher in TREC-2007. Beside the fact that in TREC-2007 the

	F-score (pyramid)
Run-1	0.275
Run-2	0.281
Run-3	0.278
Median	0.118

Table 21: Official Results of the 3 Runs Submitted to TREC-2007

pyramid extension was used to evaluate the participants submissions and it usually decreases the number of questions with zero F-score, it seems there are more vital and okay on average for the *Other* questions of TREC-2007.

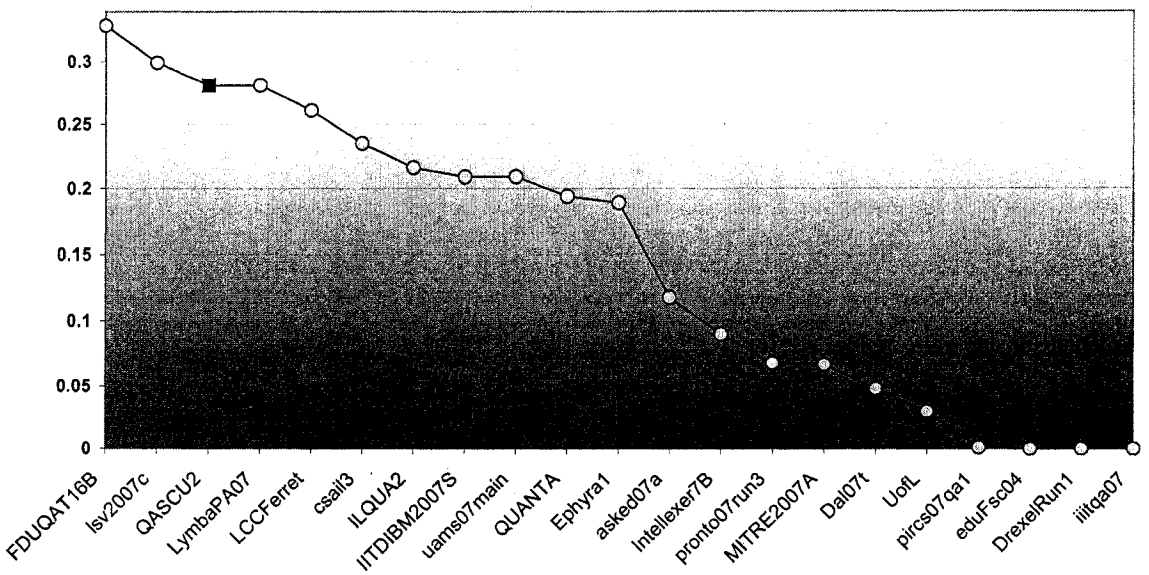

Figure 31: F-score of Best Runs of TREC-2007 Participants in *Other* subtask

Figure 31 shows the results of participants of the TREC-2007 *Other* subtask. Our system placed again third among 21 teams participated in TREC-2007 QA track in *Other* questions.

Chapter 7

Conclusion and Future Work

This chapter summarizes our approach to answering *List* questions and *Other* questions separately. It also presents our future directions in these two areas.

7.1 List Questions

We explained our approach to answering *List* questions in Chapter 3. The approach attempts to select a subset of the initial candidate answers which are extracted from a number of relevant documents. To do so, the information regarding the co-occurrences of the candidates in a larger number of related documents are extracted and a method attempts to group candidates that co-occur more often. The most likely cluster to contain the answers is returned as the final candidates using the notion of spies which are the target and the question keywords.

Using empirical results based on TREC questions and also showing TREC official results, presented in Chapter 5, we showed that our hypothesis which states that instances of the answer to a *List* question tend to co-occur within sentences and tend to co-occur with the question or the target keywords, seems to be correct. The results demonstrate the effectiveness of the approach.

While this approach has been shown promising, there is much room for improvement. The component evaluation carried out in Chapter 5 can be used to identify weak parts of the system. Table 22 shows the performance of each component.

Module	Performance
Answer Type Recognition	0.98
Document Retrieval	0.79
Candidate Answer Extraction	0.62
Clustering	0.32
Pinpointing	0.95

Table 22: Performance of Each Module

As the table shows, the Candidate Answer Extraction and Clustering modules have relatively low performance and possibly need to be improved.

Candidate Answer Extraction: In Section 5.3.3, the quality of the extracted entities for each question type was evaluated. The NE taggers for PERSON, ORGANIZATION and JOB are to be enhanced; especially, for PERSON which constitutes about 32% and 39% of the training and test sets and have a significant impact on the final results. MOVIE and OTHER are also question types that our system does not work well with. A term-type validation module needs to be developed to filter out terms that do not comply with the answer type from the initial list.

Clustering: Clustering is the essence of our approach. It is, however, the most tricky and the weakest module in our system. Although many clustering methods which need absolute data points can not be applied to this module (since we only have the similarities or distances between terms), there are still other clustering method that we could try. In addition, we can use fuzzy clustering, in which data

elements can belong to more than one cluster with a membership degree for each association. Another important issue in clustering is the similarity measure. We have experimented the three similarity measures among which Weighted Mutual Information seems the most promising. However, other similarity measures (e.g. F-test, Phi-squared, etc.) could also be tried.

We could use co-occurrence information at the paragraph level rather than at the sentence level. Therefore, the hypothesis would be "*instances of the answer to a* List *question tend to co-occur within the paragraphs of relevant documents*". In this case, Chi-square which is not appropriate for sparse data, should be replaced with another similarity measure.

7.2 Other Questions

Chapter 4 presented a keyword-based approach to extracting interesting sentences to answer *Other* questions. The method is based on the identification of target-specific and universal interest markers. Target-specific markers are identified by named entities found in the Wikipedia online encyclopedia. The frequency of these named entities in the relevant documents in the corpus are then used as a measure of how interesting they really are. Target-independent interest markers are defined as the most frequent terms in the TREC-2004 *vital* and *okay* nuggets and include superlatives, numerals and specific keywords. Using these markers, we extract and rank sentences from the corpus and return the top-scoring ones as the answer. When using the 2004 TREC data for development, the approach achieved an F-score of 0.265 with the 2005 TREC questions, placing it above the best scoring TREC-2005 system. In addition, the results of our system in TREC-2006 and TREC-2007 showed our approach is promising.

Currently, the system is highly dependent on the Wikipedia articles; we need to

have other robust alternatives in case the proper Wikipedia article is not found or it contains a large amount of data about the target as the results showed using top N corpus documents are not as appropriate. In addition, we need to perform proper co-reference resolution on the Wikipedia terms; this would allow to better rank and identify the interesting terms. Also, computing lexical chains (as in [Ferres *et al.*, 2005]) may improve results as better target-specific markers can be identified; this needs to be investigated.

To represent interesting facts, we currently consider only individual terms. A more precise method would ultimately be to expand the approach to extracting entire predicate structures; with roles and arguments.

Although the use of the universal keyword markers did not seem to improve results, we still believe it is an interesting avenue. Since we have very little training data to identify these keywords, we could try to expand the ones we have with lexical semantics. Finally, since the result of event targets is rather weak, we need to focus more on this type of targets.

Bibliography

[Adafre *et al.*, 2007] S. F. Adafre, M. de Rijke, and E. T. K. Sang. Entity Retrieval. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, Bulgaria, September 2007.

[Adler, 1951] Franz Adler. Yates' correction and the statisticians. *Journal of the American Statistical Association*, 46(256):490–501, 1951.

[Ahn *et al.*, 2005a] D. Ahn, S. Fissaha, V. Jijkoun, K. Muller, M. Rijke, and E. Sang. Towards a Multi-Stream Question Answering-As-XML-Retrieval Strategy. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Ahn *et al.*, 2005b] K. Ahn, J. Bos, J. R. Curran, D. Kor, M. Nissim, and B. Webber. Question Answering with QED at TREC-2005. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Berkhin, 2002] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[Brill *et al.*, 2001] Eric Brill, Jimmy Lin, Michele Banko, Susan T. Dumais, and Andrew Y. Ng. Data-intensive question answering. In *Proceedings of the tenth*

Text REtreival Conference (TREC-2001), Gaithersburg, Maryland, USA, November 2001. National Institute of Standards and Technology (NIST).

[Brill *et al.*, 2002] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the askmsr question-answering system. In *EMNLP'02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 257–264, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[Chakraborti *et al.*, 2007] Sutanu Chakraborti, Nirmalie Wiratunga, Robert Lothian, and Stuart Watt. Acquiring word similarities with higher order association mining. In Rosina Weber and Michael M. Richter, editors, *Case-Based Reasoning Research and Development*, volume 4626 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2007.

[Chen *et al.*, 2005] J. Chen, P. Yu, and H. Ge. UNT 2005 TREC QA Participation: Using Lemur as IR Search Engine. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Church and Hanks, 1990] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[Clarke *et al.*, 2001] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *SIGIR'01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365, New York, NY, USA, 2001. ACM.

[Cohen *et al.*, 2003] W. W. Cohen, P. Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the*

IJCAI Workshop on Information Integration on the Web (IIWeb), pages 73-78, Acapulco, Mexico, 2003.

[Cui *et al.*, 2004] Hang Cui, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen Kan. National University of Singapore at the TREC-13 Question Answering Main Task. In *Proceedings of the Thirteenth Text REtreival Conference (TREC-2004)*, Gaithersburg, Maryland, USA, November 2004. National Institute of Standards and Technology (NIST).

[Cui *et al.*, 2005] Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *SIGIR'05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407, New York, NY, USA, 2005. ACM.

[Cunningham *et al.*, 2002] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[Cunningham *et al.*, 2007] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Cristian Ursu, Marin Dimitrov, Mike Dowman, Niraj Aswani, Ian Roberts, Yaoyong Li, and Andrey Shafirin. Developing language processing components with GATE, University of Sheffield. Technical report, July 2007.

[Dalmas and Webber, 2007] Tiphaine Dalmas and Bonnie L. Webber. Answer comparison in automated question answering. *Journal of Applied Logic*, 5(1):104–120, 2007.

[Dang *et al.*, 2006] Hoa Trang Dang, Diane Kelly, and Jimmy Lin. Overview of the TREC 2006 Question Answering Track. In *Proceedings of the 15th Text REtreival*

Conference (TREC-2006), Gaithersburg, Maryland, USA, November 2006. National Institute of Standards and Technology (NIST).

[Dang *et al.*, 2007] Hoa Trang Dang, Diane Kelly, and Jimmy Lin. Overview of the TREC 2007 Question Answering Track. In *Proceedings of the 16th Text REtreival Conference (TREC-2007)*, Gaithersburg, Maryland, USA, November 2007. National Institute of Standards and Technology (NIST).

[Dornfest *et al.*, 2006] Rael Dornfest, Paul Bausch, and Tara Calishain. *Google Hacks: Tips & Tools for Finding and Using the World's Information (Hacks).* O'Reilly Media, Inc., third edition, 2006.

[Echihabi and Marcu, 2003] Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *ACL'03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 16–23, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Fano, 1961] Robert M. Fano. Transmission of Information: A Statistical Theory of Communications. *American Journal of Physics*, 29:793–794, November 1961.

[Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication).* The MIT Press, May 1998.

[Ferres *et al.*, 2005] D. Ferres, S. Kanaan, D. Dominguez-Sal, E. Gonzalez, A. Ageno, M. Fuentes, H. Rodriguez, M. Surdeanu, and J. Turmo. TALP-UPC at TREC 2005: Experiments Using a Voting Scheme Among Three Heterogeneous QA Systems. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Furnas, 2008] George Furnas. Faculty Profile: George Furnas, University of Michigan, School of Information, 2008. `http://www.si.umich.edu/people/faculty-detail.htm?sid=41` [Online; accessed 01-April-2008].

[Gaizauskas *et al.*, 2005] R. Gaizauskas, M. Greenwood, H. Harkema, M. Hepple, H. Saggion, and A. Sanka. The University of Sheffield's TREC 2005 Q&A Experiments. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Ghahramani and Heller, 2005] Z. Ghahramani and K. A. Heller. Bayesian Sets. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2005.

[Hao *et al.*, 2006] Tianyong Hao, Qingtian Zeng, and Liu Wenyin. Semantic pattern for user-interactive question answering. In *SKG'06: Proceedings of the Second International Conference on Semantics, Knowledge, and Grid*, page 17, Washington, DC, USA, 2006. IEEE Computer Society.

[Harabagiu *et al.*, 2005] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang. Employing Two Question Answering Systems in TREC-2005. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Harris, 1954] Zelling Harris. *The Structure of Language*, chapter Distributional Structure, pages 33–49. Prentice-Hall, 1954.

[Hatcher and Gospodnetic, 2004] Erik Hatcher and Otis Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA, 2004.

[Jain *et al.*, 1999] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Jijkoun *et al.*, 2004] Valentin Jijkoun, Maarten de Rijke, and Jori Mur. 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, 2004. Association for Computational Linguistics.

[Jijkoun *et al.*, 2006] Valentin Jijkoun, Joris van Rantwijk, David Ahn, Erik Tjong Kim San, and Maarten de Rijke. The University of Amsterdam at CLEF@QA 2006. In C. Peters A. Nardi and J.L. Vicedo, editors, *Working Notes CLEF 2006*, September 2006.

[Katz and Lin, 2003] Boris Katz and Jimmy Lin. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*, April 2003.

[Katz *et al.*, 2005] B. Katz, G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner, and A. Wilcox. External Knowledge Sources for Question Answering. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Ko *et al.*, 2007] J. Ko, L. Si, and E. Nyberg. A Probabilistic Graphical Model for Joint Answer Ranking in Question Answering. In *SIGIR'07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, July 2007.

[Kor, 2005] K. W. Kor. Improving answer precision and recall of list questions. Master's thesis, School of Informatics, University of Edinburgh, 2005.

[Kosseim *et al.*, 2006] Leila Kosseim, Alex Beaudoin, Abolfazl Keighobadi Lamjiri, and Majid Razmara. Concordia University at the TREC-QA Track. In *Proceedings of the Fifteenth Text Retrieval Conference (TREC-2006)*, Gaithersburg, Maryland, USA, November 2006. National Institute of Standards and Technology (NIST).

[Li, 2002] Hang Li. Word clustering and disambiguation based on co-occurrence data. *Natural Language Engineering*, 8(1):25–42, 2002.

[Lin and Demner-Fushman, 2006] Jimmy Lin and Dina Demner-Fushman. Will pyramids built of nuggets topple over? In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL-2006)*, pages 383–390, New York, 2006.

[Lin and Katz, 2003] Jimmy Lin and Boris Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *CIKM'03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 116–123, New York, NY, USA, 2003. ACM.

[Lin, 2007] Jimmy Lin. Is Question Answering Better than Information Retrieval? Towards a Task-Based Evaluation Framework for Question Series. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference (HLT-NAACL-2007)*, pages 212–219, Rochester, New York, April 2007.

[Long *et al.*, 2004] Johnny Long, Ed Skoudis, and Alrik van Eijkelenborg. *Google Hacking for Penetration Testers.* Syngress Publishing, 2004.

[Macdonald and Ounis, 2006] Craig Macdonald and Iadh Ounis. The TREC Blogs06 Collection : Creating and Analysing a Blog Test Collection. Technical report, University of Glasgow, Department of Computing Science, 2006.

[Magnini *et al.*, 2006] Bernardo Magnini, Danilo Giampiccolo, Pamela Forner, Christelle Ayache, Valentin Jijkoun, Petya Osenova, Anselmo Peñas, Paulo Rocha, Bogdan Sacaleanu, and Richard F. E. Sutcliffe. Overview of the CLEF 2006 multilingual question answering track. In Carol Peters, Paul Clough, Fredric C. Gey, Jussi Karlgren, Bernardo Magnini, Douglas W. Oard, Maarten de Rijke, and Maximilian Stempfhuber, editors, *Evaluation of Multilingual and Multi-modal Information Retrieval, 7th Workshop of the Cross-Language Evaluation Forum, CLEF 2006, Revised Selected Papers*, September 2006.

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

[Matsuo and Ishizuka, 2004] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[Miettinen, 1974] Olli S. Miettinen. Some reasons for not using the Yates continuity correction on 2 $\times$ 2 contingency tables. *Journal of the American Statistical Association*, 69(346):380–382, June 1974.

[Miller and Charles, 1991] George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[Miller, 2006] Michael Miller. *Googlepedia: The Ultimate Google Resource.* Que Corp., Indianapolis, IN, USA, 2006.

[Moldovan *et al.*, 2002] Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, V. Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest

Bolohan. LCC tools for question answering. In *Proceedings of the Eleventh Text REtreival Conference (TREC-2002)*, Gaithersburg, Maryland, USA, November 2002. National Institute of Standards and Technology (NIST).

[Moldovan *et al.*, 2003] D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. Cogex: A logic prover for question answering. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Edmonton, May-June 2003.

[Moldovan *et al.*, 2004] Dan Moldovan, Sanda Harabagiu, Christine Clark, Mitchell Bowden, John Lehmann, and John Williams. Experiments and analysis of LCCs two QA systems over TREC 2004. In *Proceedings of the Thirteenth Text REtreival Conference (TREC-2004)*, Gaithersburg, Maryland, USA, November 2004. National Institute of Standards and Technology (NIST).

[Ravichandran and Hovy, 2001] Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *ACL'02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[Ravichandran *et al.*, 2003] Deepak Ravichandran, Eduard Hovy, and Franz Josef Och. Statistical QA - classifier vs. re-ranker: What's the difference? In *Proceedings of the ACL 2003 workshop on Multilingual Summarization and Question Answering*, pages 69–75, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Razmara and Kosseim, 2007] Majid Razmara and Leila Kosseim. A little known fact is... answering other questions using interest-markers. In Alexander F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing, 8th International*

Conference, CICLing, volume 4394 of *Lecture Notes in Computer Science*, pages 518–529, Mexico City, Mexico, February 2007. Springer.

[Razmara and Kosseim, 2008] Majid Razmara and Leila Kosseim. Answering list questions using co-occurrence and clustering. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008.

[Razmara *et al.*, 2007] Majid Razmara, Andrew Fee, and Leila Kosseim. Concordia University at the TREC 2007 QA track. In *Proceedings of the Sixteeth Text Retrieval Conference (TREC-2007)*, Gaithersburg, Maryland, USA, November 2007. National Institute of Standards and Technology (NIST).

[Resnik, 2001] Philip Resnik. Corpora: Negative mutual information?, 2001. `http://listserv.linguistlist.org/cgi-bin/wa?A2=ind0103&L=corpora&P=3085` [Online; accessed 10-June-2008].

[Roussinov *et al.*, 2005] D. Roussinov, M. Chau, E. Filatova, and J. Robles-Flores. Building on Redundancy: Factoid Question Answering, Robust Retrieval and the "Other". In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Shen and Klakow, 2006] Dan Shen and Dietrich Klakow. Exploring correlation of dependency relation paths for answer extraction. In *ACL'06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 889–896, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[Soricut and Brill, 2004] Radu Soricut and Eric Brill. Automatic question answering: Beyond the factoid. In Susan Dumais, Daniel Marcu, and Salim Roukos,

editors, *Human Language Technologies 2004: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference (HLT-NAACL-2004)*, pages 57–64, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[Soubbotin, 2001] M. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of the tenth Text REtreival Conference (TREC-2001)*, Gaithersburg, Maryland, USA, November 2001. National Institute of Standards and Technology (NIST).

[Spence and Owens, 1990] Donald P. Spence and Kimberly C. Owens. Lexical co-occurrence and association strength. *Psycholinguistic Research*, 19(5):317–330, 1990.

[Toutanova *et al.*, 2003] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL'03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Turney, 2001] Peter D. Turney. Mining the Web for synonyms: PMI–IR versus LSA on TOEFL. In Luc De Raedt and Peter A. Flach, editors, *Twelth European Conference on Machine Learning (EMCL 2001)*, September 5-7 2001.

[Voorhees and Dang, 2005] Ellen M. Voorhees and Hoa T. Dang. Overview of the TREC 2005 Question Answering Track. In *Proceedings of the 14th Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Voorhees and Tice, 1999] Ellen M. Voorhees and Dawn M. Tice. The trec-8 question answering track evaluation. In *Proceedings of the 8th Text REtreival Conference (TREC-1999)*, Gaithersburg, Maryland, USA, November 1999. National Institute of Standards and Technology (NIST).

[Voorhees, 2001] Ellen M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text REtreival Conference (TREC-2001)*, Gaithersburg, Maryland, USA, November 2001. National Institute of Standards and Technology (NIST).

[Voorhees, 2004] Ellen M. Voorhees. Overview of the TREC 2004 Question Answering Track. In *Proceedings of the 13th Text REtreival Conference (TREC-2004)*, Gaithersburg, Maryland, USA, November 2004. National Institute of Standards and Technology (NIST).

[Whittaker *et al.*, 2005] Edward Whittaker, Sadaoki Furui, and Dietrich Klakow. A statistical classification approach to question answering using web data. In *CW'05: Proceedings of the 2005 International Conference on Cyberworlds*, volume 0, pages 421–428, Washington, DC, USA, 2005. IEEE Computer Society.

[Whittaker *et al.*, 2006] E. Whittaker, J. Novak, P. Chatain, and S. Furui. TREC2006 Question Answering Experiments at Tokyo Institute of Technology. In *Proceedings of the Fifteenth Text Retrieval Conference (TREC-2006)*, Gaithersburg, Maryland, USA, November 2006. National Institute of Standards and Technology (NIST).

[Wu and Strzalkowski, 2006] Min Wu and Tomek Strzalkowski. Utilizing co-occurrence of answers in question answering. In *ACL'06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 1169–1176, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[Wu *et al.*, 2005a] L. Wu, X. Huang, Y. Zhou, Z. Zhang, and F. Lin. FDUQA on TREC2005 QA Track. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Wu *et al.*, 2005b] M. Wu, M. Duan, S. Shaikh, S. Small, and T. Strzalkowski. ILQUA An IE-Driven Question Answering System. In *Proceedings of the Fourteenth Text REtreival Conference (TREC-2005)*, Gaithersburg, Maryland, USA, November 2005. National Institute of Standards and Technology (NIST).

[Xu *et al.*, 2002] J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of the Eleventh Text Retrieval Conference (TREC-2002)*, Gaithersburg, Maryland, USA, November 2002. NIST, National Institute of Standards and Technology (NIST).

[Yang and Chua, 2004] Hui Yang and Tat-Seng Chua. Fada: Find all distinct answers. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 304–305, New York, NY, USA, 2004. ACM.

[Yates, 1934] Frank Yates. Contingency table involving small numbers and the χ^2 test. *Supplement to the Journal of the Royal Statistical Society*, pages 217–235, 1934.

[Zhou *et al.*, 2006] Y. Zhou, X. Yuan, J. Cao, X. Huang, and L. Wu. FDUQA on TREC2006 QA Track. In *Proceedings of the Fifteenth Text Retrieval Conference (TREC-2006)*, Gaithersburg, Maryland, USA, November 2006. National Institute of Standards and Technology (NIST).