QUESTION ANSWERING USING DOCUMENT TAGGING AND QUESTION CLASSIFICATION

Stephen Dubien

B.Sc, University of Lethbridge, 2003

A Thesis

Submitted to the School of Graduate Studies

of the University of Lethbridge

in Partial Fulfillment of the

Requirements for the Degree

MASTER OF SCIENCE

(COMPUTER SCIENCE)

Department of Computer Science

University of Lethbridge

LETHBRIDGE, ALBERTA, CANADA

Abstract

Question answering (QA) is a relatively new area of research. QA is retrieving answers to questions rather than information retrieval systems (search engines), which retrieve documents. This means that question answering systems will possibly be the next generation of search engines. What is left to be done to allow QA to be the next generation of search engines? The answer is higher accuracy, which can be achieved by investigating methods of questions answering.

I took the approach of designing a question answering system that is based on document tagging and question classification. Question classification extracts useful information from the question about how to answer the question. Document tagging extracts useful information from the documents, which will be used in finding the answer to the question. We used different available systems to tag the documents. Our system classifies the questions using manually developed rules.

I also investigated different ways which can use both of these methods to answer questions and found that our methods had a comparable accuracy to some systems that use deeper processing techniques. This thesis includes investigations into modules of a question answering system and gives insights into how to go about developing a question answering system based on document tagging and question classification. I also evaluated our current system with the questions from the TREC 2004 question answering track.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There are many computer science innovations that make difficult tasks more manageable for people. Databases were developed to replace paper filing systems, which were difficult to manage. Faster processors are being developed for computers, so calculations that would take a human days, take only seconds for a computer to complete. Artificial Intelligence (AI) is an area of computer science concerned with operations that use intelligence to support decision making. With AI, tedious tasks that previously had to be done by a human can now be done with a computer. Question answering is one of the fields where a human task is being made easier by AI.

The problem that question answering systems consider is; given a set of documents and a question, find the answer to the question in that set of documents. This task would normally be performed by a human by indexing the collection of documents with an information retrieval system. Information retrieval systems, also known as search engines, are successful in retrieving documents, based on a query, from a collection of documents. Then when a question needs to be answered, a query will be created to retrieve documents relevant to the question. Finally, each document retrieved would be manually read until an answer to the question is found, or all the documents have been read. This method is time consuming, and a correct answer could easily be missed, by either an incorrect query, resulting in missing documents, or by careless reading. As well, the time needed for reading the documents might be wasted because the answer may not exist in any of the documents in the collection.

Question Answering (QA) systems take in a natural language question, and return the answer from the set of documents, if the answer can be found. Answer retrieval, rather than document retrieval, will be integral to the next generation of search engines. Currently, there is a web site called AskJeeves^{TM}[1] , that attempts to retrieve documents to answer a question. This handles one of the problems a question answering system has, but there is still the task of reading through the documents retrieved. Question answering systems will handle query creation, and finding the exact entity that is the answer.

For example, if someone wants to know who shot Abraham Lincoln, without a QA system, she would first do a search on a relevant set of documents (i.e. the Internet) with a search engine. She would then formulate a query such as "Abraham Lincoln shot" to retrieve documents from the search engine. She would next read through the retrieved documents, then possibly change her search parameters, and try again if she was not happy with the retrieved documents. With a QA system she will just enter in the question, "Who shot Abraham Lincoln?", and the system will retrieve the most probable answer.

QA systems employ information retrieval to retrieve documents relevant to the question, and then use information extraction techniques to extract the answer from those documents. Natural Language Processing (NLP) is the division of Artificial Intelligence (AI) responsible for understanding language. Information Extraction (IE) (Jurafsky and Martin, 2000, pages 307–317) is an area of NLP responsible for extracting information from a set of documents. IE is related to QA because IE is about extracting data from documents, and QA systems are about extracting only relevant data (Moldovan et al., 2000).

[1]http://www.askjeeves.com/

1.1 Information Extraction

The Message Understanding Conference (MUC) [2] challenged IE systems to extract different types of information from a collection of documents. The following are the tasks, with examples, from 1998 MUC-7 outline (Chinchor, 1998), which was the last year of the conference.

1.1.1 Named Entities Task

Named Entities (NE) are proper nouns and quantities of interest. They include locations, dates, times, percentages and monetary amounts. Systems had to locate and tag these entities in documents.

An example of an NE tagged document is:

> The ⟨ENAMEX TYPE="LOCATION"⟩U.K.⟨/ENAMEX⟩ satellite television broadcaster said its subscriber base grew ⟨NUMEX TYPE= "PERCENT"⟩ 17.5 percent⟨/NUMEX⟩ during ⟨TIMEX TYPE="DATE"⟩ the past year⟨/TIMEX⟩ to 5.35 million

Tagging NEs is helpful in question answering systems (Srihari and Li, 1999). For instance, in the above example, if a question answering system was looking for a percentage as an answer, it could eliminate all the words except *17.5 percent*. This is a large step towards finding a correct answer.

1.1.2 Coreference Task

This task focused on finding which entity a phrase is referring to. This also includes discovering which entity a pronoun is referring to.

An example of a coreference is:

[2]http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

⟨ENTITY1 The U.K. satellite television broadcaster ⟩ said ⟨ENTITY1 its ⟩ ⟨ENTITY2 subscriber base⟩ grew 17.5 percent during the past year to ⟨ENTITY2 5.35 million ⟩

Coreferencing finds that *its* is a reference to *The U.K. satellite television broadcaster*, and also finds that *5.35 million* is referring to *subscriber base*. If coreferencing is done over multiple sentences, the pronoun's reference can be resolved with an entity that appears a few sentences before.

Litkowski (1999) found that, in question answering, some questions could not be answered if the entities were not coreferenced. Knowing which words are referring to which entity is very helpful for systems that will form the question and the answer into Logical Form Representation (LFR) (Moldovan et al., 2002). These systems will be discussed later in this chapter.

1.1.3 Template Element Task

The following tasks involved putting data, from documents in a collection, into templates. The first template is called a Template Element (TE), and is made up of name, type, descriptor, and category slots. The name slot will be made up of all names for a single entity. The type of a TE can be a person, organization, artifact, or location. The descriptor and category are of predefined types. For instance, a person can be of the category of civilian, military, or other. An example of a TE is:

```
<ENTITY-9602040136-11> :=
ENT_NAME: "Dennis Gillespie"
ENT_TYPE: PERSON
ENT_DESCRIPTOR: "Capt."
/ "the commander of Carrier Air Wing 11"
ENT_CATEGORY: PER_MIL
```

Keeping track of entities is helpful with coreferences and attaching extra information to those entities. The QA system QUALIFIER (Yang et al., 2003) uses a type of TE representing events. When a question is asked about an event, all the data about the event is compiled into an event template and is used to help find the answer to the question.

1.1.4 Template Relation Task

Template Relations (TR) are relationships between TEs. MUC-7 has only the relations: employee_of, product_of or location_of. For instance, the employee_of will be a relation between a TE of type PERSON and a TE of type ORGANIZATION.

An example of a TR is:

```
<EMPLOYEE_OF-9602040136-5> :=
PERSON: <ENTITY-9602040136-11>
ORGANIZATION: <ENTITY-9602040136-1>
<ENTITY-9602040136-11> :=
ENT_NAME: "Dennis Gillespie"
ENT_TYPE: PERSON
ENT_DESCRIPTOR: "Capt."
/ "the commander of Carrier Air Wing 11"
ENT_CATEGORY: PER_MIL
<ENTITY-9602040136-1> :=
ENT_NAME: "NAVY"
ENT_TYPE: ORGANIZATION
ENT_CATEGORY: ORG_GOVT
```

1.1.5 Scenario Template Task

The Scenario Template (ST) is a larger template that is organized around a type of event. Participating groups in MUC-7 did not know the type of scenario until one month prior to testing.

An example of an ST event would be someone leaving a position in a company. The ST for this example will include:

- Who left
- When did this event take place
- What were their reasons for leaving
- Which company is involved
- What position did they leave
- Who is going to succeed them

Pasca and Harabagiu (2001a) suggest that a question, such as "*What management successions occurred at IBM in 1999?*", could be answered by filling templates. These templates will be similar to the ST templates from MUC, and can be used to answer questions that require this kind of summary.

These tasks were done in a restricted-domain. Restricted-domain problems are different from open-domain problems for both information extraction and question answering.

1.2 Open-Domain vs. Restricted-Domain Question Answering

This thesis is concerned with open-domain question answering. The counterpart to open-domain is restricted-domain. The domain that is being referred to is the subject of the documents being used to answer the questions. Restricted-domain documents are of a known subject, and sometimes include a known format or a knowledge base, such as the PLANES system (Waltz, 1978). The PLANES system's domain is aircraft maintenance and flight data and uses a knowledge base contained in a database. When a user asked a question, the system turned their question into a database query.

If a restricted-domain system does not use a knowledge base, the subject of the data is known beforehand. In these cases, extraction patterns can be created such that the system can easily access the information contained in the documents of a certain subject. KAAS (Diekema, Yilmazel, and Liddy, 2004) is a system that performs such extraction techniques in a restricted-domain system.

Open-domain question answering is when any document set can be used and extraction techniques will not be tailored to the subject of the data. This means questions about any subject can be asked, which makes extracting information increasingly difficult since IE systems are highly dependent on domain knowledge (Moldovan et al., 2000).

The Text REtrieval Conference[3] (TREC) is a series of workshops organized by the National Institute of Standards and Technology (NIST), designed to advance the state of the art in handling and retrieving information. One of the recent areas of interest of TREC (Voorhees, 2004) has been question answering, which is an experiment in open-domain question answering. The document set is the AQUAINT data set which is made up of newswire documents from Associated Press, New York Times, and Xinhua News Service from the People's Republic of China. Newswire documents are taken from a number of years and are on different subject matters such as entertainment, sports and politics.

The TREC question answering track currently has three types of questions:

Factoid questions that require only one answer. Example: *Who was the president of the United States in 1895?*

List questions that require a non-redundant list of answers. Example: *Which countries had terrorist attacks in 1994?*

[3]http://trec.nist.gov/

Definition type questions that require a non-redundant list of facts about a subject.
Example: *Who is Paul Martin?*

TREC questions get increasingly difficult each year, which requires groups that participate in the question answering track to improve their systems. This develops new ideas in question answering, while giving a method with which to compare systems. The papers that are in the proceedings of TREC provide a good sense of what kinds of question answering systems are currently being developed.

1.3 State of the Art Question Answering Systems

A question answering system must analyze the question, extract answers from the set of documents, and then choose an answer to the question. Groups working on question answering systems are trying new directions in QA research to see which methods provide the best results. The following are types of systems that are currently being developed. The success of each of the systems will be discussed in a later chapter.

1.3.1 Knowledge Base Systems

Some systems that participate in this open-domain track use restricted-domain methods, such as a knowledge base. Knowledge base systems involve extracting certain information beforehand and using it later.

MAYA (Kim et al., 2001) creates a database of answers before any questions are asked. There are only fifteen types of entities this system considers as answers. Each passage that contains a possible answer is kept, and when a question is asked, the answer that is contained in passages most closely related to the question is given as the answer. Katz et al. (2003) developed a similar method of question answering that uses knowledge bases to compile facts about every subject before any definition questions are asked.

Clifton and Teahan (2004) built a knowledge base of questions from the document set. They use knowledgeable agents (Teahan, 2003) that are based on the knowledge grids proposed by Cannataro and Talia (2003). These knowledgeable agents go through the documents and form questions around entities they find. For instance, from the phrase, *"John Lennon died on December 8th, 1980 during a public dramatic interpretation of J.D. Salinger's Catcher in the Rye."* their system forms the question-answer pair, *"When did John Lennon die?"* and *"December 8th, 1980"*. When a question is asked, the system will check whether it has the knowledge to answer the question by determining which questions they have identified match the incoming question.

Any information that is extracted before a question is asked will not have to be extracted when that question is asked, thus providing a faster answer than if it had to be extracted on answer retrieval time.

1.3.2 Logical Form Representation Systems

These systems attempt to form a Logical Representation (LR) of the question and sentences that contain a possible answer. They use the logical form to determine whether the logical form of a possible answer follows from the logical form of the question. PowerAnswer 2 (Moldovan et al., 2004) is a QA system that uses logical proving. Their method is outlined in Moldovan et al. (2002). The algorithm involves defining nouns as entities. These entities are then modified by verbs, adjectives and semantic categories, which are used to answer questions.

An example of how their system answers a question is:

Question: *What is the Muslim Brotherhood's goal?*

Question LR:

```
(exists x0 x1 x2 x3 (Muslim_NN(x0) & Brotherhood_NN(x1) &
nn_NNC(x2,x0,x1) & PURPOSE_SR(x3,x2)))
```

Their system defined x0 and x1 as *Muslim* and *Brotherhood* respectively, then combines them to make entity x2. Their system knows that a "goal" is equivalent to the Semantic Relation (SR) PURPOSE, and x3 will be the final goal for entity x2 (Muslim Brotherhood).

The next step is to turn passages with prospective answers into LR form. In this case the answer is contained in this passage:

> The Muslim Brotherhood, Egypt's biggest fundamentalist group established in 1928, advocates turning Egypt into a strict Muslim state by political means, setting itself apart from militant groups that took up arms in 1992.

And its logical form is:

```
(exists e1 e2 e3 e4 e5 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11
x12 x13 x14 x15 (Muslim_NN(x1) & Brotherhood_NN(x2) &
nn_NNC(x3,x1,x2) & Egypt_NN(x4) & _s_POS(x5, x4) &
biggest_JJ(x5) & fundamentalist_JJ(x5) & group_NN(x5)
& SYNONYMY_SR(x3,x5) & establish_VB(e1,x20,x5) &
in_IN(e1,x6) & 1928_CD(x6) & TEMPORAL_SR(x6,e1) &
advocate_VB(e2,x5,x21) & AGENT_SR(x5,e2) &
PURPOSE_SR(e3,e2) & turn_VB(e3,x5,x7) & Egypt_NN(x7)
& into_IN(e3,x8) & strict_JJ(x15,x14) & Muslim_NN(x8)
& state_NN(x13) & nn_NNC(x14,x8,x13) &
PROPERTY_SR(x15,x14) & by_IN(e3,x9) & political_JJ(x9)
& means_NN(x9) & MEANS_SR(x9,e3) & set_VB(e5,x5,x5) &
itself_PRP(x5) & apart_RB(e5) & from_IN(e5, x10) &
militant_JJ(x10) & group_NN(x10) & take_VB(e6,x10,x12)
& up_IN(e6,x11) & arms_NN(x11) & in_IN(e6,x12)&
1992_CD(x12) & TEMPORAL_SR(x12,e6)
```

The answer will be in a PURPOSE semantic relation and the system already knows that it will be the *Muslim Brotherhood*'s PURPOSE, which means that the answer to the question will be the other phrase involved in that PURPOSE relation. The other phrase refers to the phrase e3, which is *"turning Egypt into a strict Muslim state by political means"*. These pre-defined semantic relations make it possible to answer these types of questions.

1.3.3 Multi Corpus Systems

Question answering systems find answers to questions inside a primary collection of documents, also referred to as a corpus. Some systems also use secondary collections of documents, collections of documents, in addition to the primary collection in which the answer is to be found (Voorhees, 2003). Question answering involves finding the answer in the primary set of documents, so any answers found outside those documents can only supplement the answers from the primary set. The most popular corpus that is used to try to improve answer finding is the Internet, as seen in the following systems.

BBN's system (Xu et al., 2002) first used the original corpus to find possible answers. Then it performed a search with Google[4] with the question. When a query is entered, Google displays the top ranked documents to that query, and for each document a summary containing key words from the query is generated. Their system would rank answers by how many times the answer is in the summary of the top 100 documents from the Google query. This is a similar method to Wu et al. (2003)'s web-proofing method, where their system created a query for Google and ranked prospective answers by how many times those answers occurred compared to other prospective answers.

Lin et al. (2003) and Wu et al. (2004) took the opposite approach with their systems and rather than using the original corpus to discover the answer, they used only the Internet. They then attempted to find the answer in the original set of documents.

[4]http://www.google.com

The EagleQA system (Chen et al., 2004) extracts answers from both the web, using the Google summaries, and by a query on the text of the question using the original corpus. The Google summary answers are used later to rank the answers extracted from the primary corpus.

1.3.4 Hybrid Systems

There are many question answering systems that have a serial architecture, such as AnswerFinder (Molla and Gardiner, 2004). A serial QA system consists of modules, each designed to perform a specific task. Data is passed from one module to the next. Hybrid systems use more than one answer ranking module to rank answers. Different answers might be better found by different methods (this will be discussed further in Chapter 6) and these systems take advantage of this feature.

Hybrid systems can also take advantage of multiple corpus methods as well. An example of one of these systems is PIQUANT II (Chu-Carroll et al., 2004). PIQUANT II makes use of multiple methods of finding answers. They developed their system so that they could use any method of ranking and extracting answers and could find the answer in any corpus. For each method, it would return the top ranked answers to the system. These answers go through answer justification. In answer justification, their system ensures the answer is in the original set of documents and, if found, the answer is returned to the user.

TextMap (Echihabi et al., 2003) uses three different methods of extracting and ranking answers; knowledge-based, pattern-based and statistical-based. They use these methods on the original corpus as well as on the web, through a Google search. Then, if the top answer is found on the web, it searches the list of answers from the original corpus and, if found, it will return the top answer and the document the answer is found in. If the answer is not found, the system will return that it could not find the answer in the original corpus.

The University of Amsterdam's Quartz QA system (Jijkoun et al., 2003) (Jijkoun and de Rijke, 2004)(Ahn et al., 2004) uses multiple corpses as well as using different methods on these sources. They use a total of seven combinations of ways to extract answers. The sources include:

- Wikipedia [5] , an online encyclopedia where users can submit entries
- The Internet by using Google
- Knowledge Base created from entities from the original corpus

Their system uses these corpses, along with the original, to form a bank of answers that are all "proved" on the web, with a method similar to the one described by Magnini et al. (2002a).

All these systems have brought innovations and new developments in the field of question answering. At the University of Lethbridge, we are developing our own question answering system, with a focus on question classification and document tagging.

1.4 Contributions of this Thesis

This thesis contains the results of the research that I have done towards the development of a question answering system for the University of Lethbridge. The University of Lethbridge did not have a prior question answering system, so I investigated current question answering systems and developed a system based on successful models. The system that inspired our system was LASSO (Moldovan et al., 1999). Given time constraints, the modular design of LASSO was a good model because it allows for improvements without having to change the whole system.

The areas I felt were important to improve upon were document tagging and question classification because the LASSO system was weak in these two areas.

The following are areas of improvement:

[5]http://en.wikipedia.org

- Question classification can be performed in several ways. Lasso utilizes classification but not an extensive one. I chose the approach of analyzing a test bed of questions to manually infer extensive question categories and methods to classify questions.
- In order to aid in question answering, document tagging involves extracting knowledge from the set of documents. This knowledge can be used with the knowledge from the question classification to answer questions. There are many ways to tag documents and many different systems out there that can be used to label the information in the documents.
- I also experimented with different methods to rank the answers to choose the most likely possible answer to a question.

Implementation of our system is in Perl because of its text processing capabilities. Perl makes it very easy to take in text documents and compare strings of texts. As well, Perl has many built in features and add-ons that make communication between it and other programs simple.

Analysis of our question answering system is presented in Chapters 7 and 8.

1.5 Thesis Outline

The remaining chapters of this thesis are organized as follows:

- Chapter 2 is a summary of what information retrieval systems are, how they are useful for question answering and how I used them in our system.
- Chapter 3 is a discussion of various methods of tagging useful information in documents and how I went about implementing them in our system.
- Chapter 4 includes descriptions of the different categories I used in our system to classify questions and how our system goes about classifying questions into those categories.

- Chapter 5 describes the methods in which our system goes about extracting possible answers from documents.
- Chapter 6 outlines how our system ranks those possible answers and returns the answer that is considered the most probable.
- Chapter 7 summarizes our experience in the TREC-2004 question answering track and what we learned from it.
- Chapter 8 is an evaluation that I performed on our system to determine the overall performance.
- Chapter 9 consists of concluding remarks about my findings and my views on the future of question answering systems.

Chapter 2

Information Retrieval

Question Answering (QA) systems find the answer to a question in a collection of documents. In most cases, it is not feasible to process each document in a collection sequentially every time a new question is to be answered. An Information Retrieval (IR) system can be used to index the documents and allow a QA system to query the IR system, thus retrieving only the documents that are relevant to the question.

2.1 Indexing Documents with Inverted Files

Indexing processes documents similar to the way a textbook is indexed. At the back of most textbooks there is an index, which has a list of words in alphabetical order and the page(s) they are found on. An inverted index (Baeza-Yates and Ribeiro-Neto, 1999) for a set of documents works the same way, but with extra information. While a textbook will only index key terms for the subject, an information retrieval system will index all non stop words. Stop words (Jurafsky and Martin, 2000, page 655) are words that are not indexed because they are found to have very little meaning for the overall document. Examples of stop words are: *a, an, the, on, was,* and *is*. The set of stop words is mostly made up of determiners, pronouns, and prepositions. For each word indexed, the number of occurrences of the word in each document is saved.

Word1	D1,D2,D5,D8
Word2	D1,D2,D4,D5,D7
Word3	D1,D2,D4{2},D8
Word4	D2,D4,D6{3},D8
Word5	D3,D4,D5,D7{2}
Word6	D4,D6
Word7	D2,D3,D7

Table 2.1: Inverted File Example

To demonstrate how an inverted index stores words, here is an example with a collection that has eight documents with a total of seven different words.

- Document D1 contains {Word1, Word2, Word3}
- Document D2 contains {Word1, Word3, Word4, Word7}
- Document D3 contains {Word2, Word5, Word7}
- Document D4 contains {Word2, Word3, Word3, Word4, Word5, Word6}
- Document D5 contains {Word1, Word2, Word5}
- Document D6 contains {Word4, Word4, Word4, Word6}
- Document D7 contains {Word2, Word5, Word5, Word7}
- Document D8 contains {Word1, Word3, Word4}

The inverted file will keep track of which documents each word is in. The inverted file entries for each word are shown on Table 2.1.

Information retrieval systems can also index documents in other ways than just words. Using similar methods, an information retrieval system can create an inverted file on strings or tags, if the documents are tagged.

2.2 Queries

Once the documents are indexed, an information retrieval system allows you to retrieve documents relevant to a query. The two main ways to query the documents with an IR system are boolean and vector-space. The inverted file from Table 2.1 will be used as an example for the following sections.

2.2.1 Boolean

Boolean queries will retrieve all documents from the indexed collection that fit the query. Boolean queries are made up of basic boolean operators: *AND*, *OR* and *NOT*. In set notation, *AND* would be equivalent to the intersect of two sets, returning only elements that are common to both sets, *OR* would be equivalent to union, getting all items from both sets, and *NOT* would be elements from the first set that are not in the second set.

For instance, the query "*Word4 AND Word6*" will return the results of the intersect of the set of documents that contain Word4 and the set of documents that contain Word6. This query will retrieve documents D4 and D6.

2.2.2 Vector-Space

Vector-Space (Jurafsky and Martin, 2000, pages 647-651) queries, also called vector-cosine, are a way of ranking documents based on a query. In this ranking method, the documents and the query are represented as n dimensional vectors with each dimension representing a word in the query. The rank of the document for the query will be the cosine of the angle between the query vector and the document vector.

The query of k terms will be represented as the vector $\vec{q_k}$, where $w_{i,k}$ is the weighted term i for the word of the query k terms:

$$\vec{q_k} = (w_{1,k}, w_{2,k}, \ldots, w_{n,k})$$

Each document j will be represented as the vector $\vec{d_j}$, where $w_{i,j}$ represents the number of occurrences of term i in document j:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$

To measure the angle between vectors they should be of unit length. The process of converting vectors to unit length is called normalization. To normalize a vector, each of its dimensions are divided by the square of the length of that vector. The length of the vector is found by the formula $\sum_{i=1}^{n} w_i^2$. This formula will be factored into the final formula for this method.

For example, to normalize the vector $\vec{a}$ where $\vec{a} = (1, 1, 1)$. $\vec{a}$'s length is first calculated as $\sqrt{1+1+1}$, from the above formula. Then, normalized $\vec{a}$ will be $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$.

The similarity between two vectors can be found by the dot product of the normalized vectors. The dot product is the sum of multiplying the first vector by the transposition of the second one. This is equivalent to multiplying terms in the same dimension of each vector. The final formula for the vector cosine method for finding the angle is:

$$sim(\vec{q_k}, \vec{d_j}) = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

Documents will not be considered if they do not contain any words from the query. This formula will yield the cosine of the angle between the between the vectors.

As an example, the documents will be ranked on the query *Word3* and *Word4* and *Word6*. There are no *AND* and *OR* operations for a vector-space query. The inverted file in table 2.1 will be used as the set of documents.

In this example, the occurrences of terms will affect the weight. Since *Word6* is only used two times, we will weight it more than the other terms and the query vector will be $\vec{q} = (1, 1, 2)$. The rank for each document will be calculated with the similarity formula as follows:

$$sim(\vec{q}, \vec{d_1}) = \frac{(1 \times 1) + (1 \times 0) + (2 \times 0)}{\sqrt{1+1+4} \times \sqrt{1+0+0}} = \frac{1}{\sqrt{6}} \approx 0.40824829$$

$$sim(\vec{q}, \vec{d_2}) = \frac{(1 \times 1) + (1 \times 1) + (2 \times 0)}{\sqrt{1+1+4} \times \sqrt{1+1+0}} = \frac{2}{\sqrt{6} \times \sqrt{2}} \approx 0.57735027$$

$$sim(\vec{q}, \vec{d_4}) = \frac{(1 \times 2) + (1 \times 1) + (2 \times 1)}{\sqrt{1+1+4} \times \sqrt{4+1+1}} = \frac{5}{\sqrt{6} \times \sqrt{6}} \approx 0.83333333$$

$$sim(\vec{q}, \vec{d_6}) = \frac{(1 \times 0) + (1 \times 3) + (2 \times 1)}{\sqrt{1+1+4} \times \sqrt{0+9+1}} = \frac{5}{\sqrt{6} \times \sqrt{10}} \approx 0.64549722$$

$$sim(\vec{q}, \vec{d_8}) = \frac{(1 \times 1) + (1 \times 1) + (2 \times 0)}{\sqrt{1+1+4} \times \sqrt{1+1+0}} = \frac{2}{\sqrt{6} \times \sqrt{2}} \approx 0.57735027$$

The order in which the documents will be returned will be documents D4, then D6, then D2 and D8 (they have identical rank), and then finally D1. Documents D3, D5 and D7 will not be considered because they do not contain any of the words of the query.

Implementing a system that indexes documents is difficult to do efficiently. Information retrieval systems are implemented for these tasks.

2.3 Managing Gigabytes

We chose to use Managing Gigabytes (MG) (Witten, Muffat, and Bell, 1999) as an information retrieval system. MG is easy to customize and retrieves documents quickly. It was designed to work with large collections of documents.

MG is used in many QA systems where the developers did not develop their own IR system. Some of the groups that used MG in their QA system for TREC are:

The National University of Singapore (Yang et al., 2003), University of Edinburgh (Leidner et al., 2003), University of Southern California (Hermjakob, Echihabi, and Marcu, 2003), University of Colorado and Columbia University (Pradhan et al., 2002), ITC-Irst (Magnini et al., 2002b) and University of Girona and University Politeçnica of Catalunya (Massot, Rodriguez, and Ferres, 2003).

2.3.1 Indexing With MG

The authors of MG addressed the two main challenges with information retrieval; compressing the collection and fast retrieval. Since we have a large amount of disk space available to us, we are primarily concerned with the fast retrieval from key words. Even with the large collections we are working with, such as the AQUAINT collection of about one million documents containing 3 gigabytes of storage, MG is able to retrieve documents within seconds.

The following is a summary of how MG works, as taken from the manual pages of MG [1] :

To index the collection, MG has a program called *mg_passes* that accepts documents from *stdin*. *mg_passes* does two passes over the data:

- Pass number one creates the inverted file and generates statistics about the text and the dictionary of words.

- Pass number two compresses the inverted file with the data collected from the statistics of the first pass.

mgbuild is the driver for indexing with MG. It calls upon a program called *mg_get* to output the text of the files to *mg_passes*. *mg_passes* accepts text from *stdin* and creates a new document every time the *Control-B* character is entered. Users can modify *mg_get* to format the output from *mg_passes* any way they want.

[1] http://www.mds.rmit.edu.au/mg/man/toc.html

2.3.2 Querying With MG

Once the collection is indexed, a query can be done on the documents using *mg-query*. In *mgquery*, there are parameters that can be changed to do a boolean query or a vector-space query. There is also a parameter that will change the number of documents retrieved. For vector-space queries, it will take the top x ranked documents, while the boolean query will just take the first x number of documents that it finds, where x is the number of documents the user specified.

2.4 Other Information Retrieval Systems

We looked at other information retrieval systems that are available before deciding on MG. Here is a brief overview of some of the other systems we considered using.

2.4.1 MGPP

MGPP stands for Managing Gigabytes Plus Plus. It is based on MG with extensions that allow for deeper processing of the texts. One example of an MGPP extension is the ability to search using regular expressions. Most of the extensions of MGPP can be added to MG with a Perl script to further process the documents after they have been retrieved by the query.

2.4.2 SMART

SMART (Buckley, 1985) was used frequently as an IR system but it is not often used in current QA systems. Earlier versions of LCC's system (Pasca and Harabagiu, 2001b) used SMART, but they have changed to Lucene (Moldovan et al., 2004). The main reason we did not use SMART was that it would not run on the newer versions of Solaris and Linux Redhat. Further, the documentation is outdated, since it was last updated in 1997.

2.4.3 Lucene

Being able to use natural language to help with information retrieval has been proposed by Harabagiu and Maiorano (1999) and Mihalcea and Moldovan (2001). In the latest TREC question answering track, many systems used Lucene[2] for their information retrieval system (Cui et al., 2004) (Gaizauskas et al., 2004) (Moldovan et al., 2004) (Ageno et al., 2004). Lucene is a group of tools that lets users create their own information retrieval system, with the provided modules. Moldovan et al. (2004) notes that because of the Lucene system's ability to have a greater understanding of natural language, the passages retrieved can be more relevant to the query.

Lucene has been successful in many question answering systems. We will consider replacing MG with Lucene in later versions of our system. I only became aware of this system when it was more widely used by systems that entered the TREC-2004 or I would have already experimented with it.

2.5 Case Folding and Stemming

In an index, multiple words could be indexed as the same word in the inverted index. Case folding is when words that include upper case letters are indexed to the same entry as words without upper case letters. In stemming the words are indexed by their stems. A stem of the word is the part left after the affixes have been taken off. Affixes can take different forms and are letters that are used to modify the base form of a word. An example of this are the suffixes; ly, er, ess, ed, and ing.

MG allows you to index the documents such that case is not counted with a feature called case folding. Thus "Earth" would be indexed as the same word as "earth". This example illustrates the problem that is caused by having a case insensitive search since "earth" with a lower case letter has an entirely different meaning

[2]http://jkarta.apache.org/lucene/

than "Earth" beginning with an uppercase "E". Our queries will be on topics which, for the most part, are proper nouns. This means that first letter capitals are important and is the reason we use case sensitive searches.

Many search engines also give you an option of indexing only the stem of each word. Therefore "bank" will be indexed the same as the word "banking". This is useful because it will find more words that are related to the query term, but presents similar problems to case insensitivity.

The current version of MG uses a different stemmer than the one described in Witten, Muffat, and Bell (1999). MG uses a modified version of the Lovins Stemmer (Lovins, 1968). The Lovins Stemmer has been improved by the Porter Stemmer (Porter, 1980) which is more widely used. An example of incorrect stem handling given by MG's stemmer is mapping "dam" to the same stem as "damage" even though they have unrelated meanings.

We created two indexes with MG; one without case folding and stemming and another that included both. The methods in which we use both of these will be discussed with query creation.

2.6 Paragraph Indexing

Documents were indexed by paragraph instead of by document. A paragraph is a complete thought and most pronouns in a paragraph will represent the named entity that is already contained in the paragraph.

A paragraph that does not explicitly contain the topic will rarely involve the topic. Indexing by paragraph allows a QA system to exclude text that is unlikely to have the answer. Harabagiu and Maiorano (1999) found that there is an increase in accuracy if documents are indexed by paragraph.

2.7 Query Formation for Question Answering

Creating queries from a question is an important process that involves extracting correct terms from the question and modifying them to retrieve only the most relevant documents from the information retrieval system. The algorithm we use to extract the query requires information from the question. The question classifier in our system extracts that information. A description of query creation for question answering is presented in Chapter 4, which is devoted to question classification.

2.8 Conclusion

Information retrieval is an integral part of our question answering system. By eliminating documents that are not relevant to the question, the system only has to process documents the answer has a greater probability to be in. We are currently using MG for information retrieval but are considering using the JAVA based Lucene for future improvements.

Chapter 3

Document Tagging

The document tagging module of our system tags useful information from the passages retrieved by the information retrieval system. Our system uses information extraction techniques to extract useful information to help in question answering. I utilized different outside systems to aid in tagging documents including:

- WordNet
- OAK System
- Lingpipe

In this chapter there is information on the following ways of tagging information in documents:

- Tokenization and sentence splitting
- Part of speech tagging
- Chunked part of speech
- Word sense tagging
- Word dependency tagging
- Named entity tagging

- Coreference resolution tagging

While this is not a complete list of tags available, it is a summary of the tags most used in question answering systems. These tags deal with different depths of processing. The lowest is tokenization which separates the characters into words and punctuation. The next higher are part of speech tagging, word sense tagging, and named entity tagging. These tags give meaning to individual words. The next higher tags are chunked part of speech and coreference resolution tagging. These tags group together words that relate to each other. The highest tag we are considering is word dependency tagging which tags the relationship between words.

First, there will be an overview of the systems our program uses to tag documents. Then, each of the following sections will describe systems that can tag a document with those tags.

3.1 Systems Overview

3.1.1 WordNet 3.0

WordNet (Fellbaum, 1998) is not a system that tags documents but I utilize its features to help with tagging documents and with other modules in our system. WordNet and its features will be referred to frequently in the following chapters. WordNet is a lexical reference system with information about words and relationships between words. Words are grouped in at least one of these four categories: nouns, verbs, adjectives and adverbs. WordNet is used in most question answering systems because it is a useful tool when dealing with words.

For each word, WordNet stores each sense that the word can be used in. Each sense of a word also belongs to a synonym set (if that sense of the word has one or more synonyms). For example, the noun *car* has five senses and the synset and WordNet definition for each of them are:

car, auto, automobile, machine, motorcar 4-wheeled motor vehicle; usually propelled by an internal combustion engine; "he needs a car to get to work"

car, railcar, railway car, railroad car a wheeled vehicle adapted to the rails of railroad; "three cars had jumped the rails"

cable car, car a conveyance for passengers or freight on a cable railway; "they took a cable car to the top of the mountain"

car, gondola car suspended from an airship and carrying personnel and cargo and power plant

car, elevator where passengers ride up and down; "the car was on the top floor"

In each synonym set, the glossary definition is specified for each sense of the word. Each synonym set can be identified by a synset ID.

Another useful relation between words that WordNet includes is hyponym and hypernym for words. A hyponym for a word is a set of things that are instances of that word. For example, the hyponym set for sense one of car will include the many different types of cars such as:

- ambulance
- station wagon
- bus
- cab
- convertible
- coupe
- cruiser
- hardtop

- hot rod
- jeep
- limousine

Words within the hyponym set of a word could also have a hyponym set, for instance *berlin* is a type of limousine with a glass partition between the front and back seats and will be in the hyponym set for *limousine*.

A hypernym set is the opposite of a hyponym set. Sense one of car has a hypernym set of motor vehicle, which is a type of wheeled vehicle, which is a kind of transport, which is an instrumentation, which is an artifact, which is an entity. These sets can give relationships between two words and are useful in question answering.

With hypernym sets and hyponym sets a hierarchy is formed, with hyponym set being more specific and hypernym being less specific. A part of the hierarchy tree formed by these relations for sense one of car is in Figure 3.1.

3.1.2 OAK System

OAK System (Sekine, 2002) was developed at New York University and can tag documents in many different ways. The current version was developed in June 2001.

The OAK System has the ability to tag documents in the following ways:

- Sentence Splitter
- Tokenizer
- Part of Speech
- Chunker
- Named Entity

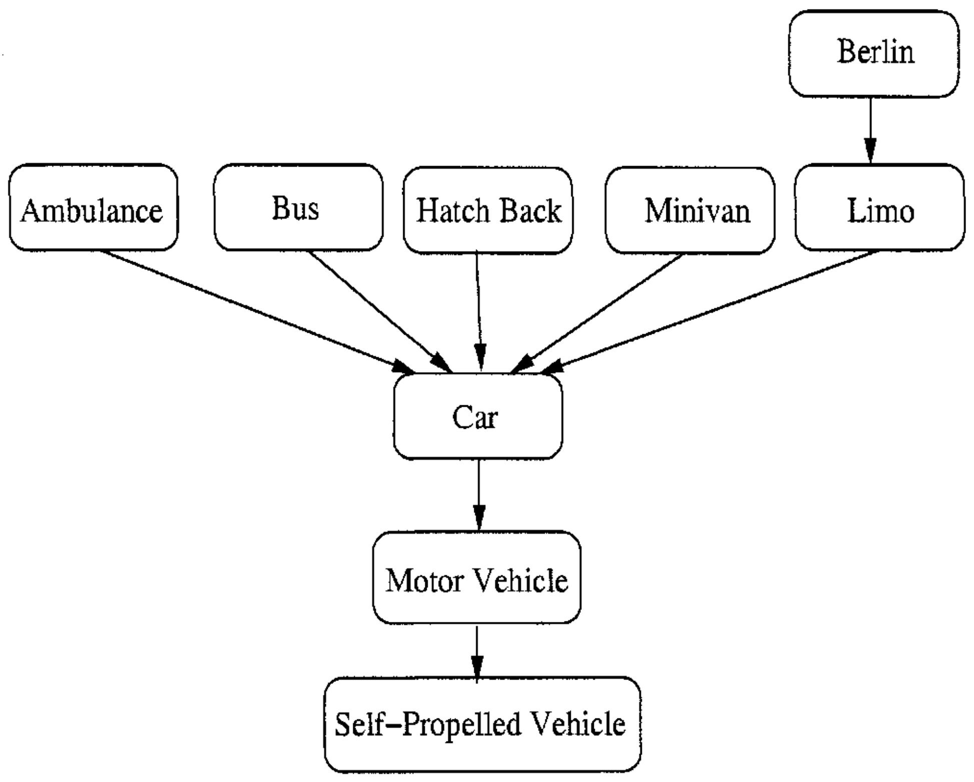

Figure 3.1: Hierarchy Tree from WordNet for Car

3.1.3 Lingpipe

Lingpipe[1] was developed by Alias-i, Inc [2]. The latest version (1.0.7) became available in August 2004.

Lingpipe can tag documents in the following ways:

- Tokenizer
- Sentence Splitter
- Named Entity
- Coreference Resolution

[1]http://www.alias-i.com/lingpipe/
[2]http://www.alias-i.com/

3.2 Tokenization and Sentence Splitting

Before text is tokenized and split into sentences, it is just a string of characters. Tokenization is splitting a string of characters into lexical elements such as words and punctuation (Jurafsky and Martin, 2000, page 298).

Sentence splitting separates the words and punctuation into their separate sentences (Jurafsky and Martin, 2000, page 180). This involves a system probabilistically determining if certain punctuation, that can be used to finish a sentence, is in fact used to end the particular sentence. For example, a period can be used in an acronym and can also be used to end a sentence, or to be more complicated a sentence can end with an acronym with the last period performing both.

This is the first step before further processing can be done to the documents. Both Lingpipe and Oak System use these techniques before further tagging documents.

3.3 Part of Speech

Each word in a sentence is classified as a Part Of Speech (POS) that depends on the way the word is being used. For instance, the word *fax* can be used as a noun (*Did you receive that fax I sent you?*) or as a verb (*Could you fax me that report?*)

Manually tagging a collection of documents, or even a single document, with these tags would be very time consuming. There are many systems available that can tag documents with part of speech with fairly high accuracy.

To be consistent, systems use sets of universal tags for parts of speech. I am using the 48 tags of the Penn Treebank POS tag set (Marcus, Santorini, and Marcinkiewicz, 1994) because this Treebank was used to train the OAK system. Tables 3.1 and 3.2 include a list of the Penn Treebank tags and examples of words that can be tagged with them.

For example, consider a sentence such as:

Tag	Description	Example
CC	Coordinating conjunction	*and, but, or*
CD	Cardinal number	*1, 2, two, 44*
DT	Determiner	*a, the, an*
EX	Existential *there*	*there*
FW	Foreign word	*moi, coupe, carpe*
IN	Preposition/subord. conjunction	*in, on, by*
JJ	Adjective	*red, mean, good*
JJR	Adjective, comparative	*faster, closer, taller*
JJS	Adjective, superlative	*fastest, closest*
LS	List item maker	*3, 1, Two*
MD	Modal	*should, can, may*
NN	Noun, singular or mass	*frog, dog, lamp*
NNS	Noun, plural	*frogs, dogs, lamps*
NNP	Proper noun, singular	*CNN, Mary*
NNPS	Proper noun, plural	*Carolinas*
PDT	Predeterminer	*all, both*
POS	Possessive ending	*'s*
PRP	Personal pronoun	*I, she, you*
PP$	Possessive pronoun	*their, your*
RB	Adverb	*slowly, never*
RBR	Adverb, comparative	*slower*
RBS	Adverb, superlative	*slowest*
RP	Particle	*up, off*
SYM	Symbol (mathematical or scientific)	+,%
TO	*to*	*to*
UH	Interjection	*um, ah, oops*
VB	Verb, base form	*sit*
VBD	Verb, past tense	*sat*
VBG	Verb, gerund/present participle	*sitting*
VBN	Verb, past participle	*sat*
VBP	Verb, non-3rd ps. sing. present	*sit*
VPZ	Verb, 3rd ps. sing. present	*sits*
WDT	*wh*-determiner	*which, that*
WP	*wh*-pronoun	*what, who*
WP$	Possessive *wh*-pronoun	*whose*
WRB	*wh*-adverb	*how, where*

Table 3.1: Penn Treebank POS Tagset with Punctuation

Tag	Description	Example
#	Pound sign	#
$	Dollar sign	$
.	Sentence-final punctuation	.?!
,	Comma	,
:	Colon, semi-colon	; : ...
(	Left bracket character	
)	Right bracket character	
	Straight double quote	
‘	Left open single quote	‘
“	Left open double quote	“
’	Right open single quote	’
”	Right close double quote	”

Table 3.2: Penn Treebank POS Tagset with Punctuation continued

> A fractal is a pattern that is irregular, but self-similar at all size scales; for example, a small patch of ground may have the same general appearance as a larger patch or even a huge area seen from high above.

This sentence with POS tags would be:

> A/DT fractal/NN is/VBZ a/DT pattern/NN that/WDT is/VBZ irregular/JJ ,/, but/CC self-similar/JJ at/IN all/DT size/NN scales/NNS ;/: for/IN example/NN ,/,a/DT small/JJ patch/NN of/IN ground/NN may/MD have/VB the/DT same/JJ general/JJ appearance/NN as/IN a/DT larger/JJR patch/NN or/CC even/RB a/DT huge/JJ area/NN seen/VBN from/IN high/JJ PP above/IN ./.

The two most popular POS taggers are the maximum entropy tagger (Ratnaparkhi, 1996) and the Brill tagger (Brill, 1994).

The maximum entropy tagger uses probabilities to tag the document with the set of tags that are most likely to be correct. These probabilities are learned though supervised machine learning techniques. In supervised machine learning, the machine learning system is given correct data, and the system derives rules or probabilities to make decisions from these rules. This tagger uses a decision tree to see all the possible tags for a sentence and finds the most probable tagging of the sentence as a whole.

The Brill tagger, also know as transformation-based tagging (Jurafsky and Martin, 2000, page 118), uses supervised machine learning as well, but learns sets of rules instead of probabilities. First it tags each word with the most probable tag for that word, and then it goes over the passage, applying the most probable set of rules for the situations.

3.3.1 OAK System

For POS tagging, our system is using the OAK System which uses a method similar to the Brill tagger, but has 13% fewer errors (Sekine, 2002). I used POS tags in patterns I developed for finding answers.

3.4 Chunked Part of Speech

Chunked part of speech is grouping words with certain parts of speech into noun phrases, preposition phrases and verb phrases. It is also referred to as a shallow parse since it is done with one pass. Ramshaw and Marcus (1995) outline a transformation-based way of tagging chunked part of speech. This machine learning method learns rules for whether a word belongs in a noun phrase, verb phrase, or preposition phrase, given the part of speech tags of the word and the words that are already in these types of phrases.

Before being tagged with chunked part of speech, the sentence looks like:

> After the announcement ceremony at the Smithsonian, Mrs. Clinton traveled to Baltimore where she announced a project to repair outdoor monuments in Baltimore, including the Francis Scott Key monument.

After getting tagged with chunked part of speech the sentence will look like:

[PP After/IN] [NP the/DT announcement/NN ceremony/NN] [PP at/IN] [NP the/DT Smithsonian/NNP] ,/, [NP Mrs./NNP Clinton/NNP] [VP traveled/VBD] [PP to/TO] [NP Baltimore/NNP] [ADVP where/WRB] [NP she/PRP] [VP announced/VBD] [NP a/DT project/NN] [VP to/TO repair/VB] [NP outdoor/JJ monuments/NNS] [PP in/IN] [NP Baltimore/NNP] ,/, [PP including/VBG] [NP the/DT Francis/NNP Scott/NNP Key/NNP monument/NN] ./.

Li and Roth (2001) used this shallow parsing for question answering, instead of a deeper syntactic parse. They found that in certain situations, such as when lower quality text is used for extracting answers, a system using a shallow parse can be more effective and flexible at answering the questions. An example of lower quality text is when the text was not edited for spelling and grammar. I designed our system for newswire documents, which are considered high quality texts, so it should be beneficial to include a syntactic parse.

We use chunked part of speech for the question classification and for tagging the documents, for reasons that will be explained later.

3.4.1 OAK System

OAK can perform a shallow parse of a document using chunked POS with a method similar to Ramshaw and Marcus (1995). Its shallow parse forms noun phrases and verb phrases with only one pass, using transformation-based rules similar to the Brill tagger.

3.5 Word Sense Tagging

Each word in WordNet has multiple senses for the different ways the word can be used. Word sense disambiguation is the process of determining in which sense a word is being used. Once the system knows the correct sense that the word is being used, WordNet can be used to determine synonyms. This is useful for seeing if a word from the question is associated with a word in the passage by being in the same synonym set.

The University of Lethbridge has used word sense disambiguation in our lexical chainer for our text summarizer (Chali and Kolla, 2004). It creates a list of all the words in the document. Then it compares the WordNet entries for their glossary, synonym set, hyponym set and hypernym set for any matches for each sense of each word. The sense that is more connected to the other words in the document is said to be the sense it is being used in.

Using this method, our system can tag each word that is contained in a lexical chain with the WordNet synset ID. The WordNet synset ID can be used to get a list of all the words belonging to that synset. This method of tagging will become useful in answer ranking, which will be described in greater detail in Chapter 6.

3.6 Word Dependencies Tagging

To discover word dependencies, a syntactic parse is used. Syntactic parsing is analyzing a sentence using the grammar rules which were used to create it. English grammar is a context free grammar that has many rules.

One method to tag word dependencies is by using the Collins parser (Collins, 1996) to get a statistical syntactic parse of the passages. These probabilities are found using supervised machine learning. The probability is the chance that two words are dependent, given certain variables including part of speech and distance. The Collins parser requires all the sentences to be tagged with part of speech before it tags the document with word dependencies.

The following is an example of a question parsed with the Collins parser:

```
(TOP is (SBARQ is (WHNP What What/WP ) (SQ is (VP is
is/VBZ (NP-A name (NPB name the/DT name/NN ) (PP of
of/IN (NPB airport the/DT airport/NN ) ) ) (PP in
in/IN (NPB Dallas Dallas/NNP Ft./PUNC. ) ) ) ) ) )
```

Pasca and Harabagiu (2001a) demonstrated that with the syntactic form you can see which words depend on other words. There should be a similarity between the words that are dependent in the question and the dependency between words of the passage containing the answer.

3.7 Named Entity Tagging

Our question classification module determines which type of entity the answer should be. In the introduction, named entities were introduced and defined as terms that refer to a certain entity. For instance, *Calgary* refers to a certain *CITY* by the name of *Calgary*, and *$12* refers to a certain quantity of money. The different classes of named entities our system recognizes are dependent on which named entities get tagged by the named entity tagger.

There are many ways to tag different named entities. One involves pattern matching from a list of examples of the entity. For instance, the entity *CITY* can be tagged by comparing an entity to a list of all cities. This will require some kind of look up table because running through the whole list for every new word will be time consuming.

Another method of tagging certain named entities is by pattern matching using regular expressions. Where the named entity is a quantity (e.g. percentage), the pattern "*NUMBER%*" or "*NUMBER percent*" could be used to discover entities that are percentages.

When our system classifies questions, it also discovers the answer type of the question. This answer type is often a named entity, and if that named entity is tagged, possible answers will be easier to extract with the answer extractor.

3.7.1 OAK System

OAK System has 150 named entities that can be tagged. They are included in a hierarchy. They are found either by pattern matching from a list of examples of entities or by regular expressions. For instance, *CITY* is best found with a list of cities, since that is almost the only way to tell if an entity is referring to a city. Appendix A outlines all the 150 named entities that OAK System currently tags.

Since OAK tags all entities at the same time, some entities that can be more than one thing will only be classified as one of them. For instance, *Paris* could be the name of a *CITY* or the name of a *PERSON*. OAK will only tag it as a *PERSON*. Using the data that OAK provides, I easily created a tagger that can tag a phrase with more than one answer type tag.

An example of a chunked part of speech and named entity tagged sentence is:

> [NP The/DT shuttle/NN] [NP ⟨SPACESHIP Discovery/NNP ⟩] [VP is/VBZ scheduled/VBN to/TO dock/VB] [PP with/IN] [NP ⟨SPACESHIP Mir/NNP ⟩] ⟨DATE later/RB [NP this/DT week/NN ⟩] and/CC [VP retrieve/VB] [NP astronaut/NN ⟨MALE_FIRSTNAME Andrew/NNP ⟩ ⟨MALE_FIRSTNAME Thomas/NNP ⟩] ,/, [NP the/DT last/JJ] [PP of/IN] [NP seven/CD] [NP ⟨NATIONALITY American/NNP ⟩] [VP to/TO work/VB] [PP on/IN] [NP the/DT station/NN] [PP in/IN] [NP the/DT ⟨YEAR_PERIOD last/JJ three/CD years/NNS ⟩] ./.

3.7.2 WordNet

As discussed previously, WordNet has sets for each word called hyponyms which are words that are examples of that word. The example that was given was that the hyponym set for car has different kinds of cars. This hyponym list can be loaded as a list of examples and a system can be made to tag these examples inside the document, similar to how OAK system tags NEs.

To use this method, our system should first know what word's hyponym list is going to be used to tag the document. Knowing the NE our system is looking for will allow our system to tag only the entities relevant to the question. These entities will be discovered with answer classification. The hyponym list is then extracted and compared to the words of the document, and the entities that are in the list are tagged as possible answers.

3.7.3 Lingpipe

Lingpipe can only tag three types of named entities which are:

- Person
- Location
- Organization

Our method of question answering requires as many named entities to be tagged as possible. Thus, our system does not use Lingpipe to tag named entities. Later in this chapter, how we are using Lingpipe in our system is discussed.

3.8 Coreference Resolution

Knowing which terms are referring to the same entities is very helpful in question answering (Ageno et al., 2004). There are different tasks when resolving the references in a document.

In the following passage, *Francis Scott Key* is referred to by his full name and also by his last name, *Key*.

> Visiting Baltimore, Hillary Rodham Clinton speaks Monday in front of the *Francis Scott Key* monument, which commemorates *Key's writing of "The Star-Spangled Banner"* in 1814. Her visit kicked off "Save

> America's Treasures," Mrs. Clinton's four-day history tour and preservation pledge drive that ends on Thursday in Seneca Falls, N.Y., at a celebration of the 150th anniversary of the first American women's rights convention.

Being able to resolve this could help the system discover that *Francis Scott Key* wrote *"The Star-Spangled Banner"*.

In this example, *spider veins* is referred to in a list:

> If *spider* and varicose *veins* are being injected and stripped and tied off and sealed shut, how does a leg still get enough blood?

In this passage *spider veins and varicose veins* are used such that the *and* makes *veins* modified by both *spider* and *varicose*.

One problem that arises is that not every instance of someone's name represents that person. A person's name can be referring to one of the things they have made, as in this example:

> Even a minor Rothko on paper, not to mention a *Picasso* or a Miro, *generally tops $250,000*.

Here *generally tops $250,000* is not a fact about *Picasso*, but rather a fact about his paintings.

Another part of coreference resolution is pronoun resolution, which is assigning which entity a pronoun is referring to. An example of how pronoun resolution can help in question answering is in the following passage:

> *Clinton* ran for president in 1992 accusing Bush of coddling China's leaders, but in 1994 *he* dramatically changed *his* position, declaring that *his* administration would no longer link trade and human rights decisions.

This passage is referring to *Clinton* and the pronouns *he* and the two *his*'s are referring to him. The question, "*Who dramatically changed their position on linking trade and human rights?*", would be easier to answer if our system could link the first *his* to *Clinton*.

3.8.1 Lingpipe

Lingpipe resolves references to the named entities; person, location and organization. Lingpipe gives each person, location or organization in the document an ID number and each reference to that person, location and organization will have the same ID number.

An example of a passage, before it is tagged with Lingpipe:

> When George Clooney leaves NBC's "ER" at the end of next season, he will rejoin Les Moonves, who as president of Warner Bros. TV signed Clooney to the "ER" cast. Moonves is now president of CBS Entertainment, which will be Clooney's new home under a two-year development deal. It calls for him to appear in two TV movies and to serve as executive producer of a new series.

That passage after being tagged with Lingpipe is:

> ⟨sent⟩When ⟨ENAMEX id="0" type="PERSON'⟩George Clooney⟨/ENAMEX⟩ leaves ⟨ENAMEX id="1" type="ORGANIZATION"⟩NBC⟨/ENAMEX⟩'s "ER" at the end of next season, ⟨ENAMEX id="0" type="MALE_PRONOUN"⟩ he⟨/ENAMEX⟩ will rejoin ⟨ENAMEX id="2" type="PERSON"⟩ Les Moonves ⟨/ENAMEX⟩, who as president of ⟨ENAMEX id="3" type ="ORGANIZATION"⟩Warner Bros. TV⟨/ENAMEX⟩ signed Clooney to the "ER" cast.⟨/sent⟩ ⟨sent⟩Moonves is now president of ⟨ENAMEX id="4" type="ORGANIZATION"⟩CBS Entertainment⟨/ENAMEX⟩, which will be Clooney's new home under a two-year development deal.⟨/sent⟩

> ⟨sent⟩It calls for ⟨ENAMEX id="0" type="MALE_PRONOUN"⟩him⟨/ENAMEX⟩ to appear in two TV movies and to serve as executive producer of a new series.⟨/sent⟩

In this passage, *George Clooney* is given the reference ID *0*. Lingpipe resolves the pronouns *he* and *him* as a reference to *George Clooney*.

This passage also illustrates one of the problems that Lingpipe has with tagging names. *George Clooney* is referred to in this passage two times by his last name, but Lingpipe failed to tag these references as a person. As well, Lingpipe does not always resolve every pronoun as in the example, *It* is not resolved.

Our system only uses Lingpipe for pronoun resolution because of this problem. Before documents are tagged by OAK system, our system tags documents with Lingpipe and replace all pronouns with the entity they are representing.

The above passage will be changed to:

> When George Clooney leaves NBC's "ER" at the end of next season, George Clooney will rejoin Les Moonves, who as president of Warner Bros. TV signed Clooney to the "ER" cast. Moonves is now president of CBS Entertainment, which will be Clooney's new home under a two-year development deal. It calls for George Clooney to appear in two TV movies and to serve as executive producer of a new series.

3.8.2 WordNet

WordNet can be used to find different forms of a name. WordNet contains many different names of people, places and organizations, and includes a synonym set for each that includes alternate forms for each of these entities. This feature of WordNet can be used to determine if a word is referring to an entity that appeared earlier in a passage.

For instance, Mars has a synonym set of *(Mars, Red Planet)*. In the following passage the entity *Mars* is referred to as both:

> Eventually, if the US government does not appear ready to do so soon enough, Mars Society president Robert Zubrin has said the society might end up sending a privately financed mission of human exploration to the red planet.

Coreferencing will help later in our system, when possible answers will be matched to patterns with entities from the question. It will also help for document retrieval because our system can retrieve documents for each term from the question that represents those terms.

3.9 Conclusion

Tagging documents allows for information from the words of the document to be labelled and used. Our system uses these tags to extract and rank possible answers. The list of tags provided in this chapter is not a complete list of all possible tags. Investigation into new tags will help improve the overall performance of our system.

Chapter 4

Question Classification

There is no limit to the type of questions that can be asked in an open-domain question answering system. Classifying questions is a way to handle a group of similar questions in a similar way, rather than handling each question individually.

Lehnert (1986) classified questions by conceptual categories. There are thirteen categories which are:

Causal antecedent *"How did James Dean die?"*

Goal orientation *"What was the purpose of the Manhattan project?"*

Ennoblement *"How did you write your thesis?"*

Causal consequent *"What happened after president Bush went into office?"*

Verification *"Did the Yankees win the World Series in 1981?"*

Disjunctive *"Did president Bush do a good or bad job?"*

Instrumental/procedural *"How do you get from Calgary to Lethbridge?"*

Concept completion *"Who owns CNN?"*

Exceptional *"Why can't ostriches fly?"*

Judgmental *"Should health-care be privatized?"*

Quantification *"How tall are the Rocky Mountains?"*

Feature specification *"What breed of dog is Scooby Doo?"*

Request *"Can you hand me that book?"*

These categories make up all the types of questions that could be asked. Questions should be answered based on which category they are in. For instance, a verification question requires a yes or no answer, while a quantification question requires a quantity as an answer.

Analyzing the question is crucial in finding the answer because the question contains the only clues available to find the answer. If information is incorrectly extracted from the question there is almost no chance to get a correct answer, but, if information is incorrectly extracted from a passage containing the correct answer, there is still a chance of finding the correct answer (Hermjakob, 2001).

Classifying a question can narrow down the entities that could be possible answers. We classify questions depending on what type the answer of the question is. Answer types are, for the most part, named entities. A named entity (NE) is a term for a specific type of entity. Examples of NEs are; people's names, organizations' names, dates, names of places and quantities. Knowing which type of information the question is asking for is a large step toward answering the question.

For instance, in the question, "In what city were the 1988 Olympics held?" the answer type is NE CITY. The possible answers can be narrowed down to only the cities found in the retrieved documents. In this case, the possible answers from the boolean query of "1988 Olympic" on the AQUAINT collection are: *Seoul, Los Angeles, Lillehammer, Calgary, Atlanta, Barcelona, Sydney, Mexico City, Washington* and *San Juan*. Since *Calgary* is the answer to the question, if a CITY is picked at random, there is a 1 in 10 chance of getting the correct answer. Systems attempt to improve this chance by getting clues from the question, and using them to rank the answers.

4.1 Corpus Questions

If we are to derive a method to extract information from questions, we first need a set of questions. The set of questions should be large enough for the classifications and categories to be meaningful. If we only use 10 questions to create classifications, our system will be less likely to classify as many questions than if we use a set of a million test questions.

The National Institute of Standards in Technology (NIST) has had a question answering track in TREC from 1999 to 2004, and has included 2,791 questions since then. We used these questions for the purpose of finding different categories and ways to classify questions, as well as ways for our system to go about answering them.

The format of the question answering track changes somewhat from one year to the next. These changes provided us with a variety of questions.

TREC-8 (Voorhees, 1999) and TREC-9 (Voorhees, 2000) had a simple format. Systems were given fact questions that either had terms or a short statement as answers. These questions are called factoid questions.

Examples of factoid questions are:

How hot is the core of the Earth?

How long would it take to get from Earth to Mars?

What is the name of the second space shuttle?

How old is the sun?

In TREC-10 (Voorhees, 2001) and TREC-11 (Voorhees, 2002), NIST added a list task, with questions that required a fixed number of answers. Factoid questions were also included these years.

Examples of list questions with a set number of answers are:

What are 3 residences of Queen Elizabeth II of the United Kingdom?

Name 22 cities that have a subway system.

List 10 countries where motorcycles are produced.

In TREC-12 (Voorhees, 2003), the definition question type was introduced. Definition questions require as much unique and important information about a topic as possible. The importance of information was judged by NIST. In TREC-12, it was specified if a question was a list, definition or factoid question.

Examples of definition questions are:

Who is Aaron Copland?

What is a golden parachute?

Who is Alberto Tomba?

What is Bausch & Lomb?

In TREC-13 (Voorhees, 2004), the format of how the questions were presented was changed. Instead of giving a basic list of questions, they grouped questions by the target of the question. The target of the group of questions is the theme of the questions. For instance, most of the targets in TREC-13 were people's names. The target is given first, and then the questions that are about the target are listed after it. In the questions, the target is often referred to by a pronoun. The format of definition questions also changed, and they are now referred to as "other" questions. Other questions require a list of unique facts about the target that have not already been given as answers to previous questions of the target, and are just stated as *"Other."*

The following are questions for the target of agouti:

What kind of animal is an agouti?

What is their average life span?

In what countries are they found?

Other

The information TREC provides about questions is useful to know, but our system should be able to handle questions, even if the TREC classification and question target is not given. All the questions from TREC, except those of TREC-13 (Voorhees, 2004), do not have to be reformatted if the target is already contained in the question. I wrote a program to change the questions' text from TREC-13, where the target appears as a pronoun. For the pronouns *he, she, it,* and *they*, our program changes the pronoun to the target, and the pronouns that signify possession, *his, her, its,* and *their*, are changed to the target, followed by an *'s*. The other questions are reformatted to the format of a definition question, i.e. "*Who is X?*", or "*Who was X?*", if the target is a person, and "*What is X?*" otherwise.

Examples of the reformulation of questions for the target of *agouti* are:

What kind of animal is an agouti?

What is an agouti's average life span?

In what countries is agouti found?

What is an agouti?

The variety of questions from 1999 to 2003 (factoid, list and definition) with varying levels of difficulty, provides many options for categorizing questions. These TREC questions provide only a sample of the number of questions that can be asked of a question answering system. Thinking outside of TREC questions, and defining categories for the set of all possible TREC questions (even if only a few can be classified as such), will provide a large domain of questions that our system can answer.

4.2 Question Normalization

Before classifying the questions, our system changes some of the questions into a standard form that will be easier to classify. These changes are performed using regular expressions in Perl, but will be explained in algorithms to facilitate understanding the methods. I developed these normalizations because they will allow for more questions to be classified and handled similarly.

4.2.1 "'s" Meaning "is"

Questions can be asked with *'s* after the question word meaning *is*. An example of this from the TREC corpus is:

- Where's Montenegro?

The *'s* is changed to an *is*. The above example will be changed to *"Where is Montenegro?"*

4.2.2 "What" Factoid Normalization

There are many different ways to say the same thing. In TREC-9, they experimented with different forms of the same question. For example:

- Name a film in which Jude Law acted.
- Jude Law was in what movie?
- Jude Law acted in which film?
- What is a film starring Jude Law?
- What film was Jude Law in?
- What film has Jude Law appeared in?

These questions all require the same information, so they should all be treated the same. As well, they should all be classified as *What* questions, and more specifically, as *What-Film* questions. Question normalization is performed by changing the form of the question so that it gets classified into the correct category.

Our system will handle questions that involve the word *which* as a *what question*. This means, all questions that begin with *Which*, *In which* and *Of which* are changed to *What*, *In What* and *Of what*.

Questions, like the above examples, *"Jude Law acted in which film?"* and *"Jude Law was in what movie?"*, that do not start with a question word will be hard to classify later as a *what question* sub-category. For easier classification, these questions are changed to *"What film was Jude Law acted in?"* and *"What movie was Jude Law in?"*, which are not in proper English, but our system will answer this form better.

For this method, our system will first put a *What* at the beginning of the question, followed by the second half the question, forming *What movie* in these examples. Then, *was* is added followed by the first half of the question, resulting in *What movie was Jude Law acted in?*.

Examples of questions this algorithm works on are:

- Musician Ray Charles plays what instrument? *to* What instrument was Musician Ray Charles plays?
- Ray Charles is best known for playing what instrument? *to* What instrument was Ray Charles is best known for playing?
- The corpus callosum is in what part of the body? *to* What part of the body was the corpus callosum is in?
- Boxing Day is celebrated on what date? *to* What date was Boxing Day is celebrated on?

Also, for any question starting with a preposition followed by the word *what* or *which*, the preposition is extracted. Another quick question reformulation is extracting *the name of* from questions that start with *What was the name of* or *What is the name of*.

Examples of these are:

- What is the name of the airport in Dallas Ft. Worth?
- What is the name of the chart that tells you the symbol for all chemical elements?
- What is the name of Ling Ling's mate?

- What is the name of the unmanned space craft sent to Mars by NASA?
- What is the name of the Chief Justice of the Supreme Court?

4.2.3 "What" Definition Normalization

In TREC-2003, there was a standard form for definition questions, while in TREC-2004 participants were just given the target. In earlier TREC question answering tracks there were no definition questions, but there were questions that asked for why someone was famous, and questions requiring a definition of what something was. We considered these questions to be definition questions for our system, and normalize all questions that wanted to know why someone was famous, or to define some target.

The normalized form is "*What is X?*", where X is the target, or "*Who is X?*", where X is the target.

Examples of pre-normalized *who-definition questions* are:

- What is Francis Scott Key best known for?
- What is Colin Powell best known for?
- What is Betsy Ross famous for?
- What is Giorgio Vasari famous for?
- What is D.B. Cooper known for?
- What is Archimedes famous for?
- What is Jane Goodall famous for?

These questions will be changed to:

- Who is Francis Scott Key?
- Who is Colin Powell?

- Who is Betsy Ross?
- Who is Giorgio Vasari?
- Who is D.B. Copper?
- Who is Archimedes?
- Who is Jane Goodall?

Examples of pre-normalized *what-definition questions* are:

- Define thalassemia.
- What does ciao mean?

These are changed to:

- What is thalessemia?
- What is ciao?

4.2.4 "What" List Normalization

The list questions from TREC-10 and TREC-11 included the number of entities to be included in the list of answers. This will be helpful in returning answers for these questions, so we extract the number, and reformat these questions to a similar format to other list questions. For each of these questions, the number is passed on to be used when the answer list is returned.

There were three patterns that the list questions were in:

- List [NUMBER]
- Name [NUMBER]
- What are [NUMBER]

For these questions, these beginnings are taken out and are replaced with *What*. Examples of normalization of these questions are:

- Name 10 auto immune diseases. *to* What auto immune diseases?
- What are 3 currencies Brazil has used since 1980? *to* What currencies Brazil has used since 1980?
- Name 5 diet sodas. *to* What diet sodas?
- What are 4 United States national parks that permit snowmobiles? *to* What United States national parks that permit snowmobiles?
- List 16 companies that manufacture tractors. *to* What companies that manufacture tractors?

4.3 Questions Categories

Our system puts questions into one of six categories:

- When Questions
- Who Questions
- Where Questions
- Why Questions
- How Questions
- What Questions

Notice that these do not include all the questions stems, since *Which, Whom* and *Name* are not included in these categories. Any question that does not include one of the first five stems (Whom is considered as Who) is considered a *what question*.

Questions are first classified into these categories by simple pattern matching to see which question stem is applicable.

Once the questions are categorized, there are other rounds of classifications, but they are different for each stem. These rounds will be more complicated, and the categories will be more specific to the type of answer expected by the question.

Classifying questions aids our system in finding the answer type of the question, as well as helping to create the query in which the documents will be retrieved from our information retrieval system.

Our system classifies questions using rules that I derived through observation. Supervised machine learning can also be used to classify questions, using methods similar to those outlined by Hermjakob (2001) Li and Roth (2002). To do this, you start with a set of questions that have already been classified, then you use machine learning to determine which patterns in the questions lead to certain classifications.

When using machine learning, there needs to be a set of example data for the rules to be trained. This involves knowing which parts of the question are clues to how to classify them certain ways. When I manually classified the questions, I was able to discover these kinds of clues. Machine learning may pick up on things it compiles as rules, but are not intuitively correct if looked at by a human. With the clues I learned from manually learning rules, I could more successfully train a system for classifying questions with machine learning.

4.3.1 When Questions

The *when questions* are made up of 250 questions from the corpus and are in two sub-categories outlined in Table 4.1, with examples in Appendix B. Both types of *when questions* can be found by simple pattern matching.

When questions will always be asking for a *DATE*, but there are a few that are only asking for a day, and not a full year. The ones asking for a day will be asking for a special day that might appear on the same day each year, like a birthday, or a day that has a rule for when it is, such as Labour Day. These types of dates are tagged by the OAK tagger. The distinction between the two will occur in the next module which is used for extracting answers.

4.3.2 Who Questions

Out of the 374 *who questions*, there are three types outlined in Table 4.2, with examples in Appendix B.

Who Definition

Who definition questions have the target passed directly to the answer extractor to extract useful facts about the target.

Who List

The list questions require more than one person. The answer type is *PERSON*, which is given to the answer extractor. As well, if a number of elements is given, then that is also passed to the answer ranking module.

Who Factoid

For these questions, this module just passes to the answer extractor the information that the answer type will either be the named entity person, or the named entity organization.

Question Type	Total	Answer Type
When-Year	234	DATE with YEAR
When-Day	16	DATE without YEAR

Table 4.1: When Question Categories

4.3.3 Where Questions

Most *where questions* require a type of location. There are other *where questions* that will ask for a certain college or university someone went to. They are outlined in Table 4.3, with examples in Appendix B.

Where School

For these questions, the answer extractor will extract schools as possible answers.

Where Location

These questions all have a location as their answer type, which includes:

- CITY
- PROVINCE
- COUNTRY
- GEOGRAPHICAL_LOCATION

These types of locations form a hierarchy. A *CITY* is in a *PROVINCE*, which is in a *COUNTRY*, which is in a *GEOGRAPHICAL_LOCATION*. The *where questions* that involve one of these entities can narrow down which entities can be possible answers. For instance," *Where is Belize located?*", involves the country of *Belize* so when discovering the answer, it could only be a *GEOGRAPHICAL_LOCATION*,

Question Type	Total	Answer Type
Who-Definition	83	FACTS
Who-Factoid	283	PERSON or ORGANIZATION
Who-List	8	PERSONS

Table 4.2: Who Question Categories

whereas, "*Where is Prince Edward Island?*", which involves a province, will either be a *GEOGRAPHICAL_LOCATION* or a *COUNTRY*.

The list of possible named entities will be passed to the answer extractor.

4.3.4 Why Questions

Our system does not handle *why questions*, as outlined in Table 4.4. These questions require reasons, which are not entities our system currently recognizes.

4.3.5 How Questions

The *how questions* have the second highest number of sub-categories, and are outlined in Table 4.5. *How questions* make up 337 questions from the corpus. Most how questions are easy to classify and are of the form, "*How X*", where *X* gives clues about what kind of entity the question is asking for.

The answer type for most of the *how questions* is straightforward. The exception is the *how many questions* because they require a count of a certain entity.

How Many Questions

These questions start with the phrase *How many* and after it comes the entity needed to be counted. Examples of *How many* questions are:

- How many rooms does the Las Vegas MGM Grand Hotel have?
- How many times a day do observant Muslims pray?

Question Type	Total	Answer Type
Where-Location	185	LOCATION
Where-School	3	UNIVERSITY

Table 4.3: Where Question Categories

- How many dimples does a regulation golf ball have?
- How many mph do you have to go to break the sound barrier?
- How many terms was Dwight D. Eisenhower president?
- How many American deaths were there in the Korean war?
- How many republics made up the Soviet Union?
- How many tentacles do squids actually have?
- How many muscles are there in the human body?
- How many layers of skin do we have?
- How many people visited Disneyland in 1999?

These questions have a pattern of where the answer is found. The entities to be counted from this list of questions are *rooms, times a day, dimples, mph, terms, deaths, republics, tentacles, muscles, layers* and *people*. For these questions this entity needs to be found and kept for the answer extraction module.

To extract the entity, first extract *How many* from the start of each question, then tag the question with a shallow parse. The reason we take off the *How many* first is because most taggers and parsers are trained on documents without question words and because of that, they do not do a very good job of tagging questions (Hermjakob, 2001). The last noun of the first noun phrase of each tagged question will be the entity that needs to be counted.

In the above example the entities will be, respectively: *rooms, times, dimples, mph, terms, deaths, republics, tentacles, muscles, layers* and *people*.

Question Type	Total	Answer Type
Why	9	REASON

Table 4.4: Why Question Categories

Question Type	Total	Answer Type
How-Large	16	AREA or VOLUME
How-Late	2	TIME or TIME_PERIOD
How-Accurate	1	PERCENTAGE
How-Many	123	NUMBER
How-Distance	50	DISTANCE
How-Often	5	TIME PERIOD
How-Long	16	DISTANCE or TIME_PERIOD
How-Much	35	MONEY or VOLUME
How-Temp	7	TEMPERATURE
How-Fast	13	SPEED or TIME_PERIOD
How-Old	17	AGE
How-Death	31	METHOD OF DEATH
How-Reason	22	REASON

Table 4.5: How Question Categories

This entity is passed to the answer extractor, to extract possible answers.

Rest of How Questions

Examples of the rest of the *how questions* and the patterns to classify them are in Appendix B.

The rest of the *how questions* will be answered with their corresponding named entities. These would be passed to the answer extractor to extract possible answers.

4.3.6 What Questions

The *what questions* total 1,633 questions and are made up of two major types. Two hundred and eight-five of these questions are classified by pattern matching and, for the rest of questions, the question focus is used to classify the questions. The *what questions* that are easily classified are in Table 4.6, with examples in Appendix B, and can be classified by simple pattern matching. The harder to classify questions that are found by discovering the focus of the question will be discussed later.

Question Type	Total	Answer Type
What-Definition	200	FACTS
What-Acro	37	INSTITUTE or ACRO or PHRASE
What-Verb	48	THING

Table 4.6: What Simple Question Categories

What Definition

These questions are handled similarly to the who definition questions, where the target is given to the answer extractor.

What Acro

These questions are either looking for the acronym for a certain entity, or wanting to know what entity a acronym stands for.

What Verb

For these questions, the type of answer is ambiguous. For example, there are many types of things that can be invented, discovered and eaten. These questions can be answered using patterns in the syntactic parse of the documents. Once we add a syntactic parse of the documents to our system, we will attempt to handle these questions.

4.3.7 What Questions Classified by Focus

The *what questions* that are not easily classified by pattern matching can be classified by discovering the focus of the question. The focus of the question is a clue about what type of entity the answer will be. In the question, "*What country is the leading exporter of goats?*", the question focus is *COUNTRY*. The question focus is discovered by looking at chunked parts of speech tagged questions. These focuses are then matched to a named entity, and that will be considered the answer type of the question.

Some named entities are not tagged. A full list of the 1,348 questions that can be classified by focus can be found in Tables 4.7 and 4.8, with examples in Appendix B. Note that some of the answer types specify that the answer type is a *type*. This is because there are two kinds of some entities: a proper named entity (e.g. Name of a certain bridge), or a type named entity (e.g. a certain type of bridge). This distinction needs to be made to improve the system, and not extract the wrong types of entities.

The importance of question focus when classifying questions is discussed in (Ferret et al., 2001). Once the questions are all tagged with chunked parts of speech, patterns are used to extract the question focus. These patterns were discovered by manually observing where the focus occurred in the questions that are harder to classify out of the 2,791 corpus questions. The question focus is always a noun phrase, and the patterns are clues as to which noun phrase in the question will be the focus. An example of a pattern is that a focus usually appears after an *'s* signifying possession. Examples of this are in the corpus questions:

1. [NP What/WP] [VP is/VBZ] [NP Carlos/NNP] [NP the/DT Jackal/NNP] [NP 's/POS real/JJ name/NN] ?/.

2. [NP What/WP] [VP is/VBZ] [NP Cassini/NNP space/NN probe/NN] [NP 's/POS destination/NN] ?/.

3. [NP What/WP] [VP are/VBP] [NP Burger/NNP King/NNP] [NP 's/POS gross/JJ sales/NNS] [NP today/NN] ?/.

4. [NP What/WP] [VP is/VBZ] [NP Eileen/NNP Marie/NNP Collins/NNP] [NP 's/POS occupation/NN] ?/.

5. [NP What/WP] [VP is/VBZ] [NP Alberto/NNP Vilar/NNP] [NP 's/POS nationality/NN] ?/.

Question Type	Total	Answer Type
What-City	89	CITY
What-Country	53	COUNTRY
What-Province	6	PROVINCE
What-Flower	5	FLOWER
What-Bird	6	BIRD
What-Tree	1	TREE
What-Date	25	DATE
What-Year	56	YEAR
What-Name	94	PERSON
What-Continent	10	CONTINENT
What-Population	22	POPULATION
What-Company	14	COMPANY
What-Instrument	9	INSTRUMENT
What-Color	16	COLOR
What-Nationality	8	NATIONALITY
What-Film	16	FILM
What-State	24	STATE
What-River	10	RIVER
What-Animal	9	ANIMAL
What-County	6	COUNTY
What-Person	56	PERSON
What-Sport-Team	12	SPORT_TEAM
What-Band	12	BAND
What-Percentage	7	PERCENT
What-Persons	10	PERSONS
What-Countries	26	COUNTRIES
What-Cities	7	CITIES
What-Companies	4	COMPANIES
What-Books	5	BOOKS
What-Songs	4	SONGS
What-States	4	STATES
What-Movies	3	MOVIES
What-Value	3	MONEY
What-Distance	11	PHYSICAL_EXTENT
What-University	17	UNIVERSITY

Table 4.7: What Questions Classified by Focus

Question Type	Total	Answer Type
What-Water	6	BODY OF WATER
What-Location	5	type of LOCATION
What-Branch	4	BRANCH OF SERVICE
What-Expectancy	4	YEAR_PERIOD
What-Currency	8	CURRENCY
What-Lake	3	LAKE
What-Job	8	OCCUPATION
What-Planet	5	PLANET
What-Language	8	LANGUAGE
What-War	5	WAR
What-Disease	6	DISEASE
What-Dog	7	DOG
What-Sport	6	SPORT
What-Focus-Undefined	431	UNKNOWN

Table 4.8: What Questions Classified by Focus continued

The question focus of each is: 1. real name, 2. destination, 3. gross sales, 4. occupation, 5. nationality. From these question focuses, our system can use pattern matching to determine the answer type of the question. For instance, the focus *real name* is going to be the name of a person, which, for our system, is the answer type *PERSON*.

One of the obvious places for the focus is at the beginning of the question. The noun phrase immediately after *"What"* is often the answer type of the question. Examples of this are:

1. [NP What/WP actress/NN] played/VBD [NP Betsy/NNP] [PP in/IN] "/" [NP Betsy/NNP] [NP 's/POS Wedding/NN] "/" ?/.

2. [NP What/WP country/NN] [VP is/VBZ] [NP the/DT leading/VBG producer/NN] [PP of/IN] [NP rubber/NN] ?/.

3. [NP What/WP country/NN] [VP celebrates/VBZ] [NP Guy/NNP Fawkes/NNP Day/NNP] ?/.

4. [NP What/WP president/NN] [VP created/VBD] [NP social/JJ security/NN] ?/.

5. [NP What/WP college/NN athletic/JJ teams/NNS] [VP are/VBP nicknamed/VBN] [NP the/DT Cougars/NNP] ?/.

6. [NP What/WP Chinese/JJ provinces/NNS] [VP have/VBP] [NP a/DT McDonald/NNP] [NP 's/POS restaurant/NN] ?/.

7. [NP What/WP foods/NNS] [VP can/MD cause/VB] [NP allergic/JJ reactions/NNS] [PP in/IN] [NP people/NNS] ?/.

8. [NP What/WP film/NN] introduced/VBD [NP Jar/NNP Jar/NNP Binks/NNP] ?/.

9. [NP What/WP country/NN] [VP is/VBZ] [NP Horus/NNP] [VP associated/VBN] [PP with/IN] ?/.

This method of finding the question focus often gets exact answer types, such as in examples 2,3,7,8 and 9. Examples 1 and 4 are of the type that has to derive that an actress and a president are people, so the answer type will be *PERSON*. Examples 5 and 6 demonstrate that list questions have a focus that contains a plural noun (NNS), usually at the end of the noun phrase (NP).

A complete list of focus extracting patterns, where X represents the focus of the question, is:

- [NP What (type or kind or breed)] [PP of] [NP *X*]
- [NP What *X*]
- [NP What] [NP *X*]
- ('s or ') *X*]
- [NP What] [VP (is or was)] [NP *X*]

- [NP Name *X*]
- [NP Name] [NP *X*]

For each of these questions, the answer type is passed to the answer extractor. Examples of questions classified by focus are presented in Appendix B.

4.3.8 "What" Focus Undefined

There is a subset of 431 questions that have a focus that is unique, and is not a named entity that is currently tagged by OAK. For these questions, the hypernym set of WordNet is used to try to get a list of entities that are of the type of the focus.

4.4 Query Creation

The more information that is known about the question, the more information that will be known about what words are likely to appear in a document with the answer.

For instance, Moldovan et al. (1999) shows that the question focus is often not found in a document with the answer. When the focus is a *state, city, country, nationality* or *date*, the answer will appear as that specific entity, and those focus words will not necessarily appear in the documents. This kind of information should be thought about when creating the query to retrieve documents from the information retrieval system.

4.4.1 Problem of Query Creation

When creating a query a problem arises when the search is not specific enough, and the information retrieval system retrieves too many documents. For instance, for the question "*How much money does the United States Supreme Court make?*", if *United States* is extracted, the query will retrieve 208,581 documents, which will contain too many unrelated terms of the same named entity as the answer type of the question. We consider 208,581 too many documents because, even if one fourth of the documents contain the answer type of the question, that means there will be over 52,000 possible answers, with only a few of them related to the actual question.

There are three choices to solve this problem. One choice is to process all 208,581 passages. A second choice is to change the search from boolean to vector space, and to a certain number of documents. The third choice is to add words to the boolean query so fewer documents are returned. Processing all the documents is a poor alternative since there is a lower chance of finding the answer. If a vector space search is done, with topics such as *United States*, it will retrieve the documents with the most occurrences of *United* and *States*. This will eliminate documents based on the *United States*, instead of the question, so there is a chance that the documents eliminated will be question related. The third choice is to find more words from the question to narrow the search. This will require development of a heuristic similar to (Moldovan et al., 1999) to discover which words will be more likely to be found in the document containing the answer. This method of going back and changing the query, based on the documents retrieved, is referred to as feedback loops and is discussed by Harabagiu et al. (2001).

There are some sentences that do not contain proper nouns or dates, so other words should be considered to narrow down the documents retrieved. Moldovan et al. (1999) discusses how verbs are unimportant to the query and should be left out. In WordNet 3.0, there are relations between verbs and nouns. This is useful when creating a query because you can expand a verb to include a noun representation of it, along with all the synonyms of it. In the question, "*Who owns CNN?*", *owns* can be found to be derived from *owner, proprietor* and *possessor*. This will be helpful if the answer is found in a sentence similar to "*Ted Turner, owner of CNN, was at the White House today.*"

Using other sets in WordNet, such as hypernym, hyponym and synonym, the words from the question can be used to create a greater set of words that are associated with the question, and to retrieve documents relevant to the question.

These are ideas that we will consider for each category of questions, but each type might not use all these methods.

4.4.2 Our Method for Factoid and List Questions

We query the documents using three methods:

- Expanded Boolean Query
- Important Word Boolean Query
- Expanded Vector-Space Query

Expanded Boolean Query

We take all the nouns, verbs and adjectives from the question that are not stop words, and find synonyms for each of them, using WordNet. We use these synonyms to form a query, where each word in the synonym set for a word is separated by *OR*, and each synonym set is separated by *AND*. Our system does not include the words contained in the question focus when creating the expanded query.

For example, if a question has two words that are extracted, say *Word1* and *Word2*, and the synset for *Word1* is *S11* and *S12* and the synset for *Word2* is *S21*, *S22* and *S23*, the boolean query will be: "(S11 OR S12) AND (S21 OR S22 OR S23)".

These queries are given to the collection we indexed with stemming and case folding such that all the forms of each word are used. This query ensures that all documents retrieved should be somehow related to the question. These queries are often too restrictive and retrieve no documents.

Important Word Boolean Query

If the first method of querying retrieves no documents, then our system forms a boolean query with the proper nouns and dates from the question. The query is done on the collection we indexed without stemming and case folding because proper nouns and dates are normally not stemmed.

This will usually retrieve the document with the answer to the question, but will also retrieve many documents unrelated to the question.

Expanded Vector-Space Query

If there are no proper nouns or dates in the question, or the previous method retrieves too many documents (over 10,000), or no documents, then a vector-space search is performed.

4.4.3 Our Method for Definition Questions

For definition questions we do a query just on the subject to be defined. If it is the name of a person (from a who-definition question), then we query the collection without stemming and case folding, and if it is not a person, we query the collection with stemming and case folding.

4.5 Conclusion

This module is where all the processing of the question is done. We have developed six main categories, based on the question word that is being used (*When, Who, Where, Why, How, What*), and further subcategories for each of them. We also developed hand drawn rules with which to classify questions. These rules were based on the trends in the 2,791 TREC questions we have compiled. Our system is, therefore, based on the TREC questions, rather than the domain of all questions that are able to be asked.

Our system can only answer three out of the thirteen types of questions discussed in the opening of this chapter; concept completion, quantification and feature specification. Being able to classify the other types of questions will allow our system to attempt to answer a greater domain of questions.

As well, with the knowledge gained from creating the methods for classifying manually, we can develop a supervised machine learning approach to question classification.

Chapter 5

Answer Extraction

For questions there are often several forms the answer can take. For instance, the question "*Where is Ayer's Rock?*" can be answered in more than one way using the following passage:

> Drive the Gunbarrel Highway, west of Ayer's Rock in the Australian outback.

Possible answers derived from this passage include:

- in the Australian outback
- Australian outback
- Australia

All of these answers are correctly describing where Ayer's Rock is. The first one tells the relative location of *Ayer's Rock*, and the second and third are also correct. When a non relative answer is given, it is assumed that it is in the answer location i.e. the answer *"Australian outback"*, is taken to mean *"in the Australian outback"*.

An example where the relative location has to be given is:

> Commonwealth Games gold medallist sprinter Nova Peris-Kneebone will be the first torch bearer and it will also be carried around the base of Uluru, formerly known as Ayer's Rock near Alice Springs, by representatives of the traditional owners.

The only correct answer from this passage is *near Alice Springs* because if the answer was given as *Alice Springs*, it would be assumed that it is inside *Alice Springs*, which is not the correct answer.

This problem needs to be kept in mind when extracting answers from passages. Our system, for the most part, extracts named entities we tagged as possible answers. This requires the system to extract the whole answer, and not just part of the answer.

Our method of extracting answers is different for list and factoid questions which require extracting named entities, and answers to definition questions that are extracted using patterns.

5.1 List and Factoid Questions

We use the OAK system to tag the documents with named entities, as discussed in chapter 3. It tags the named entities inside angle brackets with the type of named entity included just inside the first angle bracket, e.g."*⟨NE_TYPE ENTITY⟩*"

Since the answer types and question types are known from the question classifier, and the documents are tagged with both shallow parse and named entities, extracting the information from the documents will use patterns to extract the named entities associated with the answer type. The questions do not always have one named entity associated with their answer type, so the patterns need to reflect this as well. For some questions, the answer is not tagged and will be extracted with just patterns from the documents.

The answers for certain types of questions can also be extracted with patterns rather than our approach of extracting answers using the answer type. These patterns can be learned by using machine learning techniques (Ravichandran and Hovy, 2002). Many groups have used a similar technique in their systems to discover patterns for finding answers. ((Roussinov, Ding, and Robles-Flores, 2004), (Tanev, Kouylekov, and Magnini, 2004), (Echihabi et al., 2003), (Nyberg et al., 2003), (Wu et al., 2003) and (Prager et al., 2003))

5.1.1 "How Many" Questions

How many questions are answered using patterns and not named entities since these questions are looking for a count of a certain entity. I discovered that answers will appear frequently in the tagged documents in the pattern "*[NP NUMBER ENTITY]*". For the question, *"How many hexagons are on a soccer ball?"*, the entity to be counted is *hexagons*, so the pattern it is trying to match is *[NP NUMBER hexagons]*. This pattern is found in the following passage:

> [PP After/IN] [NP all/DT] ,/, [NP a/DT buckyball/NN] [NP 's/POS structure/NN] [PP of/IN] [NP 12/CD pentagons/NNS] and/CC *[NP 20/CD hexagons/NNS]* [VP is/VBZ] [ADVP just/RB] [PP like/IN] [NP that/DT] [PP of/IN] [NP a/DT classic/JJ soccer/NN ball/NN] ./.

This module will extract *20 hexagons* and send that as a possible answer, along with the passage it is found in, to the answer ranker.

For these questions, WordNet can also be used to get synonyms for the entity that is being counted. For the question, *"How many floors are in the Empire State Building?"*, the answer entity *floors* can be represented different ways. The WordNet synset for *floors* includes *floor, level, storey* and *story*. For this question *story* can be used to find the answer in the following passage:

> The Empire State Building climbed to an unthinkable height of *102 stories* in that city four years later. Two bridge projects were begun across belligerent San Francisco Bay in the early 1930s.

This method is a simple version of coreferencing.

5.1.2 Names of People

The OAK system tags people in four ways:

- PERSON
- LASTNAME
- FEMALE_FIRSTNAME
- MALE_FIRSTNAME

This presents a problem because the OAK tagger tags everything at once from a list of names. Some *LASTNAMES* are missing from the list of tags, and other names are also names of cities, such as the name *Paris*.

The answer extracting module needs to extract the full name of the person when it is available. Some examples of problems and their patterns for solutions are:

> [NP ⟨CITY DETROIT/NNP ⟩] :/: [NP ⟨MALE_FIRSTNAME Juan/NNP ⟩ Gonzalez/NNP] [VP was/VBD] [ADVP back/RB] [PP in/IN] [NP the/DT starting/VBG lineup/NN] [PP after/IN] [VP missing/VBG] [NP three/CD games/NNS] [PP because/IN of/IN] [NP a/DT sore/JJ foot/NN] but/CC [VP may/MD be/VB sidelined/VBN] [PP after/IN] [VP aggravating/VBG] [NP it/PRP] [SBAR while/IN] [VP running/VBG] [NP the/DT bases/NNS] [ADVP when/WRB] [NP he/PRP] [VP hit/VBD] [NP a/DT triple/JJ] ./.

In this example, the name *Juan* is followed by the untagged last name *Gonzalez*. This can be fixed by the pattern "*⟨[A-Z_]NAME [A-Za-z]NNP ⟩ [A-Za-z]/NNP*". Notice that it will also get *LASTNAME*, as well as the other two types. Some people have last names that are usually considered to be first names:

> [NP P.S./NNP ⟨MALE_FIRSTNAME Kevin/NNP ⟩ ⟨MALE_FIRSTNAME Ryan/NNP ⟩] [NP forwards/RB a/DT ⟨CITY Sacramento/NNP Bee/NNP ⟩] [VP clipping/VBG describing/VBG] [NP a/DT paper/NN sign/NN] [PP on/IN] [NP the/DT men/NNS] [NP 's/POS room/NN wall/NN] [PP in/IN] [NP the/DT state/NN] [NP Capitol/NNP] [NP 's/POS

⟨FACILITY Legislative/NNP Office/NNP ⟩ Building/NNP] :/: "/" [VP Please/VB wash/VB] [NP your/PRP$ hands/NNS] [PP before/IN] [VP touching/VBG] [NP legislation/NN] ./. "/"

This will present the pattern of "⟨ *NAME_TYPE NAME/NNP* ⟩ ⟨*NAME_TYPE NAME/NNP* ⟩". There is also the named entity of *PERSON* that gets tagged by OAK system like:

[PP Despite/IN] [NP some/DT overdone/VBN moments/NNS] ,/, [PP like/IN] [NP the/DT remake/NN] [PP of/IN] [NP ⟨ PERSON Van/NNP Morrison/NNP ⟩] [NP 's/POS "/" Brown-Eyed/NNP Girl/NNP] ,/, "/" [ADJP most/JJS] [PP of/IN] [NP the/DT album/NN] [VP turns/VBZ] [NP Alexakis/NNP] [NP '/POS ups/NNS and/CC downs/NNS] [PP into/IN] [NP tuneful/JJ ,/, broad-shouldered/JJ rock/NN] ./.

5.1.3 Dates

There are two types of date questions, one looking for a certain day that happens every year, and ones that are looking for a particular day. Some dates that are tagged by OAK do not fit into either of these categories and are considered relative dates. These include *today, this year, this month, next week* and *midnight*. These dates are not helpful in answering questions, and are eliminated right away by specifying that the dates have to include a number.

Also, answers to questions that are looking for a particular date should have a four digit year in them.

5.1.4 Quantities

OAK tags each quantity it sees as a quantity, but does not tag quantities together that are of the same measurement if the quantities are of different units. For instance, it will tag the measurement *4 foot 2 inches* as ⟨ *PHYSICAL_EXTENT 4 foot* ⟩ ⟨

PHYSICAL_EXTENT 2 inches ⟩. We can extract the full quantity if we use a pattern that extracts more than one quantity when two quantities of the same measurement are together.

5.1.5 Other Types of Questions

For the rest of the questions, possible answers are extracted by extracting the named entities associated with the answer type of the question. This is done by pattern matching with "⟨*NE X* ⟩" where *X* is a possible answer if *NE* is a named entity tag corresponding the the answer type of the question. These entities were listed in Appendix A.

5.2 Definition Questions

Fact finding questions are handled differently because the process for finding a fact is more involved than finding certain entities. In fact finding questions, there is always a topic that is being sought. If the question is "*Who is X?*" (X being a name of a person), there will be different methods for finding facts for it, compared to a non-person entity that will be phrased as "*What is a Y?*" We have chosen to implement a method of pattern finding to answer fact based questions.

5.2.1 Pattern Finding

The patterns we are using for definition questions use shallow parsing, also called POS chunking. Gaizauskas et al. (2004) implemented a similar method of using shallow parsing to find patterns for fact based questions. These fact finding patterns were determined by manually examining definition based questions from the TREC question test bed, which includes examples of facts for each question. We used the following four phases to determine the patterns our system uses to extract facts.

Manually Finding Facts

Sentences that include facts are found first, so patterns can be observed later. These fact sentences are found by forming a query from the topic, and retrieving relevant documents from our information retrieval system. NIST provides answers to past definition and other questions that can be used to form a query to try to get a specific fact. These facts came from systems that entered TREC, although some of the participating systems did not just use the AQUAINT data set, so some of the facts can not be found in that collection.

For the question "*Who is Aaron Copland?*"

His circle of friends included Picasso and Piet Mondrian, composer Aaron Copland, actors Charlie Chaplin and Paulette Goddard, and titans of U.S. industry such as the Fords and Rockefellers.

This sentence contains the fact that *Aaron Copland* is a *composer*.

And for the question "*What are fractals?*"

A fractal is a pattern that is irregular, but self-similar at all size scales; for example, a small patch of ground may have the same general appearance as a larger patch or even a huge area seen from high above.

This sentence contains the fact that a *fractal* is *a pattern that is irregular, but self-similar at all size scales.*

Tag Fact Sentences

The fact sentences are then POS chunked and patterns are observed. The example sentences POS chunked are;

[NP His/PRP$ circle/NN] [PP of/IN] [NP friends/NNS] [VP included/VBD] [NP Picasso/NNP and/CC Piet/NNP Mondrian/NNP] ,/, [NP *composer/NN* Aaron/NNP Copland/NNP] ,/, [NP actors/NNS Charlie/NNP Chaplin/NNP and/CC Paulette/NNP Goddard/NNP] ,/, and/CC [NP titans/NNS] [PP of/IN] [NP U.S./NNP industry/NN such/JJ] [PP as/IN] [NP the/DT Fords/NNP and/CC Rockefellers/NNP]

./.

[NP A/DT fractal/NN] [VP is/VBZ] *[NP a/DT pattern/NN] [NP that/WDT] [VP is/VBZ] [ADJP irregular/JJ] ,/, but/CC [ADJP self-similar/JJ] [ADVP at/IN] [NP all/DT size/NN scales/NNS]* ;/: [PP for/IN] [NP example/NN] ,/, [NP a/DT small/JJ patch/NN] [PP of/IN] [NP ground/NN] [VP may/MD have/VB] [NP the/DT same/JJ general/JJ appearance/NN] [PP as/IN] [NP a/DT larger/JJR patch/NN] or/CC [ADVP even/RB] [NP a/DT huge/JJ area/NN] [VP seen/VBN] [PP from/IN] [ADJP high/JJ] [PP above/IN] ./.

Pattern Creation

Patterns are formulated from manual observations of the tagged sentences. The first passage shows that information contained before the target, which is in the noun phrase, is a pattern to find out a fact about this target. The second passage shows that the target followed by *is* contains a fact after *is* until the stop of the sentence, in this case a semi-colon.

Our current definition patterns include, with *X* representing facts and *TARGET* representing the subject of the fact:

- [NP *X TARGET*]
- *[TARGET]* , *X* (, or . or ;)
- *[TARGET]* (is or are) *X* (. or ;)
- *X* called *[TARGET]*
- *[BEGINNING OF SENTENCE] [TARGET]*, *X*, is *X*
- *[TARGET]* and other [NP *X*]

Error Correction

Adding new patterns may lead to extracting some phrases that are not useful or not relevant at all. These phrases should be filtered out as generally as possible to both keep the applicable facts and so the rule can apply to many situations. These patterns should be applied to different topics to confirm that the patterns will work with more than a single topic.

For instance, the pattern "*[TARGET]* , *X* (, or . or ;)" can end up extracting the next element from a list. Take "$element_1$, $element_2$, $element_3$, ... *and* $element_n$" to represent any list in the set of retrieved documents. If the topic is $element_i$, then $element_{i+1}$ will be extracted with that extraction pattern. Our check for this is, if the extracted fact is just a name, then the fact is considered invalid.

All the incorrect facts might not be able to get filtered out. The hope is that they will get filtered out in the answer ranking module, which will rank the facts and only return relevant facts to the user.

5.3 Conclusion

Before the validity of an answer is calculated, they have to be extracted from the tagged documents. Using the tags from the last module, I formed regular expression patterns to be used to extract possible answers from the documents. For list and factoid questions, the NE of the answer will be used to determine which terms are possible answers. The next module is the answer ranking module, which will determine which possible answers are given as the answer.

Chapter 6

Answer Ranking

The previous module extracted possible answers from the set of documents. This module gives a score to each possible answer, and returns the one with the highest score, or the answers with the highest scores if a list of answers is required. It is possible that the corpus will not contain the answer to the question. Because of this, answers should only be given if the system is sure that the answer is correct. If a list of answers is required by the question, this module will only pass on answers that are over a certain rank. Xu et al. (2002) refer to the rank as a confidence estimation; it is the confidence level in the correctness of the answer, given the knowledge about the answer.

6.1 Question Answer Patterns

As stated in the previous chapter, many systems use lexical patterns to extract answers from the documents, in contrast to our approach of extracting named entities. Our system used extracting patterns instead to rank answers. The reason we took this approach is because we have discovered that many answers do not appear in the set of patterns. If a possible answer appears in one of the following patterns, it will be given a higher rank.

These patterns benefit from the coreference resolution our system performs with Lingpipe. When the answer of the question is being referred to in one of these patterns it may be represented by a pronoun.

6.1.1 Date of Birth and Date of Death

The date of birth and date of death of a person are sometimes put in brackets, after a person's name, e.g. "PERSON (DATE - DATE)". In this pattern, the first date represents the birth date and the second represents the date of death.

An example of a passage that contains three such patterns is:

> Elvis Presley (1935-1977), James Dean (1931-1955) and Marlon Brando (1924-) are the new men

6.1.2 "What" Location Questions

When a location is the answer type of the question, and the question contains a preposition like *in, on, from* or *near*, those words will frequently appear before the location entity that is the answer. For questions that contain *on* and *from*, *in* can also appear before the answer.

Examples of these types of questions are:

- What continent is Togo on?
- What continent is India on?
- What city is the Kentucky Horse Park near?
- What countries is Burger King located in?
- What country is Hyundai from?
- What country was Catherine the Great from?

An example of a sentence with this pattern, for the question "*What continent is Togo on?*" is:

> The president, who returned home early on Tuesday after a three-day visit to Mali and Togo *in West Africa*, agreed to undergo the most intensive medical check-up to date as a bid by the government and the African National Congress to refute damaging rumours that his health was deteriorating.

6.1.3 "When" Questions Ending in a Verb

Some questions end with a verb that represents the particular action that is being asked about. Examples of these questions are:

- When was Microsoft established?
- When was the NFL established?
- When was the USS Constitution commissioned?
- When was the first Wal-Mart store opened?
- When was Hiroshima bombed?
- When was President Kennedy shot?

For these questions, an answer is usually found in the following pattern, "*VERB in* ⟨*DATE*⟩", where *VERB* represents a verb with a stem that is a synonym of the last verb from the question.

An example of a sentence with this type of pattern for the question, "*When was the first Wal-Mart store opened?*", is:

> Supercenters, the first of which *opened in 1988*, had already transformed the $420 billion-a-year grocery business.

6.1.4 Who Action

Who questions often do not contain an action, other than the action of *being*. For example:

- Who was Khmer Rouge's first leader?
- Who were leaders of the Khmer Rouge?
- Who are the Wiggles members' names?
- Who is the Queen of Holland?
- Who is the president of the Spanish government?
- Who is the prime minister of Japan?

There are some questions that contain a physical action that are not being:

- Who wrote "Dubliners"?
- Who wrote "Hamlet"?
- Who created the literary character Phineas Fogg?
- Who founded Rhode Island?
- Who signed the Declaration of Independence from Vermont?
- Who discovered prions?

These actions will be represented in the passages that the answers are in, but might take different forms. For these types of questions, our system will only take the pattern of the whole action. For example, the question "*Who wrote "Dubliners"?*", has the action "*wrote "Dubliners"*", and will be the pattern our system uses. This pattern is similar to the previous pattern, but the action happens after the possible answer. Here is an example of a sentence the answer can be found in for the question "*Who wrote "Dubliners"*":

> In the 12 years Joyce lived in Trieste, *he wrote "Dubliners,"* "Portrait of the Artist as a Young Man" and part of "Ulysses."

Notice that this answer requires that we resolve the pronoun *he* so that our system knows it is referring to *Joyce*.

6.2 Word Association

The words in the question can give clues as to what words will be around the words of the answer. Without any further expansion of the question words, there will only be a small chance that the exact words of the question would be found around the answer. The words that are contained in the question should be represented in the passage with the answer. A question without any classification is just a "bag of words". We add information about the words by classifying the questions and tagging the words in the questions in various ways.

Words express ideas and ideas can be expressed using different words. Lexical chains (Morris and Hirst, 1991) are created when you associate words together that have a similar theme. Moldovan and Novischi (2002) discussed using WordNet to help with the associations between words. Each word in WordNet is part of a synset of words that have the same meaning. Each synset has a hypernym set, a hyponym set and a glossary definition. Each word in WordNet has a derived form as well. The derived form for a noun is a verb that is associated with the noun. For example, *celebration* is a derived form of *celebrate*. With these tools there can be a path formed from the question words to the words of the passage of a possible answer.

Our system also uses word sense disambiguation in this module. If the word from the question is associated with the sense of a word that is disambiguated, then the word from the question also fits in with a lexical chain from the passage. Our system uses lexical chains to find the sense for each word, so if the sense is found, then that word is closely related to the passage.

6.3 WordNet Glossary

For most words, WordNet has a glossary entry which contains a definition of the word. Our system extracts the glossary entry for the proper nouns in the question. For definition questions, the glossary entry for the target is used.

For factoid questions, our system tags the WordNet glossary entry for named entities, and if it contains the answer type, that answer is given a higher rank. For definition questions, the WordNet glossary entry is passed to the redundancy checker to rank facts.

For the question, *"Where is Belize located?"*, *Belize* is a proper noun and its WordNet gloss is:

> a country on the northeastern coast of Central America on the Caribbean; formerly under British control.

The gloss tagged with POS and NE is:

> a/DT country/NN on/IN the/DT northeastern/JJ coast/NN of/IN ⟨ GEOLOGICAL_REGION Central/NNP America/NNP ⟩ on/IN the/DT ⟨ GEOLOGICAL_REGION Caribbean/NNP ⟩ ;/: formerly/RB under/IN ⟨ NATIONALITY British/JJ ⟩ control/NN.

For *where questions*, *GEOLOGICAL_REGION* is an accepted answer type, and both *Central America* and *Caribbean* are acceptable answers for this question.

6.4 Target distance

One other method that is commonly used to rank answers is the distance between a possible answer and keywords from the question (Kwok, Etzioni, and Weld, 2001)(Chen et al., 2004). Our system calculates distance as the count of words and punctuation between the important words from the question and the possible answer. If there are no such entities in a question, then this method is not used to rank answers for that question.

For example, the question "*What does Kurt Vonnegut do for a living?*", has a target of *Kurt Vonnegut*. The following passage contains the answer to this question:

> Trust a crowd, says *author Kurt Vonnegut*, to look at the wrong end of a miracle every time.

The answer *author* is considered, by our system, to be one word away from the target of the question.

6.5 Redundancy Checking

A possible answer can be discovered in more than one passage. Answers that appear frequently should be given a higher rank if the retrieved documents are related to the question.

If the question requires more than one answer, each answer returned by the system should be unique. When this module returns an answer, it performs a check to see if the answer has already been included in the final answer list. This check is performed by checking whether the answer contains nouns that have already appeared in a previous answer.

For definition questions, an answer can contain more than one fact. For example for the target *Jane Goodall*, the phrase, *the British primatologist*, gets extracted. This contains the fact that she is *British*, and the fact that she is a *primatologist*. If this is the case, then the phrase that contains two or more facts is added to the list of answers, and if either of the contained facts were found in the list, their score will be added to the score of the combination fact.

Our system counts each redundancy as an occurrence, which will be used in ranking answers.

6.6 Answer Ranking Formula

Each of the previous sections discovers facts about different answers. This ranking formula was developed by reviewing which weights for the methods above yielded the highest accuracy on the corpus of questions and their answers.

6.6.1 Factoid and List Questions

Our system's answer ranking formula for factoid and list questions uses the following variables:

w_1 denotes whether the answer is found in a pattern associated with the question type. The value will be 1 if it was found in such a pattern, and 0 if it was not.

w_2 denotes how many words from the question are represented in the passage with the answer, plus 3 more points for each word that is represented by a disambiguated word.

w_3 denotes if the answer appears in the WordNet glossary for important words from the question. (value of 0 if it does not and 1 if it does)

w_4 denotes the distance between the important words from the question and the answer.

Each occurrence of an answer is given the following rank:

$$(6 * w_1) + (w_2) + (3 * w_3) + \frac{1}{w_4}$$

Each answer's final rank is the sum of the ranks of the occurrences of that answer. This means, if a particular answer appears in ten passages, then this formula is used for each passage and the scores are added together. This method of giving a higher score to answers that appear more than once is discussed in Clarke, Cormack, and Lynam (2001).

This formula was derived using a test bed of 300 questions. These questions from TREC-1999 to TREC-2004 were chosen because our system extracts the answer to them. The average chance of our system, at random, picking a correct answer to a question in this test bed is about nine percent. This means approximately one in ten extracted answers are correct. With this ranking formula, our system is able to answer this test bed correctly sixty four-percent of the time. This shows that this method of ranking is improving the chance of picking a correct answer.

6.6.2 Definition Questions

Of all our methods of ranking answers, only answer redundancy is applicable to definition questions. As stated in the previous chapter, our system will sometimes extract facts that are invalid, and we hope that invalid facts will be eliminated because they will not be repeated, and thus will have a low rank.

We rank answers to definition questions, giving them one point for repetition and four points if found in the WordNet gloss.

6.7 Answer Thresholds

When an answer is given for a question, the confidence that the answer is correct should be high. Even if the question answering system could return more than one candidate answer to the user, only the ones that have a chance of being correct should be shown (Xu, Licuanan, and Weischedel, 2003). For questions where a list of answers is required, all the answers should be correct, so the threshold should be a bit greater than that of an answer candidate.

Our system does not currently use an answer checking threshold because our method for ranking answers is inconsistent for different types of questions. The threshold needs to be set as a relative value or a constant value, and we have not yet determined an appropriate relative score for which an answer is considered to be correct or incorrect. This is a direction that requires further consideration.

Chapter 7

TREC-2004 Evaluation

The questions in the 2004 TREC question answering track were set up differently than past years of TREC question answering tracks. For TREC-2004, there was an answer target and there were questions about the target that often referred to the target by a pronoun. The questions were classified as; factoid questions that require one answer and list questions that require a non-redundant list of answers. For each target, there was also a question referred to as "other". The other questions require a list of non-redundant facts that were not already given by a previous answer. Of the three question types, we focused our efforts primarily on the factoid questions, which made up 50 percent of a system's overall score in the TREC-2004. List and other questions made up a quarter each of the overall score.

The following is a short description of the system that we used in the TREC-2004, and a discussion of the results.

7.1 Our System Overview

Our TREC system was designed to handle groups of questions. The questions appeared in this format:

```
<target id = "1" text = "Crips">
    <qa>
```

```
        <q id = "1.1" type="FACTOID">
            When was the first Crip gang started?
        </q>
    </qa>
    <qa>
        <q id = "1.2" type="FACTOID">
            What does the name mean or come from?
        </q>
    </qa>
    <qa>
        <q id = "1.3" type="LIST">
            Which cities have Crip gangs?
        </q>
    </qa>
    <qa>
        <q id = "1.4" type="FACTOID">
            What ethnic group/race are Crip members?
        </q>
    </qa>
    <qa>
        <q id = "1.5" type="FACTOID">
            What is their gang color?
        </q>
    </qa>
    <qa>
        <q id = "1.6" type="OTHER">
            Other
        </q>
    </qa>
```

```
</target>
```

For the "other" questions, all previous answers should be saved, so the system can check for redundancy.

The system we used to answer the questions on the TREC-2004 question answering track is described in Chali and Dubien (2004). The architecture of our TREC system was similar to the general system we have implemented currently, and is outlined in Figure 7.1.

First, our system separated the questions by target. For each target, it formed the query for the information retrieval system. The documents which were retrieved were tagged, and used to answer all questions pertaining to that target. The questions were then classified, and the possible answers were extracted from the set of retrieved documents, and then ranked.

7.2 Results of TREC-2004 QA Track

The results for TREC-2004 QA track, sorted by factoid, are in Tables 7.1 and 7.2. These results are available on the TREC website[1].

7.2.1 NIL Results

There were 22 factoid questions that had no answer in the documents, so the only answer accepted for those questions was NIL (given for no answer). Eleven of the runs in the TREC-2004 would have scored better on the factoid portion if they had given no answer for every factoid question. Getting 22 out of the total 230 factoid questions would give a system a score of about 0.096.

[1]http://trec.nist.gov/pubs/trec13/appendices/qa.results.html

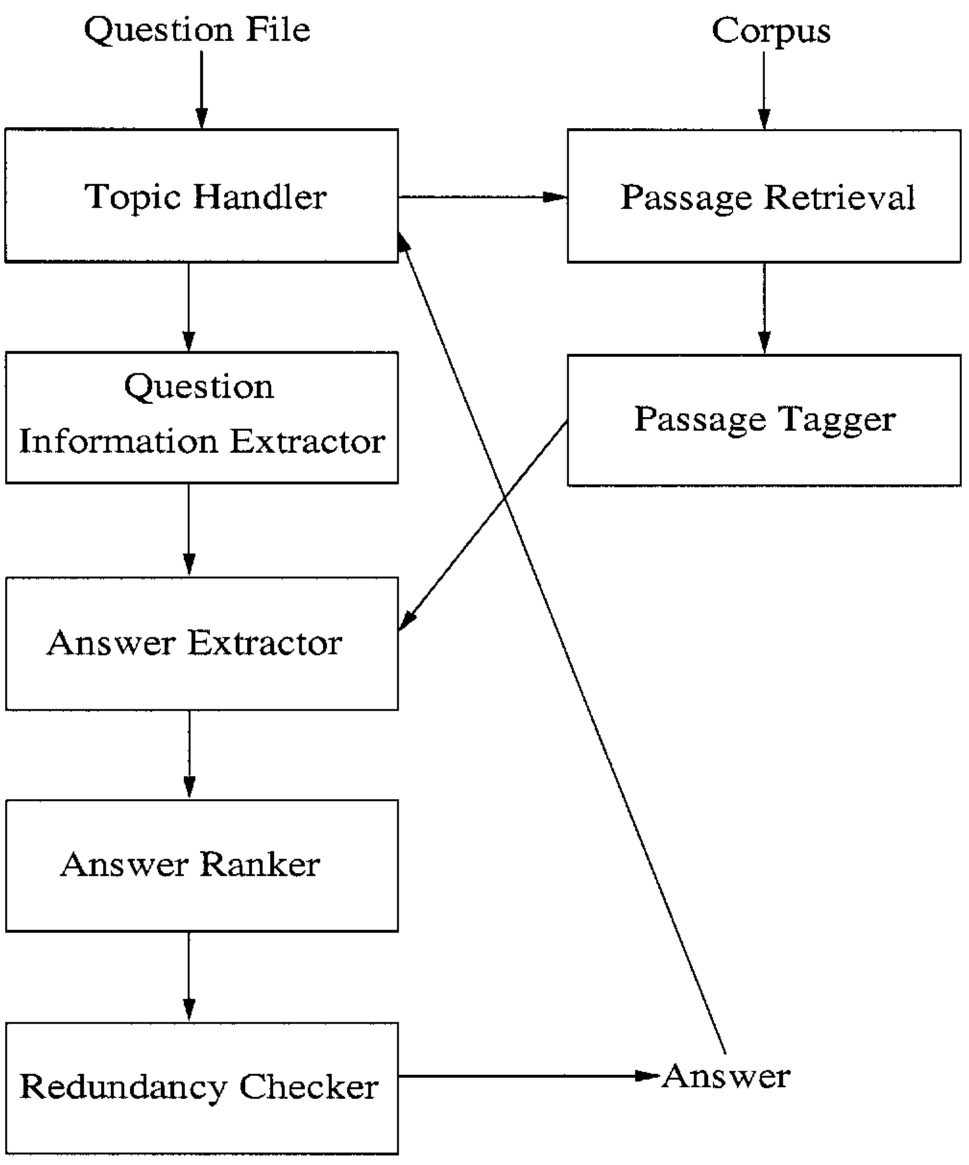

Figure 7.1: Model of Our QA System from TREC-2004

Organization	Factoid	List	Other	Final
LexiClone	0.009	0.000	0.000	0.005
University of Sheffield	0.043	0.053	0.317	0.114
University of Sheffield	0.061	0.058	0.321	0.125
University of Alberta	0.074	0.022	0.000	0.043
University of Alberta	0.074	0.022	0.000	0.043
University of Alberta	0.074	0.022	0.000	0.043
Monash University	0.078	0.021	0.003	0.045
University of Edinburgh and Sydney	0.091	0.036	0.068	0.072
University of Iowa	0.091	0.000	0.118	0.075
University of Iowa	0.091	0.000	0.130	0.078
University of Edinburgh and Sydney	0.091	0.043	0.194	0.105
Tsinghua University	0.100	0.085	0.055	0.085
Macquarie University	0.100	0.080	0.080	0.090
Macquarie University	0.100	0.080	0.080	0.090
Macquarie University	0.100	0.081	0.080	0.090
University of Lethbridge	0.104	0.024	0.023	0.064
National Central University	0.109	0.000	0.077	0.074
National Central University	0.109	0.000	0.080	0.075
Dalhousie University	0.113	0.105	0.112	0.111
Dalhousie University	0.126	0.051	0.048	0.088
University of Amsterdam, Informatics Institute	0.126	0.087	0.184	0.131
University of Amsterdam, Informatics Institute	0.126	0.085	0.207	0.136
Dalhousie University	0.130	0.052	0.049	0.090
University of Limerick	0.130	0.087	0.164	0.128
University of Amsterdam, Informatics Institute	0.135	0.094	0.210	0.144
University of Edinburgh and Sydney	0.143	0.054	0.152	0.123
Arizona State University	0.148	0.000	0.000	0.074
Universitat Politçnica de Catalunya (UPC)	0.157	0.031	0.165	0.128
Universitat Politçnica de Catalunya (UPC)	0.157	0.031	0.197	0.136
University of Limerick	0.161	0.092	0.138	0.138
CL Research	0.161	0.064	0.239	0.156
University of Limerick	0.170	0.101	0.171	0.153
National Security Agency (NSA)	0.183	0.104	0.062	0.133
The MITRE Corporation	0.183	0.131	0.129	0.157
Korea University	0.187	0.157	0.229	0.190
Korea University	0.187	0.157	0.247	0.195
University of North Texas	0.187	0.128	0.305	0.202
University of North Texas	0.187	0.127	0.307	0.202

Table 7.1: TREC-2004 QA Track Results continued

Organization	Factoid	List	Other	Final
National Security Agency (NSA)	0.187	0.098	0.355	0.207
The MITRE Corporation	0.191	0.143	0.151	0.169
University of North Texas	0.196	0.123	0.305	0.205
National Security Agency (NSA)	0.204	0.071	0.367	0.212
University of Sheffield	0.213	0.125	0.312	0.216
Korea University	0.222	0.159	0.246	0.212
ITC-irst	0.239	0.100	0.200	0.195
Fudan University (Wu)	0.257	0.141	0.367	0.256
Fudan University (Wu)	0.257	0.143	0.389	0.262
Fudan University (Wu)	0.257	0.143	0.404	0.265
ITC-irst	0.278	0.090	0.207	0.213
ITC-irst	0.291	0.103	0.207	0.223
Massachusetts Institute of Technology (MIT)	0.313	0.113	0.186	0.231
Massachusetts Institute of Technology (MIT)	0.313	0.119	0.184	0.232
IBM T.J.Watson Research Center (Prager)	0.313	0.200	0.227	0.263
IBM T.J.Watson Research Center (Prager)	0.313	0.200	0.285	0.278
Saarland University	0.339	0.111	0.181	0.243
Language Computer Corporation	0.339	0.182	0.154	0.254
Saarland University	0.343	0.096	0.145	0.232
Saarland University	0.343	0.125	0.211	0.256
National University of Singapore (Chua)	0.500	0.480	0.379	0.465
National University of Singapore (Chua)	0.600	0.485	0.460	0.536
National University of Singapore (Chua)	0.626	0.481	0.448	0.545
University of Wales, Bangor	0.643	0.258	0.000	0.386
Language Computer Corporation	0.770	0.622	0.240	0.601
AVERAGE	0.209	0.115	0.183	0.179

Table 7.2: TREC-2004 QA Track Results ctd

7.2.2 Factoid Gap

Of the systems entered in the TREC-2004 QA track, three systems achieved scores in the factoid section that were much higher than the other systems: LCC (Moldovan et al., 2004), University of Wales, Bangor (Clifton and Teahan, 2004), and National University of Singapore (Cui et al., 2004). National University of Singapore, the lowest of the three, scored almost 0.300 better than the next highest system in the factoid section. Therefore, the top three systems must use superior answer finding modules and the knowledge of what made those systems perform better on this set of questions can provide information on how to further the state of the art.

LCC

This system is the only one that uses a logical representation of both the answer passage and the question passage. They achieved more than 0.100 ranks better than the next highest system. As well, LCC has consistently developed the highest ranked question answering system in TREC since the first question answering track. Groups developing question answering systems should consider logical representation if accuracy is a key requirement.

LCC also entered a system that did not do as well. This was because they entered a question answering system that will return an answer in 30 seconds. There are no time constraints in the TREC question answering track, except that systems are required to return their answers to the questions within two weeks of NIST posting the questions.

University of Wales, Bangor

This group also submitted a unique type of system. They used knowledgeable agents (Teahan, 2003), that were derived from knowledge grids (Cannataro and Talia, 2003). They created a data base of possible questions and their answers. When a question gets asked, their system compares the question to those already

Question Type	Worst	Median	Best	UofL1
Factoid	0.009	0.170	0.770	0.104
List	0.000	0.094	0.622	0.024
Other	0.000	0.184	0.460	0.023

Table 7.3: Evaluations Results for UofL in QA Track from TREC-2004

formulated, and returns the answer to the question that most closely resembles the question asked.

This system does not use deep processing and relies on part of speech and named entity tagging. These results are surprising because, as LCC's system shows, deeper processing methods, such as logical representation work very effectively. This system might benefit from a deeper parse of the text, since a deeper parse has been successful for LCC's system.

National University of Singapore

This system uses event-based question answering (Yang et al., 2003). This year they attempted to improve their system by using word dependencies found by using a syntactic parse (Cui et al., 2004). They propose that one can use dependencies of words of the question, and a possible answer, to form a dependency path. If such a path can be formed between a possible answer and the question, their system considers the answer correct.

7.3 Our Results

Table 7.3 shows the results of our system (UofL1) in TREC compared to the worst, best and median scores.

7.3.1 What We Learned

Our experience in TREC-2004 has taught us many things about what makes a good question answering system, and the following are lessons we learned.

Lesson 1

Our system was not ready for the categories of questions that were asked. We were only able to attempt to answer 56 questions out of the 230 factoid questions because of poor classification. We only used the questions from TREC-2003 to create question categories and methods for classifying question. One example of this is that *Who* and *Where* questions made up one percent of questions from TREC-2003 but made up 25 percent of questions in TREC-2004. Our submitted system did not properly classify these questions because we primarily trained it on the TREC-2003 data.

This showed that a bigger corpus of questions will be needed, if we are to classify a greater variety of questions effectively.

Lesson 2

Our system relied too heavily on the target given for each question. We used the target to extract documents to answer the questions, not knowing that some of the targets would represent a theme, rather than the actual entity of the questions. An example of this is the questions for the last target:

```
<target id="65" text="space shuttles">
<qa>
<q id="65.1" type="LIST">
    What are the names of the space shuttles?
</q>
</qa>
<qa>
```

```
<q id="65.2" type="FACTOID">
    Which was the first shuttle?
</q>
</qa>
<qa>
<q id="65.3" type="FACTOID">
    When was the first flight?
</q>
</qa>
<qa>
<q id="65.4" type="FACTOID">

    When was the Challenger space shuttle disaster?
</q>
</qa>
<qa>
<q id="65.5" type="FACTOID">

    How many members were in the crew of the Challenger?
</q>
</qa>
<qa>
<q id="65.6" type="FACTOID">

    How long did the Challenger flight last before it exploded?
</q>
</qa>
<qa>
<q id="65.7" type="OTHER">
```

```
    Other
</q>
</qa>
</target>
```

The last three questions are talking about the *Challenger space shuttle*, rather than just *space shuttles* in general. A large number of systems in the TREC-2004 reformulate the questions to include the target. If the target is being referred to in the question, the reference would be by a pronoun. The questions can be reformulated by replacing the pronoun in the question with the target of the question. Then these systems treated the reformulated question as a new question, where the target is unknown.

Lesson 3

Our classification of list questions was poor as well, and we only attempted to answer 4 out of the 55 questions asked. Knowing how to classify and use some of the answer types available to use with OAK would have helped greatly.

Lesson 4

Our system did not extract many good facts for the "other" questions. We only found a vital piece of information for 4 out of the 64 targets (and only one in each of the 4 cases). To do better in this section, we would have to improve and develop new patterns to find facts about a target.

Developing these fact finding patterns using a full set of definition questions from all the TREC question answering tracks will help us have better patterns for TREC-2005.

Lesson 5

Our system had a very low accuracy rating, even on the questions that we did answer. We correctly answered only 8 out of the 54 questions that our system attempted to answer. This percentage (14%) will need to be improved if we hope to be over the median score (0.170) for factoid questions.

7.4 Conclusion

The TREC-2004 changed its format from previous TRECs, which provided insight into how systems perform on different types of questions. Only three out of the twenty-eight participants submitted systems that achieved a rank over 0.350 for the factoid questions. For the University of Lethbridge, as well as the rest of the systems that participated in TREC-2004, it was a chance to find and improve on shortcomings to help further develop our system.

Chapter 8

Evaluation

The TREC question answering track provides a method to rank systems on how well they answer a set of questions. NIST only observes the answers a system gives to the specific set of questions and evaluates the correctness of those answers. This provides information about the overall accuracy of the participating systems, but provides little information about where the systems went wrong. Every question answering system is made up of modules, with each module handling different tasks in answering the question. Each module has the potential to create errors, and these errors can lead to a question being improperly answered. This chapter will be an evaluation of the errors that occur in the modules of our system.

Knowing which module is the weakest link will help with knowing which modules should be improved to increase our system's overall accuracy. Moldovan et al. (2004) performed a similar evaluation of their own modules to discover how much error is produced by the individual modules of their system.

Our system has been trained on the TREC questions from 1999 to 2004. NIST provided the answers to the TREC-2004 question answering track questions. We used these answers to evaluate each of our modules for factoid, list and other questions. For this evaluation, the questions should be unknown, so I performed this evaluation with only classifications and rules that we derived from the TREC questions from 1999 to 2003.

8.1 List Questions

Our system had a very poor performance in TREC-2004. It only found 20 out of the 484 answers for the list questions. I evaluated our current system on how many answers our system can find for the list questions from TREC-2004.

8.1.1 Overall Evaluation

With the improved classification we added to our current system we are able to retrieve 149 out of the 484 answers to the questions. This means our system found 29.1 percent of the answers. This percentage is sufficient for us to start working on better redundancy checking, and returning a higher percentage of correct answers.

8.2 Other Questions

In TREC-2004, we only found 5 out of the 234 vital facts about the targets. Currently, we are not considering redundancy. Because of the poor performance our system had in TREC-2004 we would like to focus on how our system finds facts.

8.2.1 Overall Evaluation

With our updated system we were able to retrieve 34 vital facts about the targets, rather than the 5 that were found with our 2004 system. We expected this number to be higher because of the success our patterns had with the earlier years of TREC definition questions, which were similar to the "other" questions.

Many of the vital facts that were found in TREC-2004 were not found in the AQUAINT collection. They were found in other sources, and the AQUAINT collection was only used to back up the fact. This approach has been found to be very effective, and has been used by many systems (Hildebrandt, Katz, and Lin, 2004). To improve our system's ability to find facts for targets, we should investigate this multi-document approach.

8.3 Factoid Questions

We are only going to consider the 208 questions that include an answer in the AQUAINT data set. This will ensure that if a question is not answered, it is the fault of a module of our program.

8.3.1 Passage Retrieval

The passage retriever module will take in the query from the question classifier, and then use that query to retrieve relevant documents, using MG, to be passed to the passage tagger. An information retrieval system is used to eliminate unrelated documents, but, if the query is not phrased properly, the document containing the answer could be eliminated. This module will be evaluated based on whether it finds at least one document containing the answer. If a document with the answer is not found, it will be impossible to answer the question.

To do the test we retrieved the documents using the methods discussed in the question classification chapter, and checked whether the document set contained the answer. If the answer was not contained in the documents retrieved, either the query was created improperly or the documents were improperly indexed.

Out of the 208 questions, 38 of the queries failed to retrieve a document containing the answer. This is about an 18% error rate.

8.3.2 Answer Extractor

The answer extractor uses the answer type of the answer, and extracts all phrases of that type, along with the passage the phrase appeared in. All phrases and their associated passages are then passed to the answer ranker.

Evaluation of this module will be based on how many times an answer is unsuccessfully extracted when the answer is known to be in the retrieved documents. A fault in this module will be caused by an answer type that was not tagged, or by the question being improperly classified.

Out of the 170 questions that had documents successfully retrieved, with the answer, by the information retrieval system, only 88 of them had the correct answer extracted, giving an error rate of 48.3% for this module.

This error rate is very high and means that improvements in document tagging, question classification and answer extraction would greatly improve the performance of our system. Determining which of these modules produced the most error will require further evaluation.

8.3.3 Answer Ranker

The answer ranker takes in all possible answers from the answer extractor, and ranks them by the criteria discussed earlier.

The answer ranker should produce better results than random chance. We are only concerned with cases where the correct answer is in the set of answers being ranked.

Out of the 88 times the answer was successfully extracted, the correct answer was ranked the highest 66 times or 75% of the time.

8.3.4 Overall Evaluation

Our system picked the correct answer 66 times out of 208 questions, giving our system an overall accuracy of 31.7%. If we considered the set of all 230 TREC-2004 questions and assume that we answered all the 22 questions with no answer incorrectly, we achieved a rank of 0.287 (65 divided by 230). A rank of 0.287, in the factoid part of TREC-2004, is 0.078 above the average and 0.117 above the median.

We will classify the questions from TREC-2004 and use them to improve our hand drawn classifications. Our system is more prepared and we expect a large increase in accuracy for TREC-2005.

Chapter 9

Conclusion

Our approach to question answering extracts possible answers that fit the type of answer, dictated by the type of question, from the document set. Other methods use pattern matching to extract possible answers, or a logical form to try to deduce an answer. Our method of question answering creates queries, then our information retrieval system extracts the documents our system finds most relevant to the question. After the possible answers are extracted, our system uses an answer ranking formula to choose the answer ranked most probable by our system.

In this approach, we had to use different methods of processing both the question and the documents. The most important part of our system is our question classifier, which uses rules we determined by manually observing a bank of sample questions. Our system classifies questions by question type, and the type of the answer. A query is formed from the words of the question, to retrieve documents that pertain to the question.

The documents retrieved from the query are then tagged to enable our system to extract different information from the documents. Named entities are one of the types of information extracted from the documents. Some of these named entities will represent the answer type of the question. Terms tagged with the named entities, that are of the same type as the answer type, are considered possible answers and are then extracted for ranking.

Our system ranks the answers according to various criteria that I determined gave our system the best accuracy on the test set of questions as compared to other formulas I tried. This ranking includes how frequently the possible answer appears in the documents retrieved, if it appears in an answer pattern associated with a question, and if words from the question are represented in the passage with the possible answer.

Our method of question answering does not include any deep processing of the documents, and the run time is dependent on how many documents are retrieved for the question being asked. In our recent evaluation, we discovered that our system performed with about 31% accuracy on a set of 208 test questions. The average accuracy from TREC-2004 question answering track was 21% for factoid questions, with only six groups submitting systems that were above 31%. These results show that we have a relatively good accuracy compared to the other systems that participated in the TREC-2004. These systems may have been improved since then, so the TREC-2005 question answering track will show how our new system compares to newer versions of the other systems.

9.1 Future Work

We found that taggers and parsers that are not trained on questions do not work well on questions. To continue research into question answering and question classification, we would want to have trainable taggers so that we can tag questions correctly. Tagging questions correctly will lead to more accurate classifications.

With the knowledge we gained from classifying questions, we could experiment with machine learning techniques for classifying questions. We could develop patterns with machine learning to extract the question focus, and use it to classify the question. Machine learning could also be used to discover patterns in the document with which to extract answers. If we could collect a complete list of patterns we could extract possible answers using both patterns and named entities. Then our system would extract a list of possible answers that contain a better percentage of correct answers.

9.2 Perspectives

The Turing test (Turing, 1950) is a test a system should meet in order to be considered intelligent. The Turing test involves two humans and a computer. One human acts as the interrogator and goes into a separate room. He asks questions to both the computer and the other human, and has to tell which one is which. If the interrogator can not distinguish between the human and the computer, then the system has passed the Turing test.

This test can be used in many situations, and can be applied to question answering. Our approach of extracting answers to questions with named entities rather than by patterns, I argue, will pass the Turing test more often than systems that use pattern matching to extract answers.

For the question *"How far is it from Mars to Earth?"*

The following were the answers for the question [1] :

- 1894 NYT19990211.0075 -1 142 million miles
- 1894 APW19990803.0303 -1 117 million miles
- 1894 NYT19981211.0308 1 416 - million - mile
- 1894 APW19990923.0168 -1 121.9 million miles
- 1894 APW19980705.0043 1 700-million-kilometer (440-million-mile)
- 1894 NYT19991122.0091 -1 three
- 1894 XIE19990316.0235 -1 1976
- 1894 NYT19990524.0252 -1 54 million miles away

[1]http://trec.nist.gov/data/qa/2003_qadata/03QA.tasks/t12.judgments.main.txt

- 1894 NYT19990923.0315 1 249 million miles
- 1894 NYT19990923.0365 1 249 million miles
- 1894 NYT20000131.0402 1 the 190 million miles
- 1894 NYT19981210.0562 -1 about 3 feet
- 1894 APW19990923.0014 -1 262-mile
- 1894 APW19990103.0013 -1 4 million mile
- 1894 NYT20000824.0279 -1 30 percent
- 1894 APW19990803.0031 -1 219 million miles
- 1894 NYT20000324.0337 -1 more than two years
- 1894 NYT19981212.0029 1 416 million miles
- 1894 NYT19981208.0039 -1 500 miles
- 1894 XIE19981212.0306 -1 260-mile
- 1894 APW19990803.0031 -1 about 117 million miles
- 1894 NYT19981212.0029 1 416-million-mile
- 1894 APW19990429.0018 -1 geologically dead
- 1894 XIE19980813.0089 -1 one scientist
- 1894 APW19990429.0249 -1 geologically dead

The first number is the question number, the second is the document ID, the third is a -1 for wrong and 1 for correct, and the last is the answer given. If we consider the wrong answers; *geologically dead, one scientist, 1976, 30 percent* and *more than two years*, are mistakes that a human would not make. A human knows a *How far* question will be asking for a distance and will give a distance as an answer, and will normally not make a mistake about what type of entity the question is asking for. These wrong answers might not have been discovered using pattern matching, but if the question has been classified correctly and a system only considered named entities associated with the answer type of the questions, these types of wrong answers would not be given.

The Turing test for question answering is, "Can a computer find the answer in the set of documents as well as a human can?" Erbach (2004) did a study on how well human subjects do in question answering, and found that under no time constraints, humans achieved an average accuracy of 95 percent. This test was performed on three human participants who were asked 200 questions from the CLEF question answering track [2] .

Ninety-five percent accuracy is higher than the best systems that entered TREC question answering track, in which the only time constraint is that systems have 15 days to answer the entire test bed of questions. With the human participants' answers, from the study, they found that the accuracy decreased to around 40 percent for 40 seconds, and around 20 to 30 percent for 30 seconds. Comparing these to the 0.339 that was achieved by LCC in their 30 second time constraint system, the LCC system actually performed better.

There is a need to get fast and accurate answers to many questions that are asked every day. Question answering systems are being developed to fill that need. However, it is not yet apparent whether they will become commercially viable in the same way as search engines such as Google and Yahoo.

Google represents the current generation of search engines (handling 250 million searches per day in 2003[3]). The next generation will provide exact answers, instead of finding documents that contain the answers you need. This will be feasible when question answering is fast enough to retrieve accurate answers from huge

[2]http://clef-qa.itc.it/2004/guidelines.html
[3]http://searchenginewatch.com/reports/article.php/2156461

collections such as the Internet.

Question answering systems currently are being tested on mainly factoid, definition, and list questions. For question answering systems to be the next generation of search engines, they would need to handle most questions that a user can think of. The future of question answering will be to handle as many situations as possible, and make finding information on the web, or anywhere else, easier. TREC question answering track will keep challenging systems to work towards this goal by increasing the difficulty and variety of questions. Systems need to achieve higher accuracy before question answering research will be expanded to include timing and usability issues.

Appendix A

OAK System 150 Named Entities

Tag	Example
NAME	
PERSON	Bill Clinton, Satoshi Sekine
LASTNAME	Clinton, Sekine
MALE_FIRSTNAME	Bill, George
FEMALE_FIRSTNAME	Mary, Catherine
ORGANIZATION	United Nations, NATO
COMPANY	IBM, Microsoft
COMPANY_GROUP	Star Alliance, Tokyo-Mitsubishi Group
MILITARY	The U.S. Navy
INSTITUTE	the National Football League, ACL
MARKET	New York Exchange, NASDAQ
POLITICAL_ORGANIZATION	
GOVERNMENT	Department of Education, Ministry of Finance
POLITICAL_PARTY	Republican Party, Democratic Party, GOP
GROUP	The Beatles, Boston Symphony Orchestra
SPORTS_TEAM	the Chicago Bulls, New York Mets
ETHNIC_GROUP	Han race, Hispanic
NATIONALITY	American, Japanese, Spanish
LOCATION	Times Square, Ground Zero
GPE	Asia, Middle East, Palestine
CITY	New York City, Los Angeles
COUNTY	Westchester
PROVINCE	State (US), Province (Canada), Prefecture (Japan)
COUNTRY	the United States of America, Japan, England

Table A.1: OAK System's Named Entities

Tag	Example
REGION	Scandinavia, North America, Asia, East coast
GEOLOGICAL_REGION	Altamira
LANDFORM	Rocky Mountains
WATER_FORM	Hudson River, Fletcher Pond
SEA	Pacific Ocean, Gulf of Mexico
ASTRAL_BODY	Halley's comet, the Moon
STAR	Sirius, Sun, Cassiopeia
PLANET	the Earth, Mars, Venus
ADDRESS	
POSTAL_ADDRESS	715 Broadway, New York, NY 10003
PHONE_NUMBER	222-123-4567
EMAIL	sekine@cs.nyu.edu
URL	http://www.cs.nyu/cs/projects/proteus
FACILITY	Empire State Building, Hunter Mountain Ski Resort
GOE	Pentagon, White House, NYU Hospital
SCHOOL	New York University, Edgewood Elementary School
MUSEUM	MOMA, the Metropolitan Museum of Art
AMUSEMENT_PARK	Walt Disney World, Oakland Zoo
WORSHIP_PLACE	Canterbury Cathedral, Westminster Abbey
STATION_TOP	
AIRPORT	JFK Airport, Narita Airport, Changi Airport
STATION	Grand Central Station, London Victoria Station
PORT	Port of New York, Sydney Harbour
CAR_STOP	Port Authority Bus Terminal, Sydney Bus Depot
LINE	Westchester Bicycle Road
RAILROAD	Metro-North Harlem Line, New Jersey Transit
ROAD	Lexington Avenue, 42nd Street
WATERWAY	Suez Canal, Bering Strait
TUNNEL	Euro Tunnel
BRIDGE	Golden Gate Bridge, Manhattan Bridge
PARK	Central Park, Hyde Park
MONUMENT	Statue of Liberty, Brandenburg Gate
PRODUCT	Windows 2000, Rosetta Stone
VEHICLE	Vespa ET2, Honda Elite 50s
CAR	Ford Escort, Audi 90, Saab 900, Civic, BMW 318i

Table A.2: OAK System's Named Entities continued

Tag	Example
TRAIN	Acela, TGV, Bullet Train
AIRCRAFT	F-14 Tomcat, DC-10, B-747
SPACESHIP	Sputnik, Apollo 11, Space Shuttle Challenger, Mir
SHIP	Titanic, Queen Elizabeth II, U.S.S. Enterprise
DRUG	Pedialyte, Tylenol, Bufferin
WEAPON	Patriot Missile, Pulser P-138
STOCK	NABISCO stock
CURRENCY	Euro, yen, dollar, peso
AWARD	Nobel Peace Prize, Pulitzer Prize
THEORY	Newtons law, GB theory, Blum's Theory
RULE	Kyoto Global Warming Pact, The U.S. Constitution
SERVICE	Pan Am Flight 103, Acela Express 2190
CHARACTER	Pikachu, Mickey Mouse, Snoopy
METHOD_SYSTEM	New Deal Program, Federal Tax
ACTION_MOVEMENT	The U.N. Peace-keeping Operation
PLAN	Manhattan Project, Star Wars Plan
ACADEMIC	Sociology, Physics, Philosophy
CATEGORY	Bantam Weight, 48kg class
SPORTS	Men's 100 meter, Giant Slalom, ski, tennis
OFFENCE	first-degree murder
ART	Venus of Melos
PICTURE	Night Watch, Monariza, Guernica
BROADCAST_PROGRAM	Larry King Live, The Simpsons, ER, Friends
MOVIE	E.T., Batman Forever, Jurassic Park, Star Wars
SHOW	Les Miserables, Madam Butterfly
MUSIC	The Star Spangled Banner, My Life, Your Song
PRINTING	2001 Consumer Survey
BOOK	Master of the Game, 1001 Ways to Reward Employees
NEWSPAPER	The New York Times, Wall Street Journal
MAGAZINE	Newsweek, Time, National Business Employment Weekly
DISEASE	AIDS, cancer, leukemia
EVENT	Hanover Expo, Edinburgh Festival
GAMES	Olympic, World Cup, PGA Championships
CONFERENCE	APEC, Naples Summit
PHENOMENA	El Nino
WAR	World War II, Vietnam War, the Gulf War
NATURAL_DISASTER	Kobe Earthquake
CRIME	Murder of Black Dahlia, the Oklahoma City bombing
TITLE	Mr., Ms., Miss., Mrs,
POSITION_TITLE	President, CEO, King, Prince, Prof., Dr.

Table A.3: OAK System's Named Entities continued

Tag	Example
LANGUAGE	English, Spanish, Chinese, Greek
RELIGION	Christianity, Islam, Buddhism
NATURAL_OBJECT	mitochondria, shiitake mushroom
ANIMAL	elephant, whale, pig, horse
VEGETABLE	spinach, rice, daffodil
MINERAL	Hydrogen, carbon monoxide
COLOR	black, white, red, blue
TIME_TOP TIMEX	
TIME	10 p.m., afternoon
DATE	August 10, 2001, 10 Aug. 2001
ERA	Glacial period, Victorian age
PERIODX	2 semesters, summer vacation period
TIME_PERIOD	10 minutes, 15 hours, 50 hours
DATE_PERIOD	10 days, 50 days
WEEK_PERIOD	10 weeks, 50 weeks
MONTH_PERIOD	10 months, 50 months
YEAR_PERIOD	10 years, 50 years
NUMEX	100 pikel, 10 bits
MONEY	$10, 100 yen, 20 marks
STOCK_INDEX	26 5/8,
POINT	10 points
PERCENT	10%, 10.5 percent
MULTIPLICATION	10 times
FREQUENCY	10 times a day
RANK	1st prize, booby prize
AGE	36, 77 years old
MEASUREMENT	10 bytes, 10 Pa, 10 millibar
PHYSICAL_EXTENT	10 meters, 10 inches, 10 yards, 10 miles
SPACE	10 acres, 10 square feet
VOLUME	10 cubic feet, 10 cubic yards
WEIGHT	10 milligrams, 10 ounces, 10 tons
SPEED	10 miles per hour, Mach 10
INTENSITY	10 lumina, 10 decibel
TEMPERATURE	60 degrees
CALORIE	10 calories
SEISMIC_INTENSITY	6.8 (on Richter scale)

Table A.4: OAK System's Named Entities continued

COUNTX N_PERSON	10 biologists, 10 workers, 10 terrorists
N_ORGANIZATION	10 industry groups, 10 credit unions
N_LOCATION	10 cities, 10 areas, 10 regions, 10 states
N_COUNTRY	10 countries
N_FACILITY	10 buildings, 10 schools, 10 airports
N_PRODUCT	10 systems, 20 paintings, 10 supercomputers
N_EVENT	5 accidents, 5 interviews, 5 bankruptcies
N_ANIMAL	10 animals, 10 horses, 10 pigs
N_VEGETABLE	10 flowers, 10 daffodils
N_MINERAL	10 diamonds

Table A.5: OAK System's Named Entities continued

Appendix B

Question Type Examples

B.1 When Questions

- *WHEN DAY* - Pattern "*When is*"
  - Examples:
    - When is the Tulip Festival in Michigan?
    - When is Dick Clark's birthday?
    - When is hurricane season in the Caribbean?
    - When is the summer solstice?
    - When is Father's Day?
- *WHEN YEAR* - No pattern. Anything not a *WHEN DAY*
  - Examples:
    - When was 'Tale of Genji' written?
    - When was Florence Nightingale born?
    - When was the Khmer Rouge removed from power?
    - When was Nimitz born?
    - When was the USS Constitution commissioned?
    - When was the Nobel prize first given?

        * When was Sacajawea born?
        * When was the IFC established?
        * When was Abu Nidal born?
        * When was Carlos the Jackal captured?

B.2 Who Questions

- *WHO DEFINITION* - Pattern "*Who [is or was] [NAME]?*"
    - Examples:
        * Who is Barbara Jordan?
        * Who is William Wordsworth?
        * Who is Desmond Tutu?
        * Who is Peter Weir?
        * Who is Zebulon Pike?
        * Who is Langston Hughes?
- *WHO LIST* - Pattern "*Who are*"
    - Examples:
        * Who are professional female boxers?
        * Who are 6 actors who have played Tevye in "Fiddler on the Roof"?
        * Who are 3 authors who have written books about near death experiences?
- *WHO FACTOID* - All the rest.
    - Examples:
        * Who is the richest person in the world?
        * Who was the first coach of the Cleveland Browns?

    * Who is the prophet of the religion of Islam?
    * Who was the architect of Central Park?
    * Who invented paper?
    * Who was the first king of England?
    * Who is the richest woman in the world?
    * Who invented basketball?
    * Who was considered to be the father of psychology?
    * Who built the first pyramid?
    * Who invented the game of bowling?

B.3 Where Questions

- *WHERE SCHOOL* - Pattern "*college, university, degree*"
  - Examples:
    * Where did Hillary Clinton graduate college?
    * Where did David Ogden Stiers get his undergraduate degree?
    * Where did Bill Gates go to college?
- *WHERE LOCATION* - All the rest
  - Examples:
    * Where is Glasgow?
    * Where is Amsterdam?
    * Where did guinea pigs originate?
    * Where did Woodstock take place?
    * Where is Venezuela?
    * Where is Las Vegas?

    * Where is Tufts University?
    * Where did Wicca first develop?
    * Where are diamonds mined?
    * Where is Windsor Castle?

B.4 How Questions

- *HOW LARGE* - Pattern "*How [big or large]*"
  - Examples:
    * How big is the Electoral College?
    * How big is Mars?
    * How big is the sun?
    * How big does a pig get?
    * How big do iguanas get?
- *HOW LATE* - Pattern "*How late*"
  - Examples:
    * How late is Disneyland open?
    * How late in pregnancy will airlines let you fly?
- *HOW ACCURATE* - Pattern "*How accurate*"
  - Examples:
    * How accurate are HIV tests?
- *HOW DISTANCE* - Pattern "*How [far or tall or wide or short or high or close or deep]*"
  - Examples:

    - How far is Pluto from the sun?
    - How wide is the Milky Way Galaxy?
    - How far away is the moon?
    - How tall is the Sphinx?
    - How tall is Tom Cruise?
    - How far would you run if you participate in a marathon?
    - How far away from the sun is Saturn?
    - How tall is Mount McKinley?
    - How tall is the Eiffel Tower in France?

- *HOW OFTEN* - Pattern "*How [often or frequent]*"
  - Examples:
    - How often is someone murdered in the United States?
    - How often does the men's soccer World Cup take place?
    - How often does Hale Bopp comet approach the Earth?
- *HOW LONG* - Pattern "*How long*"
  - Examples:
    - How long in miles is the Columbia River?
    - How long did the Charles Manson murder trial last?
    - How long does it take to travel from Tokyo to Niigata?
    - How long is the Great Barrier Reef?
- *HOW MUCH* - Pattern "*How much*"
  - Examples:
    - How much is the international space station expected to cost?
    - How much vitamin C should you take in a day?

    * How much sleep should a child get at night?
    * How much did the first Barbie cost?
    * How much calcium is in broccoli?
    * How much is the Sacajawea coin worth?

- *HOW TEMP* - Pattern "*How [warm or cold or hot]*"
  - Examples:
    * How hot is the core of the Earth?
    * How hot does the inside of an active volcano get?
    * How hot is the sun?
- *HOW FAST* - Pattern "*How fast*"
  - Examples:
    * How fast is the speed of light?
    * How fast is sound?
    * How fast does a cheetah run?
    * How fast does an iguana travel (mph)?
    * How fast is the world spinning?
    * How fast is an eye blink?
- *HOW OLD* - Pattern "*How old*"
  - Examples:
    * How old was the youngest president of the United States?
    * How old do you have to be to get married in South Carolina?
    * How old was Nolan Ryan when he retired?
    * How old was George Washington when he died?
    * How old is the universe?

    * How old must you be to become President of the United States?
    * How old is the Red Pyramid?
    * How old was Babe Ruth when he died?
    * How old was Elvis when he died?

- *HOW DEATH* - Pattern "*How did [NAME] die?*"
  - Examples:
    * How did Harry Chapin die?
    * How did Joseph Smith die?
    * How did John Quincy Adams die?
    * How did Anne Frank die?
    * How did Brandon Lee die?
    * How did John Dillinger die?
    * How did Julius Irving's son die?

- *HOW METHOD* - No pattern. Questions not classified yet are classified as this.
  - Examples:
    * How did the Lindy Hop get its name?
    * How did Hawaii become a state?
    * How did Cincinnati get its name?
    * How did Minnesota get its name?

B.5 What questions

- *WHAT DEF* - Patterns "*What ((is) or (are)) [TERM TO BE DEFINED]*"
  - Examples:

    * What is a nematode?
    * What is a meerkat?
    * What is porphyria?
    * What is anorexia nervosa?
    * What is an atom?
    * What is autism?
    * What is epilepsy?
    * What is a biosphere?
    * What is bipolar disorder?
    * What is cholesterol?
    * What is caffeine?
    * What are invertebrates?

- *WHAT ACRO* - Patterns "*stand for* or *stands for* or *an acronym for* or *is the abbreviation for* or *the ((acronym) or (abbreviation))*"
  - Examples:
    * What does NAFTA stand for?
    * What does hazmat stand for?
    * What does CNN stand for?
    * What does CPR stand for?
    * What is the abbreviation for Original Equipment Manufacturer?
    * What does EKG stand for?
    * What is the abbreviation for limited partnership?
- *WHAT VERB* - Pattern is question ends in a verb
  - Examples:
    * What do ladybugs eat?

    * What did Charles Babbage invent?
    * What did Alfred Noble invent?
    * What do manatees eat?
    * What did Vasco da Gama discover?
    * What do river otters eat?

- *WHAT CITY* - Focus Pattern "*city or town or capital or village*"
  - Examples:
    * What city in China has the largest number of foreign financial companies?
    * What city is the US Declaration of Independence located in?
    * What city has the oldest relationship as a sister-city with Los Angeles?
    * What city is Disneyland in?
    * What city is the River Seine in?
    * What city is the home to the Rock and Roll Hall of Fame?
- *WHAT COUNTRY* - Focus Pattern "*country*"
  - Examples:
    * What country is the worlds leading supplier of cannabis?
    * What country is Aswan High Dam located in?
    * What country did Catherine the Great rule?
    * What country made the Statue of Liberty?
    * What country is the largest in square miles?
- *WHAT PROVINCE* - Focus Pattern "*province*"
  - Examples:

    * What French province is cognac produced in?
    * What province in Canada is Niagara Falls located in?
    * What province is Calgary located in?

- *WHAT FLOWER* - Focus Pattern "*flower*"
  - Examples:
    * What flower did Vincent Van Gogh paint?
    * What is Australia's national flower?
    * What is Hawaii's state flower?
    * What is the Illinois state flower?
- *WHAT BIRD* - Focus Pattern "*bird*"
  - Examples:
    * What is the Ohio state bird?
    * What is Maryland's state bird?
    * What is the smallest bird in Britain?
- *WHAT TREE* - Focus Pattern "*tree*"
  - Examples:
    * What is California's state tree?
- *WHAT DATE* - Focus Pattern "*date or day*"
  - Examples:
    * What date did the United States civil war start?
    * What date did the Lusitania sink?
    * What date was the Declaration of Independence signed on?
- *WHAT YEAR* - Focus Pattern "*year*"

- Examples:
  - What year did "Snow White come out?
  - What year did Nintendo 64 come out?
  - What year was Ebbets Field, home of Brooklyn Dodgers, built?
  - What year was the phonograph invented?

- *WHAT NAME* - Focus Pattern "*name*"
  - Examples:
    - What was W.C. Fields' real name?
    - What was Dr. Seuss' real name?
    - What is Mark Twain's real name?
    - What is Tina Turners real name?
    - What is Marilyn Monroe's real name?

- *WHAT CONTINENT* - Focus Pattern "*continent*"
  - Examples:
    - What continent is Bolivia on?
    - What continent is Egypt on?
    - What continent is Argentina on?
    - What continent is India on?
    - What continent is Scotland in?

- *WHAT POPULATION* - Focus Pattern "*population*"
  - Examples:
    - What is the population of Mexico?
    - What is the population of Kansas?
    - What is the population of Mozambique?

    * What is the population of Ohio?
    * What is the population of the United States?
    * What is the population of Mississippi?

- *WHAT COMPANY* - Focus Pattern "*company*"
  - Examples:
    * What company is the largest Japanese ship builder?
    * What company created the Internet browser Mosaic?
    * What company manufactures Sinemet?
    * What company owns the soft drink brand "Gatorade"?
- *WHAT INSTRUMENT* - Focus Pattern "*instrument*"
  - Examples:
    * What instrument did Glenn Miller play?
    * What instrument does the concertmaster of an orchestra usually play?
    * What instrument measures radioactivity?
- *WHAT COLOR* - Focus Pattern "*color*"
  - Examples:
    * What color is the top stripe on the United States flag?
    * What color belt is first in karate?
    * What color hair did Thomas Jefferson have before gray?
- *WHAT NATIONALITY* - Focus Pattern "*nationality*"
  - Examples:
    * What nationality is Sean Connery?

∗ What nationality is Pope John Paul II?

∗ What nationality is architect Frank Gehry?

∗ What is Al Jolson's nationality?

References

Ageno, A., D. Ferres, E. Gonzalez, S. Kanaan, H. Rodriguez, M. Surdeanu, and J. Turmo. (2004). TALP-QA system at TREC-2004: Structural and hierarchical relaxation over semantic constraints. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Ahn, D., V. Jijkoun, J. Kamps, G. Mishne, K. Muller, M. de Rijke, and S. Schlobach. (2004). The University of Amsterdam at TREC2004. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*.

Baeza-Yates, R. and B. Ribeiro-Neto, (1999). *Modern Information Retreival*, chapter 8, pages 192–199. Pearson Education Limited.

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *National Conference on Artificial Intelligence*, pages 722–727, Seattle, Washington.

Buckley, C. 1985. Implementation of the smart information retrieval system. Technical report.

Cannataro, M. and D. Talia. (2003). The knowledge grid. *CACM*, 46(1):89–93.

Chali, Y. and S. Dubien. (2004). University of Lethbridge's participation in TREC-2004 QA track. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Chali, Y. and M. Kolla. (2004). Summarization techniques at DUC 2004. In *Proceedings of the Document Understanding Conference*, pages 123 - 131, Boston. NIST.

Chen, J., G. He, Y. Wu, and S. Jiang. (2004). Unt at TREC 2004: Question answering combining multiple evidences. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Chinchor, N. A. (1998). Overview of MUC-7/MET-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA.

Chu-Carroll, J., K. Czuba, J. Prager, A. Ittycheriah, and S. Blair-Goldensohn. (2004). IBM's PIQUANT II in TREC2004. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Clarke, C. L. A, G. V. Cormack, and T. R. Lynam. (2001). Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365, New Orleans, Louisiana.

Clifton, T. and W. Teahan. (2004). Bangor at TREC 2004: Question answering track. In *Proceedings of the Thirteenth Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL-96*, pages 184–191, Santa Cruz, CA.

Cui, H., K. Li, R. Sun, and T. Chun M. Kan. (2004). National University of Singapore at the TREC-13 question answering main task. In *Proceedings of the Thirteenth Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Diekema, A. R., O. Yilmazel, and E. D. Liddy. (2004). Evaluation of restricted domain question-answering systems. In *Proceedings of EACL Workshop on Question Answering in Restricted Domains*, Barcelona, Spain.

Echihabi, A., U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. (2003). Multiple-engine question answering in Textmap. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 772–781, Gaithersburg, Maryland.

Erbach, G. 2004. Evaluating human question-answering performance under time constraints. In *Proceedings of Cross-Language Evaluation Forum Workshop 2004 (CLEF 2004)*, Bath, UK.

Fellbaum, C. (1998). Wordnet - an electronic lexical database. Cambridge, MA. MIT Press.

Ferret, O., B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, and A. Vilnat. (2001). Finding an answer based on the recognition of the question focus. In *Proceedings of the Tenth Text REtreival Conference (TREC 2001)*, Gaithersburg, Maryland.

Gaizauskas, R., M. A. Greenwood, M. Hepple, I. Roberts, and H. Saggion. (2004). The University of Sheffield's TREC 2004 Q&A experiments. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Harabagiu, S., D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. (2001). The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics (ACL-2001)*, pages 274–281, Toulouse, France.

Harabagiu, S. M. and S. J. Maiorano. (1999). Finding answers in large collections of texts: Paragraph indexing + abductive inference. In *AAAI Fall Symposium on Question Answering Systems*, pages 63–71, North Falmouth, Massachusetts.

Hermjakob, U. 2001. Parsing and question classification for question answering. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 17–22, Toulouse, France.

Hermjakob, U., A. Echihabi, and D. Marcu. (2003). Natural language based reformulation resource and web exploitation for question answering. In *Proceedings* *of the Eleventh Text REtreival Conference (TREC 2002)*, page 801, Gaithersburg, Maryland.

Hildebrandt, W., B. Katz, and J. Lin. (2004). Answering definition questions using multiple knowledge sources. In *Proceedings of the Human Language Technology conference / North American chapter of the Association for Computational Linguistics 2004*, Boston, Massachusetts.

Jijkoun, V. and M. de Rijke. (2004). Answer selection in a multi-stream open domain question answering system. In *Proceedings 26th European Conference on Information Retrieval (ECIR'04)*, volume 2997 of LNCS, pages 99–11, Springer.

Jijkoun, V., G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. (2003). The University of Amsterdam at the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 586–593, Gaithersburg, Maryland.

Jurafsky, D. and J. H. Martin, (2000). *Speech and Language Processing*. Prentice Hall.

Katz, B., J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora. (2003). Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 426–435, Gaithersburg, Maryland.

Kim, H., K. Kim, G. G. Lee, and J. Seo. (2001). Maya: A fast question-answering system based on a predictive answer indexer. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 9–16, Toulouse, France.

Kwok, C., O. Etzioni, and D. Weld. (2001). Scaling question answering to the web. In *World Wide Web*, pages 150–161, Hong-Kong.

Lehnert, W. G., (1986). *Readings in natural Language Processing*, A Conceptual Theory of Question Answering, pages 651–658. Morgan Kaufmann Publishers Inc.

Leidner, J. L., J. Bos, T. Dalmas, J. R. Curran, S. Clark, C. J. Bannard, B. Webber, and M. Steedman. (2003). QED: The Edinburgh TREC2003 question answering system. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 631–635, Gaithersburg, Maryland.

Li, X. and D. Roth. (2001). Exploring evidence for shallow parsing. In *Proceedings of the Fifth Workshop on Computational Language Learning (CoNLL-2001)*, Toulouse, France.

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

Lin, J., A. Fernandes, B. Katz, G. Marton, and S. Tellex. (2003). Extracting answers from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the Eleventh Text REtreival Conference (TREC 2002)*, page 447, Gaithersburg, Maryland.

Litkowski, K. C. (1999). Question-answering using semantic relation triples. In *Proceedings of the Eighth Text REtrieval Conference (TREC 8)*, page 349, Gaithersburg, Maryland.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31.

Magnini, B., M. Negri, R. Prevete, and H. Tanev. (2002)a. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of the 40th* *annual meeting of the association for computational linguistics (ACL)*, pages 425–432, Philadelphia, PA.

Magnini, B., M. Negri, R. Prevete, and H. Tanev. (2002)b. Mining knowledge for repeated co-occurrences: Diogene at TREC-2002. In *Proceedings of the Eleventh Text REtreival Conference (TREC 2002)*, page 349, Gaithersburg, Maryland.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. (1994). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Massot, M., H. Rodriguez, and D. Ferres. (2003). QA udg-upc system at TREC-12. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 762–771, Gaithersburg, Maryland.

Mihalcea, R. and D. Moldovan. (2001). Document indexing using named entities. *Studies in Information and Control*, 10(1).

Moldovan, D., S. Harabagiu, C. Clark, M. Bowden, J. Lehmann, and J. Williams. (2004). Experiments and analysis of LCC's two QA systems over TREC2004. In *Proceedings of the 13th Text REtrevial Conference (TREC 2004)*, Gaithersburg, Maryland.

Moldovan, D., S. Harabagiu, R. Girju, P. Morarescu, F. Lactusu, A. Novischi, A. Badulescu, and O. Bolohan. (2002). LCC tools for question answering. In *Proceedings of the Eleventh Text REtreival Conference (TREC 2002)*, page 388, Gaithersburg, Maryland.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. (2000). The structure and performance of an open-domain question answering system. In *38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus. (1999). Lasso: A tool for surfing the answer net. In *Proceedings of the 8th Text REtreival Conference (TREC 1999)*, Gaithersburg, Maryland.

Moldovan, D. and A. Novischi. (2002). Lexical chains for question answering. In *Proceedings of COLING 2002*, pages 674–680, Taipei, Taiwan.

Molla, D. and M. Gardiner. (2004). Answerfinder at TREC 2004. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Morris, J. and G. Hirst. (1991). Lexical cohesion computed by thesaural relations as an indicator of structure of text. *Computational Linguistics*, 17(1):21–48.

Nyberg, E., T. Mitaamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupsc, L. V. Lita, V. Pedro, D. Svoboda, and B. Van Durme. (2003). The javelin question-answering system at TREC 2003: A multi-strategy approach with dynamic planning. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, Gaithersburg, Maryland.

Pasca, M. and S. M. Harabagiu. (2001)a. Answer mining from on-line documents. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 38–45, Toulouse, France.

Pasca, M. A. and S. M. Harabagiu. (2001)b. High performance question/answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 366–374, New Orleans, Louisiana.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137.

Pradhan, S. S., G. Illouz, S. J. Blair-Goldensohn, A. H. Schlaikjer, V. Krugler, E. Filatova, P. A. Duboue, H. Yu, R. J. Passonneau, S. Bethard, V. Hatzivassiloglou, W. Ward, D. Jurafsky, K. R. McKeown, and J. H. Martin. (2002). Building a foundation system for producing short answers to factual questions. In *Proceedings of the Eleventh Text REtreival Conference (TREC 2002)*, page 621, Gaithersburg, Maryland.

Prager, J., J. Chu-Carroll, K. Czuba, C. Wlty, A. Ittycheriah, and R. Mahindru. (2003). IBM's PIQUANT in TREC2003. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 283–292, Gaithersburg, Maryland.

Ramshaw, L. and M. Marcus. (1995). Text chunking using transformation-based learning. In D. Yarovsky and K. Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania.

Ravichandran, D. and E. Hovy. (2002). Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual meeting of the association for Computational Linguistics (ACL)*, pages 41–47, Philadelphia, PA.

Roussinov, D., Y. Ding, and J. A. Robles-Flores. (2004). Experiments with web QA system and TREC2004 questions. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Sekine, S. (2002). Proteus project oak system (english sentence analyzer), http://nlp.nyu.edu/oak.

Srihari, R. and W. Li. (1999). Information extraction supported question answering. In *Proceedings of the Eighth Text REtreival Conference (TREC 1999)*, Gaithersburg, Maryland.

Tanev, H., M. Kouylekov, and B. Magnini. (2004). Combining linguistic processing and web mining for question answering: Itc-irst at TREC-2004. In *Proceedings of the Thirteenth Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Teahan, W. J. (2003). Knowing about knowledge: Towards a framework for knowledgeable agents and knowledge grids. Technical report, Artificial Intelligence and Intelligent Agents Tech Report AIIA 03.2, School of Informatics, University of Wales, Bangor.

Turing, A. M., (1950). *Computing Machinery and Intelligence*, Chapter 59, pages 433–460.

Voorhees, E. M. (1999). Overview of the TREC 1999 question answering track. In *Proceedings of the 8th Text REtreival Conference (TREC 1999)*, Gaithersburg, Maryland.

Voorhees, E. M. (2000). Overview of the TREC 2000 question answering track. In *Proceedings of the 9th Text REtreival Conference (TREC 2000)*, Gaithersburg, Maryland.

Voorhees, E. M. (2001). Overview of the TREC 2001 question answering track. In *Proceedings of the 10th Text REtreival Conference (TREC 2001)*, Gaithersburg, Maryland.

Voorhees, E. M. (2002). Overview of the TREC 2002 question answering track. In *Proceedings of the Eleventh Text REtreival Conference (TREC 2002)*, Gaithersburg, Maryland.

Voorhees, E. M. (2003). Overview of the TREC 2003 question answering track. In *Proceedings of the 12th Text REtreival Conference (TREC 2003)*, pages 54–68, Gaithersburg, Maryland.

Voorhees, E. M. (2004). Overview of the TREC 2004 question answering track. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Waltz, D L. (1978). An english language question answering system for a large relational database. *Communications of the ACM*, 21(7).

Witten, I., A. Muffat, and T. Bell. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kaufmann.

Wu, L., X. Huang, L. You, Z. Zhang, X. Li, and Y. Zhou. (2004). FDUQA on TREC2004 QA track. In *Proceedings of the 13th Text REtreival Conference (TREC 2004)*, Gaithersburg, Maryland.

Wu, M., X. Zheng, M. Duan, T. Liu, and T. Strzalkowski. (2003). Question answering by pattern matching, web-proofing, semantic form proofing. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 578–585, Gaithersburg, Maryland.

Xu, J., A. Licuanan, J. May, S. Miller, and R. Weischedel. (2002). TREC2002 QA at bbn: Answer selection and confidence estimation. In *Proceedings of the Eleventh Text REtreival Conference (TREC 2002)*, pages 96–101, Gaithersburg, Maryland.

Xu, J., A. Licuanan, and R. Weischedel. (2003). TREC2003 QA at BBN: Answering definitional questions. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 98–108, Gaithersburg, Maryland.

Yang, H., H. Cui, M. Maslennikov, L. Qui, M. Kan, and T. Chua. (2003). Qualifier in TREC-12 QA main task. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003)*, pages 480–488, Gaithersburg, Maryland.