Exploitation of Redundant Inverse Term Frequency for Answer Extraction

by

Thomas Richard Lynam

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2002

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronicaly available to the public

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

An automatic question answering system must find, within a corpus, short factual answers to questions posed in natural language. The process involves analyzing the question, retrieving information related to the question, and extracting answers from the retrieved information. This thesis presents a novel approach to answer extraction in an automated question answering (QA) system.

The answer extraction approach is an extension of the MultiText QA system. This system employs a question analysis component to examine the question and to produce query terms for the retrieval component which extracts several document fragments from the corpus. The answer extraction component selects a few short answers from these fragments. This thesis describes the design and evaluation of the Redundant Inverse Term Frequency (RITF) answer extraction component.

The RITF algorithm locates and evaluates words from the passages that are likely to be associated with the answer. Answers are selected by finding short fragments of text that contain the most likely words based on: the frequency of the words in the corpus, the number of fragments in which the word occurs, the rank of the passages as determined by the IR, the distance of the word from the centre of the fragment, and category information found through question analysis. RITF makes a substantial contribution in overall results, nearly doubling the Mean Reciprocal Rank (MRR), a standard measure for evaluating QA systems.

Acknowledgements

I would like to thank everyone that has given their support and assistance to the completion of the this thesis.

Special thanks must go to Gordon Cormack for his dedication and support as well as Charlie Clarke and Nick Cercone for their contributions.

I would also like to express my gratitude to Kimberly Becken for her patience and assistance.

Finally, thanks must be given to the University of Waterloo for its support.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

This thesis describes a new approach to answer extraction in an automated question answering (QA) system. The purpose of a QA system is to find, within a corpus, short factual answers to questions posed in natural language. For example , "Alexei Leonov" would be an appropriate answer to the question "Who was the first person to walk in space?"

A typical QA system has three components: Question Analysis, Information Retrieval, and Answer Extraction as shown in Figure: 1.1. The analysis component examines the question and produces query terms for the retrieval component which extracts several documents or fragments from the corpus that are likely to be relevant to the question. The Answer Extraction component selects a few short answers from these fragments.

This thesis describes the design and evaluation of the Redundant Inverse Term

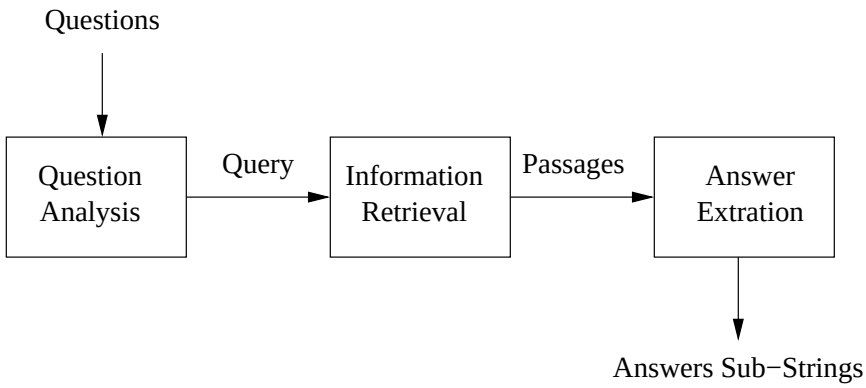

Figure 1.1: Standard QA Approach

Frequency (RITF) answer extraction component, deployed within the MultiText QA system for participation in the Ninth Text REtrieval Conference (TREC-9). Overall, the MultiText system's results ranked among the best three participating systems. The analysis presented in this thesis shows that RITF makes a substantial contribution in overall results, nearly doubling[1] the performance of MultiText's TREC-8 system[8], which lacked RITF.

The crux of RITF is to discover words or short phrases likely to be associated with the answer, and to select an answer that is associated with a large number of these words. This likelihood is based on: the query frequency of the words in the corpus, the number of fragments in which the word occurs, the rank of the passages as determined by the information retrieval component, the distance of the word from the centre of the fragment, and category information gleaned from the question analysis component.

[1] As measured by Mean Reciprocal Rank (MRR), the standard TREC measure, on the TREC-9 QA task.

Chapter 2 describes Question Answering and question answering systems in general. The chapter focuses specifically on the TREC QA tasks and MultiText system which forms the foundation of this work. Chapter 3 details RITF, while Chapter 4 evaluates the impact of RITF on QA performance. Chapter 5 presents conclusions and Chapter 6 suggests avenues for further investigation.

Chapter 2

Foundation

2.1 Introduction to Question Answering

The object of question answering (QA) systems is to discover short factual answers in a corpus for a posed question. This objective differs from that of traditional information retrieval systems, which is to discover documents or document fragments relevant to a particular topic, but not necessarily answering a specific question. Nevertheless, the techniques of IR find common use in QA systems.

A typical QA system will first analyze the question to determine the key question words and concepts from which a query for the IR component is formulated. Second, the query is given to the IR component to retrieve documents or document fragments that contain possible answers to the question. Finally, the answer extraction component searches for short answers in the retrieved information.

Question answering systems use many different types of question analysis, information retrieval and answer extraction components. The architecture of a general question answering system is shown in Figure: 1.1.

2.2 Question Analysis

The analysis of the question serves two main purposes. First, question analysis is used to determine the key terms in the question. These key terms are used to produce queries for the information retrieval aspect of the system. Second, question analysis is responsible for the classification of the question; for example, the question may be classified into the traditional "who", "what", "where" and "why", as the question categories. There are several different methods used to determine key terms and question classification. QA systems may perform other tasks such as creating patterns that may identify answers or using a database to find possible answers independent of the corpus during question analysis.

A wide range of approaches may be used to find the key question terms to be used in information retrieval query. The most naive would be to use every word in the question as a query terms. More sophisticated approaches might be to exclude stop words or to stem suffixes or to add synonyms. These are all standard IR preprocessing techniques. More advanced systems may use Natural Language Processing (NLP)[25, 21, 15, 2] techniques to identify parts of speech, sentence components, question category, or other semantic information.

Question classification is an important part of question analysis. Searching for a date or monetary value is very different from searching for a person's name. Question classification may be accomplished using a wide range of techniques. Possible methods are keyword analysis, pattern matching, and NLP parsing. Pattern matching fits questions to predetermined classifications. Using this, the question "What is/a[n] X?" would be classified as a definition question.

It is possible for question analysis to do early answering for some questions. For the example question "What is leukemia?" it is possible to use an external database such as WordNet[18] to find the answer to this question. WordNet's short definition of leukemia is "cancer of the blood". Knowing this, the system has the option of adding these terms to the query.

2.3 Information Retrieval

The role of the Information Retrieval (IR) component in QA is to retrieve relevant documents or fragments for the given query derived from question analysis.

IR techniques are traditionally evaluated using precision and recall[33]. Precision is defined as the fraction of relevant documents a system returns. Recall is defined as the fraction of all relevant documents returned by the system. Recall has little relevance to QA - what is important is that at least one returned item contains the answer. It may be assumed that if an IR system has high precision for the items it returns, one of these items contains the answer.

For the purpose of question answering, two different approaches to retrieval should be considered. QA systems use either document or passage retrieval. Document retrieval returns full documents whereas passage retrieval returns only parts of documents. It is an advantage in QA to have a system that returns only passages, provided that the passages have good precision, because passage retrieval narrows the amount of information that needs to be processed during answer extraction.

2.4 Answer Extraction

Answer Extraction (AE) locates answers in the retrieved documents or passages. First, extraction methods find candidate answers, which are then evaluated. Some of the methods to discover candidate answers are pattern matching, Information Extraction (IE)[3] and grammatically matching questions and retrieved text segments using NLP techniques.

Pattern matching is a simple form of Information Extraction that may, for example, use upper case words where a proper name is required, or numbers where a quantity is required. IE is particularly suitable for finding candidate answers when the question has been correctly categorized. Answer Extraction may use the IE technique of named entity tagging to locate candidate answers. This technique can discover names of people, places, organizations, dates, times, amounts, etc., which suits answer extraction provided that the question is correctly categorized by the analysis component. Through the Message Understanding Conference (MUC)[3],

techniques of information extraction have been researched and evaluated.

Once candidate answers have been discovered, the best ones must be identified. A statistical approach could use the frequency of candidate answers in the corpus. A voting scheme could use frequency of candidate answers in the retrieved passages. An external database could be used to verify the answer or the category of the answer information for some question classification. A question requiring a country as an answer could use a database to verify that any candidate answer was a country.

2.5 Text REtrieval Conference

The Text REtrieval Conference (TREC) started in 1992[13], although the question answering track was not added until 1999[31, 30]. The conference is co-sponsored by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA). The program committee is made up of government, industry, and academia members.

The goal of the conference is to encourage research in information retrieval and to create an open forum for communication to increase the sharing of research ideas. However, the important element TREC provides is a standard and unbiased manner to evaluate what techniques perform best. Knowing which techniques work well for a given task allows for the sharing of ideas and techniques to improve.

The TREC conference is different from most conferences in that its participants are required to create a system for one or more of its research tracks. A list of

the tracks is in Table: 2.1 along with the number of participants[9]. While the organization of all the tracks is similar, this thesis only involves the QA track.

	TREC-8	TREC-9
Ad Hoc	41	-
Cross-Language	12	16
Filtering	14	15
Interactive	7	6
Question Answering	20	28
Query	5	6
Spoken Document	10	3
Small Web	17	23
Large Web	8	-

Table 2.1: TREC Participants per Track

For each track, documents and questions are provided. Each participant runs their system on the data to create a set of results that is given to NIST. A run must be a fully automated process except for the interactive track, which is based on manual interaction. NIST pools all the different systems' results together and judges them for correctness[28, 29]. This means that if two systems produce exactly the same result for a question they will have the same evaluation.

NIST provides the TREC data sets and evaluation software for research in information retrieval. These evaluation suites allow the ability to evaluate either a new system or the differences that parts of a system provide.

2.6 TREC-8 QA

The Question Answering track was introduced at TREC-8[28]. The new track required the QA systems to find short answers to each of a set of 200 questions. The process of finding short answers was to be fully automatic. Systems would be evaluated on their ability to return short extracts containing a correct answer justified by a document in the corpus.

TREC-8 required the system to find solutions to 200 questions. Each question was guaranteed to have an answer that appeared in the corpus. Participants were given 30 training questions in advance of the evaluated trial. A sample of TREC-8 questions can be found in Table: 2.2. TREC-8 questions were developed by NIST manually, creating each question or deriving them from FAQFinder[29] log word queries into questions.

Q #	Question
9	How far is Yaroslavl from Moscow?
30	What are the Valdez Principles?
122	Who is section manager for guidance and control systems at JPL?
145	What did John Hinckley do to impress Jodie Foster?
164	Who is the leading competitor of Trans Union Company?

Table 2.2: Some TREC-8 Questions

The QA corpus was fixed. All documents in the corpus were newspaper articles from four newspapers. Participants were aware of the source of the data and could

therefore infer that it had a fairly homogeneous format and the information was of fairly high quality. The size of the corpus was 979,000 documents totaling 3GB.

Systems were judged over two different runs, one restricting answers to a maximum of 250-bytes; the other imposing a more restrictive maximum of 50-bytes. Answer extracts were required to be strings from the corpus. For TREC-8 many groups including MultiText submitted only the (easier) 250-byte run.

At the end of TREC-8 it was announced that the 50-byte run would be compulsory for TREC-9; providing impetus for the work reported here.

2.7 Mean Reciprocal Rank

TREC's standard measure for QA system performance is Mean Reciprocal Rank (MRR). MRR is defined as follows:

> Define $Q = \{q_1, q_2, \ldots, q_n\}$ to be a set of questions.
>
> Define $a = \{docid, answersubtring\}$ to be a result returned containing an answer substring and the document in which it was found.
>
> Define $A = a_{1,1}, a_{1,2}, \ldots a_{1,5}, a_{2,1}, \ldots, a_{n,5}$ to be a sequence of 5 ranked results for each question.

A judgment for each question result is *justified*, *not justified*, or *incorrect*. A *justified* result answer extract contains a correct answer and has a document

identification that supports the answer. A result *not justified* contains a correct answer but its document identification does not support the answer. An *incorrect* result answer extract does not contain the answer. A correct answer is defined as the part of the answer substring that satifies the question as determined by the NIS—T judges.

$$Judge(q_i, A_{ij}) \rightarrow \{justified, not\, justified, incorrect\}$$

$$\begin{aligned} RankStrict(q_i) = \quad & if\ \exists_j Judge(q_i, A_{i,j}) = justified\, then \\ & \min_j(Judge(q_i, A_{i,j}) = justified)) \\ & else\ \infty \end{aligned}$$

$$MRRstrict(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{RankStrict(q_i)}$$

$$\begin{aligned} RankLenient(q_i) = \quad & if\ \exists_j Judge(q_i, A_{i,j}) \neq incorrect\, then \\ & \min_j(Judge(q_i, A_{i,j}) \neq incorrect) \\ & else\ \infty \end{aligned}$$

$$MRRlenient(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{RankLenient(q_i)}$$

The *justified* and *not justified* judgments are used to compute strict and lenient Mean Reciprocal Rank accordingly. Mean Reciprocal Rank is calculated by finding the highest ranking correct answer extract. The sum of the inverse rank (or 0 when no correct answer extract has been found) is divided by the number of questions.

MRR ranges from 0 to 1; higher values denote better performance.

Evaluation Scripts

The TREC QA runs are evaluated manually by NIST staff using the pooling method described in Section: 2.5. The set of all answers judged correct or incorrect is archived and available as a test collection for future experiments. For IR experiments this archive can be used directly for evaluation. It has been observed that the pooling method identifies enough of the relevant documents; therefore, unjudged documents can be assumed "not relevant". This observation allows future experiments to be evaluated automatically. Because QA requires an answer string rather than a document, there are a very large number of correct answer substrings and it is unlikely that the archive would contain exactly the same string as that yielded by a future run.

To facilitate automatic judging, NIST examined the archive and manually prepared a set of Perl patterns to recognize correct answer strings. These patterns were found to approximate the MRR lenient measure when applied automatically to the TREC run[32].

2.8 TREC-8 QA Results

The TREC-8 questions were non-complex, resulting in many systems performing well[29]. The top 10 participants for the 50-byte run are shown in Figure: 2.1. The unofficial 50-byte MultiText run judged by the evaluation script has been added to show the need for an answer extraction component. The top 10 participants for

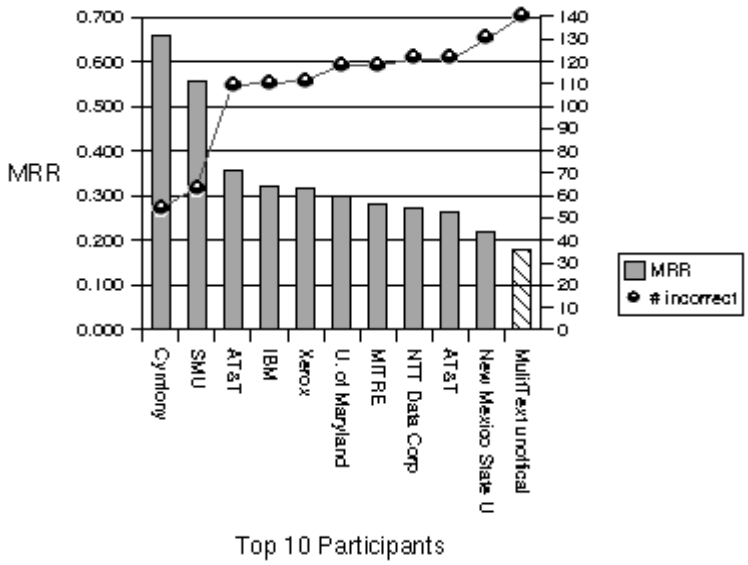

Figure 2.1: TREC-8 50-byte Results

250-byte run are shown in Figure: 2.2. MultiText performed well in the 250-byte

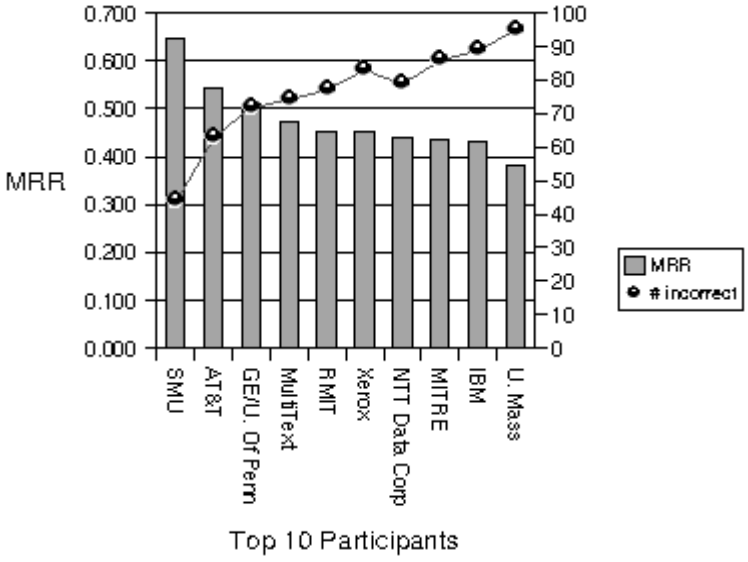

Figure 2.2: TREC-8 250-byte Results

run, proving that it is retrieving passages that contain the answers.

Almost all of the questions in TREC-8 could be answered with a standard Named Entity (NE). The Cymfony QA system[27] was the best performing QA system for the TREC-8 50-byte run. Cymfony's heavy use of NE recognition allowed it to perform well. Southern Methodist's LASSO system[19] also made extensive use of NE recognition as well as other NLP techniques earning it second best 50-byte and best 250-byte results. The Xerox QA[16] system performed well by combining NLP components(parser, sentence boundary identifier, and proper noun tagger). AT&T[26] and University of Pennsylvania[20] QA systems use of passage retrieval allow them to be among the top 250-byte systems. IBM QA[22] system use of predictive annotation allowed it to also perform well.

2.9 MultiText

The MultiText group from the University of Waterloo have participated in many of TREC's tracks, starting in 1995. MultiText created an arbitrary passage retrieval system that was first evaluated in TREC-5[7]. The retrieval system ranks a passage based on its length and the query terms in it. To rank the passages, the cover density algorithm is used. The cover density algorithm essentially considers the shortest covers containing the best query terms as the most relevant. A cover is defined as a document fragment that satisfies a subset of the query terms; that document fragment does not contain a shorter cover that satisfies the same subset of query

terms.

2.10 MultiText TREC-8 QA Architecture

Question Analysis

To produce the query terms, the system examines the question and eliminates stop words. Some simple stemming is done on the question terms. With stop word elimination, sometimes key words are eliminated. An example of this is found in the TREC-8 question, "Who was the first American in space?" whose query is "American", "space".

There was no categorization of the questions, meaning there was only one answer extraction method to handle all the different types of questions.

Information Retrieval

The information retrieval component treated the query terms as a bag of words, a standard IR method. Treating the query term as a bag of words means that all terms are considered to have equal value to the question.

Passages were retrieved using the MultiText search engine. The MultiText engine returns a portion of text that satisfies the query; this portion of text is defined as a cover. A cover cannot contain a smaller cover that satisfies the query. A passage is a cover with text added at both ends. Therefore the hot spot of a

passage is its cover because it is the part of the passage that is most related to the question.

Passages are ranked using cover density that incorporates the number of query terms in the cover and the size of the cover. The more query terms in the cover and the shorter it is, the higher the rank of the cover.

For TREC-8 five passages were found for each question.

Answer Extraction

The TREC-8 QA system used simplistic answer extraction. Covers shorter than 250-bytes were extended on both sides by adjacent words from the original source; covers longer than 250-bytes were truncated. The 50-byte run was not submitted; however, the performance that would have been achieved by this technique is shown in figure 2.1. The answer substrings were kept in the same order that the IR system ranked them.

Though the MultiText question answer system evaluated in TREC-8 is quite simple, it was one of the better systems in the 250-byte run. This good result was based entirely on the strength of the MultiText passage retrieval method.

2.11 Need for Answer Extraction

The requirements for TREC-8 allowed the MultiText QA system to perform well on the 250-bye run but for TREC-9, the 50-byte run was required and stressed

as the more important run[30]. NIST also wanted the system to be able to have exact answers in the future. With the required changes, answer extraction became a necessary component.

2.12 Other QA Approaches for TREC-9

Southern Methodist University's Falcon System

The FALCON[25] question answering system developed at Southern Methodist University outperformed all other systems when evaluated at TREC-9. Questions are first grammatically parsed with the implementation of Collins parser[6]. From the parse question categories, focus and query terms are generated. The Falcon system then uses a modified version of the SMART IR system[2] that is capable of completing Boolean retrieval. Only paragraphs containing query terms are used from the retrieved documents. Named Entity recognizers are then used to find candidate answers. Justification of candidate answers is completed by grammatically comparing answers with the question in what is called an abductive proof[12]. FALCON makes extensive use of the publicly available resource, WordNet[18], for question classification and answer justification.

IBM's GuruQA

The GuruQA[23, 24] has many similarities to the MultiText QA system in its method of answer extraction; GuruQA does not rely heavily on NLP in its answer extraction component. The system's use of predictive annotation allows it to perform well. The question is first analyzed to determine answer type. Next, a query is created by replacing the interrogative word with system defined QA-Tokens. An example of this is changing "when" to DATE$, TIME$, YEARS$. The system utilizes approximately 400 patterns to do this conversion. For a question that does not match a pattern, WordNet is used to find hypernyms that relate to the QA-Tokens.

The system has its own passage retrieval engine that can exploit these QA-Tokens that have been predictive annotated. GuruQA system utilizes a simple version of IDF weighting though it applies it to lexical classes and terms to rank the candidate passages.

Name Entity recognition is the core of the system used for both predictive annotation and for finding candidate answers in the passages. The candidate answers are evaluated with rule based function that includes engine ranking, distance from the beginning of a sentence, or the presence of a QA-Token. The combination of these techniques produces a solid question answer system.

University of Southern California's Webclopedia

The Webclopedia[15] is another system that relies heavily on natural language techniques using an external parser, CONTEX[14], as the core of its system. To train the parser, 250 manually parsed questions were created. The parser is first used to analyze the questions and produce queries and question type. The system has a standard information retrieval engine that returns documents. These documents are broken into small segments which are then ranked. These segments are then parsed so that a comparison evaluation between the segments and question can be obtained.

The system relies heavily on manually built QA patterns and question category targets. Over 500 QA patterns were created. For the amount of work needed, these patterns had very little success. It would be possible to improve the system greatly if a method for creating QA patterns was derived automatically. The system uses a similar inverse log frequency as part of its candidate answer evaluation process but does not take into account redundant answers.

Chapter 3

Redundant Inverse Term Frequency

RITF is one of two new components developed in the course of MultiText's participation in the TREC-9 QA task. Also new was a question analysis component based on a probabilistic context free parser, developed in-house. The parser supplies query terms to the IR component and question category information to RITF. The IR component was essentially unchanged from TREC 8. The TREC-9 architecture is detailed in Figure: 3.1

The primary input to RITF is a set of passages returned by the IR component. From these passages, RITF constructs a ranked list of short substrings likely to contain the answer. To accomplish this task, RITF uses the following additional information:

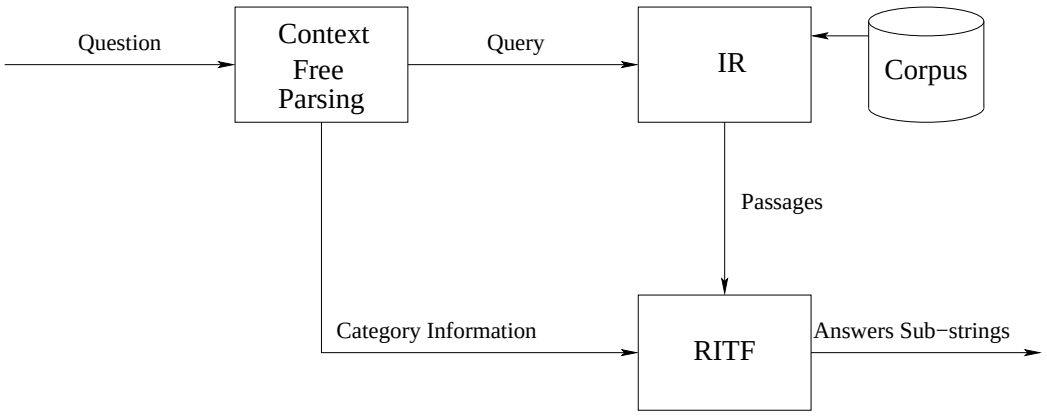

Figure 3.1: QA System Architecture

- the frequency of terms within the corpus
- the category of the question as determined by question analysis
- category-specific rules, patterns, and auxiliary corpora

For example, consider TREC-9 question 330 "When was the slinky invented?" The input and output from the components is the following:

Parser Input

Input for the parser is the question as presented in Table: 3.1.

Number	Question
330	When was the slinky invented?

Table 3.1: Q330 Parser Input

Parser Output

The Output from the parser for TREC-9 question 330 is in Table: 3.2. The parser

Parse Infomation	Parser Output
Query terms	slinky "$invented"
Category	date
Question Type	what
Instance of	NA
Subject	slinky
Predicate	invented

Table 3.2: Q330 Parser Output

creates question type, instance of, subject, and predicate information that is not used by the other components.

IR Output

The IR component returns passages containing covers. In Table: 3.3 the output for TREC-9 question 330 is shown. The italic bold text represents the cover identified by the IR component and the underlined bold text is the answer, yet to be identified.

RITF Output

RITF finds the correct answer substring as its first response. The answer substrings in Table: 3.4 are the output for the RITF component for TREC-9 question 330.

Rank	Cover Length	Passages
1	2	...Betty James took control of the business in 1960, when her husband, who ***invented Slinky***, lost interest and left her and six children to join what she called a religious cult in Bolivia ...
2	2	...It's a fitfully interesting double album of droning guitar-based noodlings that echo '60s sounds but don't achieve their ***inventiveness.*** — ***Slinky*** Jody Watley got her start as a dancer on "Soul Train."...
3	29	...***Slinky's Developer. By CASSANDRA BURRELL. Associated Press Writer. Children around the world have played with springy Slinkies for 43 years, but the 70-year-old woman who helped invent...*** came up with the idea of the toy in ***1943*** while aboard a ship...
...	...	...
10	318	***...invented Audrey Hepburn's best gamine" ... sequined checkerboard decor for the evening wear in trapeze topped minis or slinky...***

Table 3.3: Q330 IR Output

Rank	Answer Substring
1	the idea of the toy in **1943** while aboard a ship.
2	rol of the business in 1960, when her husband, wh
3	, moved Oct. 6 as b0260 for Monday PMs, Oct. 10.
4	kies for 43 years, but the 70-year-old woman who
5	10-block area of 19th-century department stores a

Table 3.4: Q330 RITF Output

Question Analysis

The RITF requires the question analysis component directly as it creates question categories and indirectly because the question analysis component creates queries to retrieve the passages RITF needs. The MultiText QA system uses a context free parser based on a probabilistic version of Earley's Algorithm[10] to perform question analysis. It determines all possible parses of the grammar and selects the most probable. The grammar contains only 80 manually constructed production rules[4].

Part of speech analysis is necessary because the parser uses a context free grammar. WordNet is used to determine whether each word of the question is likely a noun, verb, adjective, or adverb. A simple strategy suggested in WordNet documentation reference[1] is used: The assumption that the probability of a particular part of speech was proportional to the number of senses listed for that part of speech is made. For example, the word "head" has 30 noun senses, 6 verb senses, 2 adjective senses, and no adverb senses, producing probabilities of 0.79, 0.16, 0.95 and ε to these senses.

The information yielded by the parser that is relevant to RITF is the category designation for each question. RITF and the parser were developed independently so their category designations are not identical. Table: 3.5 maps the parser's categories to the eight used by RITF; specifically: PROPER, PLACE, DATE/TIME, MEASUREMENT, DISTANCE, NUMBER, MONEY, OTHER.

Parser	Answer Extraction
Proper	PROPER
Person	PROPER
Place	PLACE
Country	PLACE
City	PLACE
Date	DATE
Time	DATE
Duration	MEASUREMENT
Age	MEASUREMENT
Value	NUMBER
Length	DISTANCE
Money	MONEY
None	OTHER

Table 3.5: Parser to Answer Extraction Category Conversion

IR Component

The IR component regards the corpus as sequence of terms(words). The sequences of terms comprising documents in the corpus are considered to be concatenated in some arbitrary order.

Given a set of query terms, the IR component finds substrings of the corpus that contain several of the query terms but do not traverse a document boundary. These substrings, denoted cover, are ranked according to an estimate of the likelihood that they are relevant to the subject of the query terms. The likelihood of relevance is estimated based on two factors.

1. The length of the cover. A shorter cover is considered more likely relevant than a longer cover containing the same query terms.

2. The terms in the cover. A cover containing terms with lower frequency in the corpus is considered more likely relevant.

The IR component finds the k most-likely-relevant covers, and returns them in order of likelihood of relevance. RITF makes use of context - the terms immediately adjacent to the cover in the corpus. Therefore the results of the IR system were extended to 1000 words by fetching an equal number of terms to the left and the right of each cover.

3.1 RITF Overview

The RITF component's goal is to extract five different answer substrings from the provided passages for each question. To accomplish this goal RITF uses five functions as shown in Figure: 3.2. Candidate information includes a term's position, passage rank, score, weight, the number times it appear in the corpus and the number of passages it appears in.

Find Candidates (Section: 3.2)

Candidate answer are extracted from the retrieved passages based on hand coded patterns for each question category. Candidate information is pass to the RITF Evaluation element.

RITF Evaluation (Section: 3.3)

Each candidate is evaluated based on the likelihood of the term being a correct answer. This likelihood is computed using RITF (Formula: 3.1).

Heuristic Evaluation (Section: 3.4)

Each candidate is evaluated based on likelihood of the term's passage position and rank containing the correct answer. Candidates are also evaluated based on the likelihood that the term is answer for the given question category.

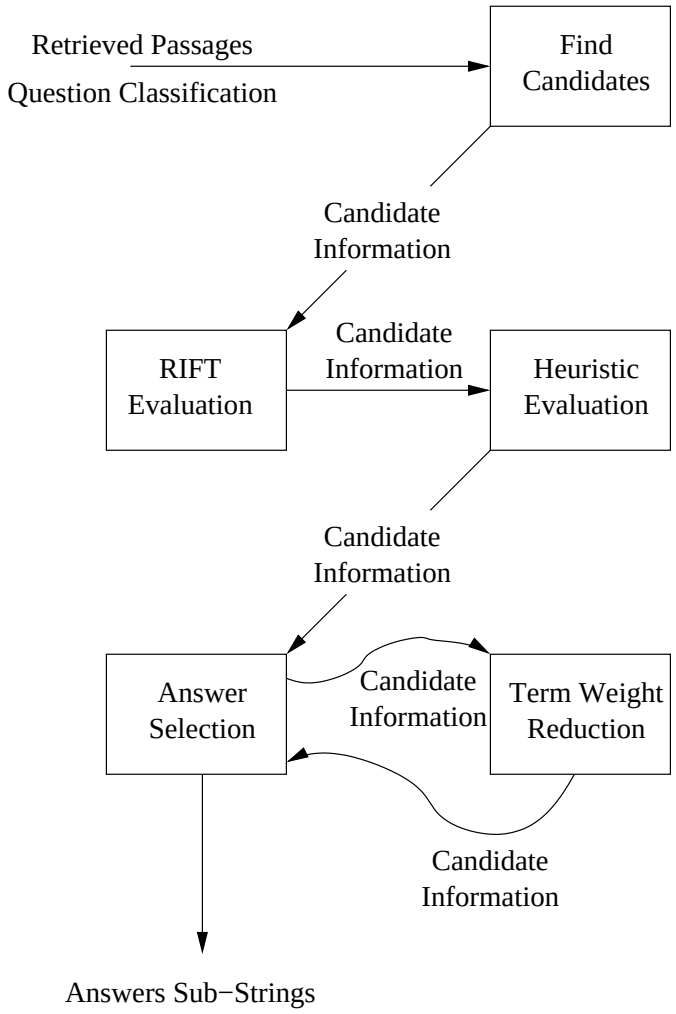

Figure 3.2: RITF Component Architecture

Answer Selection (Section: 3.5)

The answer selected is the substring of chosen length containing the most likely(highest weight) candidate terms.

Term Weight Reduction (Section: 3.6)

A candidate matching a term in the answer substring weights are reduced. The next best substring is selected using the answer selection function.

3.2 Pattern Matching

The purpose of pattern matching is to extract candidate answer terms. Therefore a candidate answer term is defined as any term in the passages that satisfies a pattern for the given question category. All question category patterns only match single terms. Thus no candidate answer will contain more than one word. Each word in the passage either matches a pattern or not.

Candidate terms are located using standard pattern matching techniques:

$$match(qp, P) =< p, i, j > \qquad (p = passage, i = startpos, j = endpos)$$

where qp is the question pattern and P is the set of passages, match then returns a passage and the start and end bytes of the match. Candidate terms are therefore sequences of bytes from a passage. The function is used to find all matches to the pattern.

RITF uses simple regular expression patterns that have been hand-coded[17]. The patterns for each question category are in Table: 3.6.

PROPER	[A-Z][A-Za-z]+	At least two characters long, starts with a capital letter
PLACE	[A-Za-z]{2,}	At least two characters long
DATE	[12][0-9]{3,0} (January|...|December) (Jan|Feb|...|Dec) (Monday|Tuesday|...|Sunday) (Mon|Tue|...|Sun)	Month, weekday, four digit, number maybe followed by an s (ie 1980s)
MONT	$[1-9][0-9]* (Dollars|Euro|Peso|...|Mark)	Currency signs followed by numbers, currency names
NUM	[1-9](,[0-9]{3,3})* [1-9][0-9]*(.[0-9]*)? written_number ={ one|two|... |nineteen) (twenty|thirty|...| ninety) (hundred|thousand|...|trillion)	Written numbers, digits more than 999 must have comas
DIST	[1-9][0-9]* written_number (meters|miles|...|yards)	Written numbers, digits, distance units
MEAS	[1-9][0-9]* written_number (meters|miles|...|yards) (Celsius|Fahrenheit|volts| ...)	Written numbers, digits, measurement units
OTHER	[A-Za-z]{2,} [1-9](,[0-9]{3,3})* [1-9][0-9]*(.[0-9]*)?	Numbers or words but not a combination of both

Table 3.6: Question Category Patterns

Patterns do not contribute to the terms' weight. Each word in the passages is evaluated against the patterns for the question category type. If a word satisfies

a pattern it is added to a list of candidate terms. Stop words and terms in the question are not added to the list of candidate terms. Information about a candidate term's location, length, rank, corpus frequency, weight, and the number of passages containing it must be kept. The list will contain candidate terms more than once if they are found in different places.

3.3 Evaluating Candidate Answers

Candidate answer evaluation is the core of the Redundant Inverse Term Frequency component. It is used to estimate the likelihood of correctness for a candidate term. The power of the answer extraction component is almost entirely derived from this step.

Several different formulas were evaluated for determining term weights. The different formulas range from voting scheme to term location base analysis. Tests of the system using different formulas were done as reported in Chapter 4 (Table: 4.6). The RITF formula provided the best results. The rationale for the Redundant Inverse Term Frequency formula is as follows. Less frequent terms in the corpus that occur in more passages retrieved are more like to be associated with a correct answer[5].

The RITF formula requires the term frequency, the number of retrieved passages it occurred in, and the size of the corpus to function. These are derived in the following manner:

$$C = z_1, z_2, \ldots, z_n$$

$$p = t_u, t_{u+1}, \ldots, t_v$$

$$P = p_1, p_2, \ldots, p_m$$

Consider the corpus a sequence of documents; those documents contain a sequence of terms. Therefore, the corpus C can be considered a sequence of terms. The passage p is a substring of the corpus and therefore also a sequence of terms. The set of ranked passages is defined as P. To find the frequency of the term in the corpus a count of the number of times the term appears is taken.

$$f_t = |\{i|t = z_i\}|$$

Similarly the number of passages that a term occurs in is found through the following formula:

$$c_t = |\{P|\exists_i.t = p_i\}|$$

The main assumption driving RITF is that rare terms occurring in the passages are more likely to be important than common terms. The probability $p(t)$ that any given term t is in the corpus can be estimated by t's frequency:

$$p(t) = \frac{f_t}{|C|}$$

Terms with lower $p(t)$ are more important. The importance (or rareness) is defined to be the logarithm of the reciprocal of the value:

$$s(t) = log(|C|/f_t)$$

where t is the term and C is the corpus.

The term's rareness score is based on the chance that any given term is t. The formula must be modified to reflect the chance of finding a term in more than one passage. Thus, the number of passages the term appears in is multiplied by the term's rareness score. Therefore, each term's RITF value is calculated by the following Formula: (3.1)

$$\lambda_t = c_t log(|C|/f_t) \tag{3.1}$$

where t is the term and C is the corpus and c_t is number passages term appears in, f_t is the frequency of the term in the corpus and λ_t is the overall estimate of the importance of this term to question.

Table: 3.7 shows the most likely candidate terms found in the passages retrieved for TREC-9 question 330.

The formula performs well on the boundary scenarios when a correct answer is very rare or very common in the corpus. A very rare term may not appear very many times in the passages retrieved but its natural rareness is very strong, producing a high score. Also, a very common term will have a poor corpus rareness but the term weight will improve as the number of passages it appears in increases.

3.4 Location and Question Category Heuristics

Term weights are modified depending on a term's location and chance of being an answer to the question category. Location heuristics consider position in a passage and rank of the passage. Question category heuristics modify the term weight based on the likelihood of the term being an answer of the chosen category.

Location Heuristics

The cover is a sequence of terms that satisfies the query derived from the question. Therefore terms closer to the cover are more likely to be related to the question. Thus candidate answer terms closer to the cover are more likely to be answers than terms farther away. To exploit this information the term weight is modified to account for its distance from the centre of the passage. The farther from the centre, the more a candidate term weight is decreased. Based on testing different position heuristics the following calculation was adopted:

$$PosHeur_t = 1 - \frac{1}{1000 - d_t}$$

where d_t is the term's distance in bytes from the centre of the passage in which the candidate term was found.

It can also be seen that as the ranks of passages decrease, so does the likelihood of a passage being relevant. Therefore candidate term weight is modified according to the rank of the passage in which it was found; the lower the ranking, the more the term weight is decreased. The term weight is modified by the following formula:

$$RankHeur_t = 1 - \frac{1}{1000 - r_t}$$

where r_t is the rank of the passages in which the candidate term was found.

Question Category Heuristics

The question category heuristic change the term weights depending on the likelihood of the term to be an answer for the question category. If a term has been determined likely to be an answer, its weight is multiplied by a number that is greater than one. Similarly if a term has been determined to be of low value, it is multiplied by a number less than one.

$$CatHeur_t = EvalHeuristic(t, qc)$$

where t is the term and qc is the question category. The category heuristics had little impacted on the system performance due to over training. Though there is evidence that category heuristic can improve the system. The PLACE category heuristic made an improvement by using a external database of places. This database contained countries, major world cities, and US states.

A term's weight is assigned as the product of the RITF score and the position, rank and category heuristics scores as follows:

$$w_t = \lambda_t * PosHeur_t * RankHeur_t * CatHeur_t$$

The RITF component successfully identifies the correct answer as seen in Table: 3.4 based on the the terms' weight as seen in Table: 3.7.

Q330: When was the slinky invented?

Term	f_t	c_t	$\sim w_t$
1943	2107	2	500000
1960	9216	1	26400
2107	62464	1	51100
monday	171008	2	41000
old	240640	3	8340
years	553984	7	7020
century	46080	1	1420
days	179200	1	305
late	140288	3	58

Table 3.7: Q330 Term Weights

3.5 Answer substring Selection

For TREC-9, the system was required to produce 50- and 250-byte substrings. The answer substring score is based only on the candidate terms appearing in it. The order in which the terms appear does not matter. Each substring is assigned a score equal to the sum of the cube of the terms' weight within it as follows:

$$score_a = \sum_{i=u}^{v} w_{t_i}^3$$

where a is the answer substring and u and v are the first and last candidate term in that substring.

Using a sliding text window of the required size, the best answer is the substring with the highest $score_a$. Term weight is cubed to eliminate the problem of a substring containing many small low scoring terms out scoring one extremely good term. This provides the one best answer substring; however, for TREC-9 five answer substrings are required. A goal of the MultiText QA system is to provide five different answers for each question.

3.6 Finding Multiple Answers

To accomplish the goal of having five different answers, the weight of any term matching a term in the answer substring selected is reduced. Terms' weight in the answer substrings are reduced to zero. Terms vital to answering the question are

reduced by half; monetary units (dollar, pound, etc.) for the MONEY category and distance units (miles, feet, meters, etc) for the DISTANCE category. Otherwise, any term matching a term in the answer substring has its weight reduced to zero.

Answer substring selection continues until the number of desired substrings is fulfilled. Reducing the terms' weight to zero allows for distinction between each of the answers, eliminating answers that are almost the same.

3.7 Putting the Pieces Together

The algorithm performs a linear number of calls to the database to lookup corpus term frequency. It also requires a linear number of dictionary lookups to find the number of passages that a term is in. Finally the algorithm's sliding window for answer selection is done in linear time for each answer required.

The full algorithm proceeds as follows:

Inputs:

$P = p_1, p_2, \ldots, p_m$ - a set of passages of size m and $p_i = t_1, t_2, \ldots t_n$

$Q = q_1, q_2 \ldots q_o$ - the question terms

$C = z_1, z_2, \ldots, z_x$ - the corpus

qc - the question category

M - the number of answers required

len - the length of the answer substrings

$L = l_1, l_2 \ldots l_y$ - a list of common words.

Output:

$A = \{a_1, a_2, \ldots a_M\}$ - the ranked answer substrings

$D = \{d_1, d_2, \ldots d_M\}$ - the document identifiers containing the answer substrings

Functions:

$f_t = |\{i|t = z_i\}|$

$c_t = |\{P|\exists_i.t = p_i\}|$

$PosHeur_t = (1 - \frac{1}{1000 - d_t})$

$RankHeur_t = (1 - \frac{1}{1000 - r_t})$

$CatHeur_t = EvalHeuristic(t, qc)$

$p_{i,j} = \{t|p_i = p_x \& t_j = t_y\}$

$start(p_{i,j})$ is the starting character of the term $p_{i,j}$ in p_i

$end(p_{i,j})$ is the ending character of the term $p_{i,j}$ in p_i

Algorithm 1 RITF

$Select(P, m, qc, M, len, Q, L) \equiv$
for $r \leftarrow 1$ to M **do**
  $bestScore \leftarrow 0$
  for $i \leftarrow 1$ to m **do**
    for $j \leftarrow 1$ to $|p_i|$ **do**
      $k \leftarrow j$
      while $k < |p_i|$ and $end(p_i, j) - start(p_i, j) + 1 \leq len$ **do**
        $k \leftarrow k + 1$
      end while
      if $j < k$ **then**
        $score \leftarrow 0$
        for $z \leftarrow j$ to $k - 1$ **do**
          let t be $p_{i,z}$
          if $t \notin usedTerms \cup Q \cup L$ **then**
            $score \leftarrow score + (c_t log(N/f_t) * PosHeur_t * RankHeur_t * CatHeur_t)^3$
          end if
        end for
        if $score > bestScore$ **then**
          $i' \leftarrow i$
          $j' \leftarrow j$
          $k' \leftarrow k$
        end if
      end if
    end for
  end for
  for $z \leftarrow j$ to $k - 1$ **do**
    $usedTerms \leftarrow usedTerms \cup (p_{i',z})$
  end for
  for $z \leftarrow 1$ to len **do**
    $a_r[z] = p_{i'}[k' + z]$ as the r-ranked answer snippet
  end for
  $d_r = i'$ as the document identifier
end for

Chapter 4

Results

In the TREC-9[9] experiments the system was one of the top performers, achieving second and third best MRR results in the 50-byte and 250-byte tasks, respectively. The experiments described in this chapter illustrate the contributions of RITF to these results. It is also important to examine the contribution of various elements of this component. The elements analyzed include the Redundant Inverse Term Frequency formula and the location and category heuristics. Different answer categories are easier to answer than others so an analysis has been done on a category basis. The system's performance is also affected by the size to which passages were expanded about the cover. For TREC-9 the algorithm used very simple patterns. Small changes to these patterns can have an effect on the overall system.

4.1 TREC-9

TREC-9 QA track required the systems to find solutions to 682 questions. Questions were guaranteed to have an answer that could be found in the 3 GB corpus comprised of newspaper articles. The systems were evaluated using TREC's standard measure of mean reciprocal answer rank (MRR). Each participant ran their system against the questions, and returned a ranked list of five answer substrings for each question. NIST judges used two types of judging: strict and lenient. For a strict judgment the document had to support the answer substring. The lenient judgment only had to have the correct answer substring. Post-hoc analysis were completed using the answer patterns and judging scripts supplied by NIST.

Official TREC-9 Result

The MultiText question answering system was submitted to TREC-9 for evaluation; in the 50-byte task the system achieved the second best results among the systems evaluated. The TREC-9 judges evaluated two runs per answer length per group as seen in Table: 4.1. The only difference in MultiText's runs with the same answer length was the answer extraction component. Runs *uwmt9qas0* and *uwmt9qal0* are the runs using the RITF. Runs *uwmt9qas1* and *uwmt9qal1* use an enhanced version of the answer extraction method employed at TREC-8. Table: 4.1 shows the MultiText QA TREC-9 runs.

To evaluate the accuracy of judging script, MultiText performed manual and

run id	length (bytes)	NIST strict judgements	NIST lenient judgements	MultiText manual judgements	NIST script judgements
uwmt9qas0	50	0.321	0.339		
uwmt9qas1	50	0.257	0.264		
uwmt9qal0	250	0.456	0.475	0.486	0.506
uwmt9qal1	250	0.460	0.465	0.456	0.470

Table 4.1: Official TREC-9 Results

scripted judgments on the results for comparison also seen in Table: 4.1. The evaluation script produces MRR values that are comparable to those produced manually. It should be noted that the TREC-9 script does produce slightly higher MRR than a judge.

Figure: 4.1 demonstrates how the system performed among the top ten TREC-9 question answering systems for the 50-byte run and Figure: 4.2 shows the results for the 250-byte run. The bars represent MRR and the dots are the number of questions with no correct answer.

Though MultiText performed well, there are many systems with similar results. This is fairly common with retrieval systems in that the next best system is only slightly better than a lower performing system. It is important to note that MultiText was well above the average MRR of 0.218 and 0.350 for the 50- and 250- bytes respectively.

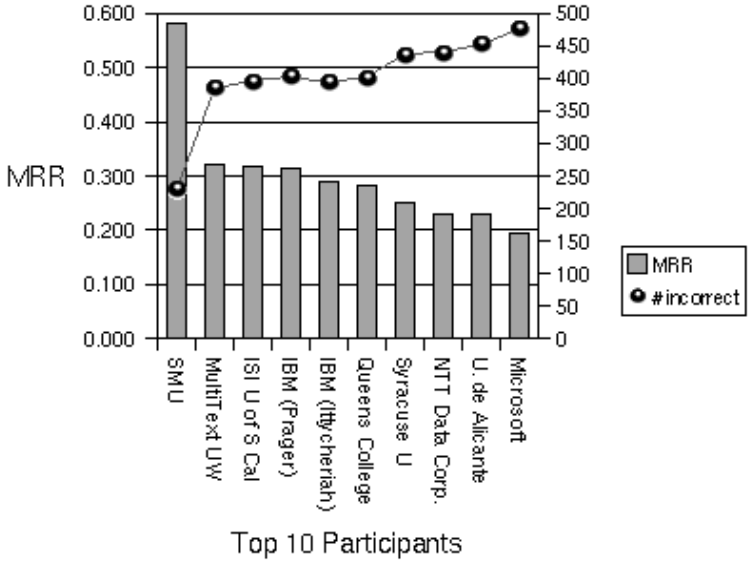

Figure 4.1: TREC-9 50-byte Results

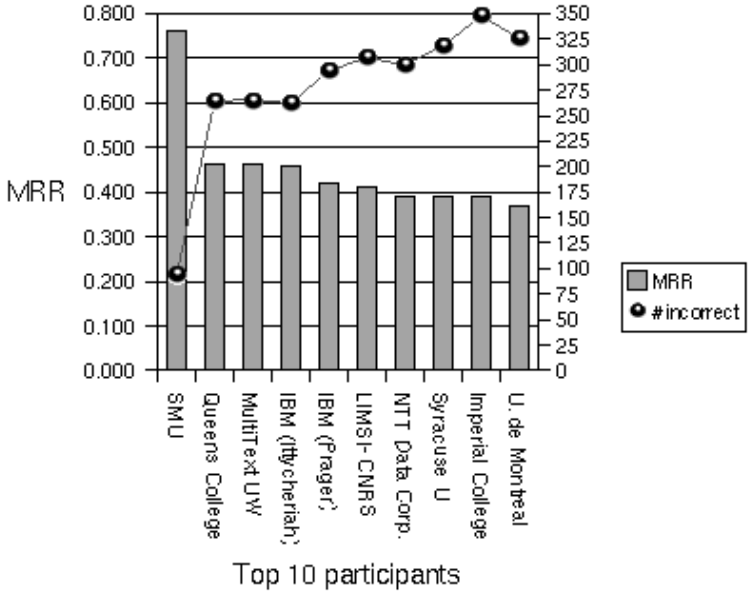

Figure 4.2: TREC-9 250-byte Results

Southern Methodist University outperformed all of the other systems by a significant margin.

4.2 Overall Improvement

While the TREC-9 results indicate the overall performance of the system, they do not directly indicate the contribution of RITF and its components. To evaluate these contributions several more experiments were conducted; the TREC-8 system was used as a baseline. It was not possible for the NIST judges to evaluate the baseline; therefore, to measure the improvement of the system the TREC-9 evaluation script was utilized. With the combination of the parser-generated queries and the answer extraction components, there is a total improvement in MRR from .189 to .377 (99%) and from .407 to .504 (24%) for 50- and 250-byte runs respectively. It is not surprising that the answer extraction component has a greater impact when the answer substring requirement is shorter. Table: 4.2 reveals the improvements that each component makes to the system.

RITF without parser assisted IR[1] improves 50-byte MRR from .189 to .357 (89%). The combination of the two new components is greater than either individually.

[1]Note that RITF takes question categories from the parser, but the parser generated IR query terms were discarded.

Improvements	50-byte answer MRR		250-byte answer MRR	
Baseline (none)	0.189		0.407	
Parse-generated Queries	0.191	(+1%)	0.464	(+14%)
Baseline + RIFT	0.338	(+78%)	0.462	(+13%)
Baseline + RITF + parser categories	0.357	(+89%)	0.467	(+15%)
TREC-9 system	0.377	(+99%)	0.504	(+24%)

Table 4.2: TREC-9 Evaluation Script Results

4.3 Question Category Performance

The MultiText question answering system has relatively few question categories. Each category has to include a broad range of questions, and there are many questions the system cannot classify. Table: 4.3 shows the number of TREC-9 questions for each category and the mean reciprocal rank of that category.

The MRR was produced by the TREC-9 script. The number of incorrect results for the different question categories varies tremendously. The system performs very well on the PROPER and PLACE categories. The PLACE category achieved the highest result because it is the only category to use an external source for information. As mentioned, it has a database of world cities and countries as well as U.S. states. Answer quality improves for answers that are more likely to occur in the corpus such as PROPER, PLACES and DATE. Analysis shows that there may be many answers for a MEASUREMENT or NUMBER classification but there

Category	# of question	MRR	# incorrect
Proper	202	0.492	82
Place	96	0.535	33
Date	67	0.357	39
Measurement	16	0.198	11
Number	36	0.273	21
Distance	0	0.000	0
Money	2	0.250	1
Other	263	0.259	155
Total	682	0.377	342

Table 4.3: Question Category Results

are usually very few that are exactly the same. Therefore the redundancy factor cannot be exploited, producing poor results for those categories.

Improving Patterns

The question patterns employed in the TREC-9 system did not handle punctuation. A slight modification was made to the patterns so that excess punctuation would be stripped away when finding candidate terms. This unsophisticated change resulted in a noticeable improvement.

With such minor changes in patterns leading to improvements it should be possible for high quality patterns to produce even better results. Also, as more

		Old Patterns	New Patterns	
Category	# of questions	MRR	MRR	Improvement
Proper	202	0.492	0.502	2.03%
Place	96	0.535	0.547	2.24%
Other	263	0.259	0.286	10.42%
Total	682	0.377	0.391	3.71%

Table 4.4: Effect of Improving Patterns

categories are added, it will be easier to create a more specialized pattern. As a result, the quality of patterns will also be improved. The change to these categories results in an advancement to the MRR from 0.377 to 0.391 for a gain of 3.7%. For all other analysis tests these new patterns are used.

4.4 Answer Extraction Elements Evaluation

There are several different parts to the answer extraction component. Each of these parts improves the system by varying degrees. Through the process of removing each element from the system the effect that the different elements has on the overall system can be seen. Table: 4.5 shows the MRR and the number of incorrect answers for the system without the various elements of:

$$w_t = c_t log(|C|/f_t) * PosHeur_t * RankHeur_t * CatHeur_t$$

From examining this Table: 4.5 it is apparent that the RITF formula is almost

Missing Element	w_t Element	MRR	#incorrect	Effect
RITF	$c_t log(\|C\|/f_t)$	0.195	475	-50.2%
Term Rareness Score	$log(\|C\|/f_t)$	0.345	368	-11.8%
Term Redundancy Score	c_t	0.241	434	-38.4%
Position Heuristic	$PosHeur_t$	0.379	354	-3.07%
Rank Heuristic	$RankHeur_t$	0.384	341	-1.79%
Category Heuristics	$CatHeur_t$	0.390	339	-0.25%
Complete Formula		0.391	338	0%

Table 4.5: Answer Extraction Elements Analysis

fully responsible for extracting the answers. Redundancy makes the most significant difference but the combination of the two produces a greater mean reciprocal rank. The heuristic makes very little contribution; however, it does assist in the process.

Redundant Inverse Term Frequency Analysis

In a large corpus there is duplicate or supporting information for almost any given question. The RITF formula utilizes this knowledge, through two simple premises: the more a term is repeated, the more likely it is the correct answer, and the lower the probability that a term appears by chance, the more likely it is correct.

Several formulas were used to evaluate a term's value. Here are the effects of variants of the term evaluation formulas tested:

In Table: 4.6 the f_t is the number of times the term is in the corpus, c_t is the

Term Evaulation Formula	Description	MRR	#incorrect
$\lambda_t = c_t log(\|C\|/f_t)$	RITF	0.391	338
$w_t = c_t$	Voting Scheme	0.345	368
$\lambda_t = log(\|C\|/f_t)$	ITF	0.241	434
$\lambda_t = c_t(\|C\|/f_t)$	non logarithic RITF	0.138	505
$\lambda_t = \frac{c_t}{1+c_t} log(\|C\|/f_t)$	Modified Redudancy Factor	0.286	425
$\lambda_t = \|C\|/f_t$	non logarithic ITF	0.121	528

Table 4.6: Different Term Evaluation Formulas

number of times the term is in the set of passages, and $|C|$ is the total number of terms in the corpus.

Analysis of the duplication component in the RITF formula is done by modifying its level of importance to the formula. This is accomplished by adjusting the size of α in this modified RITF formula:

$$w_t = c_t^{\alpha} log(N/f_t) * PosHeur_t * RankHeur_t * CatHeur_t$$

Figure: 4.3 helps determine how the RITF formula can be improved. By setting the formula ($\alpha = 0$) no redundancy will be taken into account in the calculation. When this occurs, the system only achieves a mean reciprocal rank of 0.241, demonstrating the importance of redundancy. The question that has to be examined is the exact significance of redundancy. If the duplication element is too much or not enough of a factor in the calculation, system performance will suffer.

The graph reveals that α should be in the range of 1 to 2. As expected, the

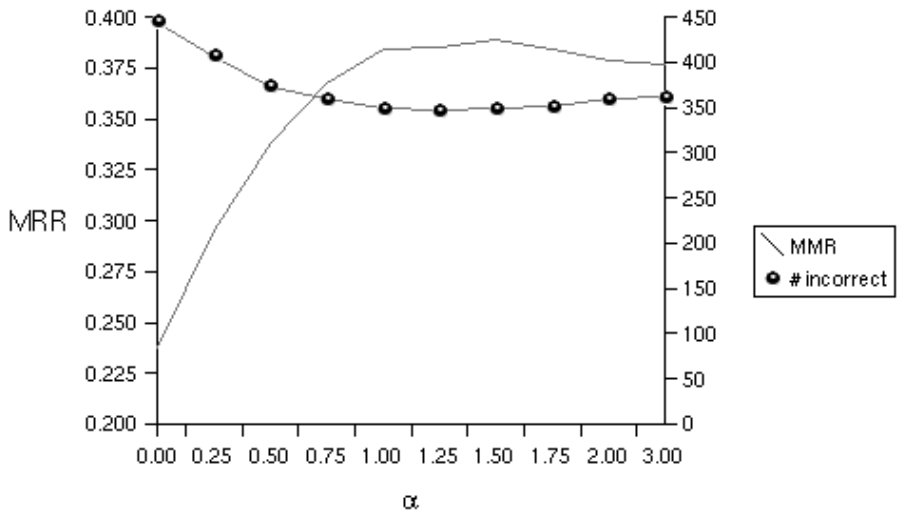

Figure 4.3: Redundancy Factor in RITF formula

graph demonstrates that the value of this part of the formula reaches a maximum before decreasing the system's overall accuracy. There is not enough evidence that the system would be improved if the RITF formula should use a different value then $\alpha = 1$ because the performance is very similar to the best value for α .

4.5 Location and Category Heuristic Evaluation

The addition of heuristics made a small improvement to the performance of the system. Notably, the position and rank heuristic did not have a significant impact on the system given the results in Figure: 4.4. The figure shows answer position relation to the passage; it can be seen that most answer locations are in the centre of the passages. Figure: 4.4 shows answer placement from all the first ranking

passages for the TREC-8 question. There is a distinct spike of answers in the centre of the passages which suggest answers are much more likely to be found near the cover. Heuristics were hand coded based on TREC-8 question.

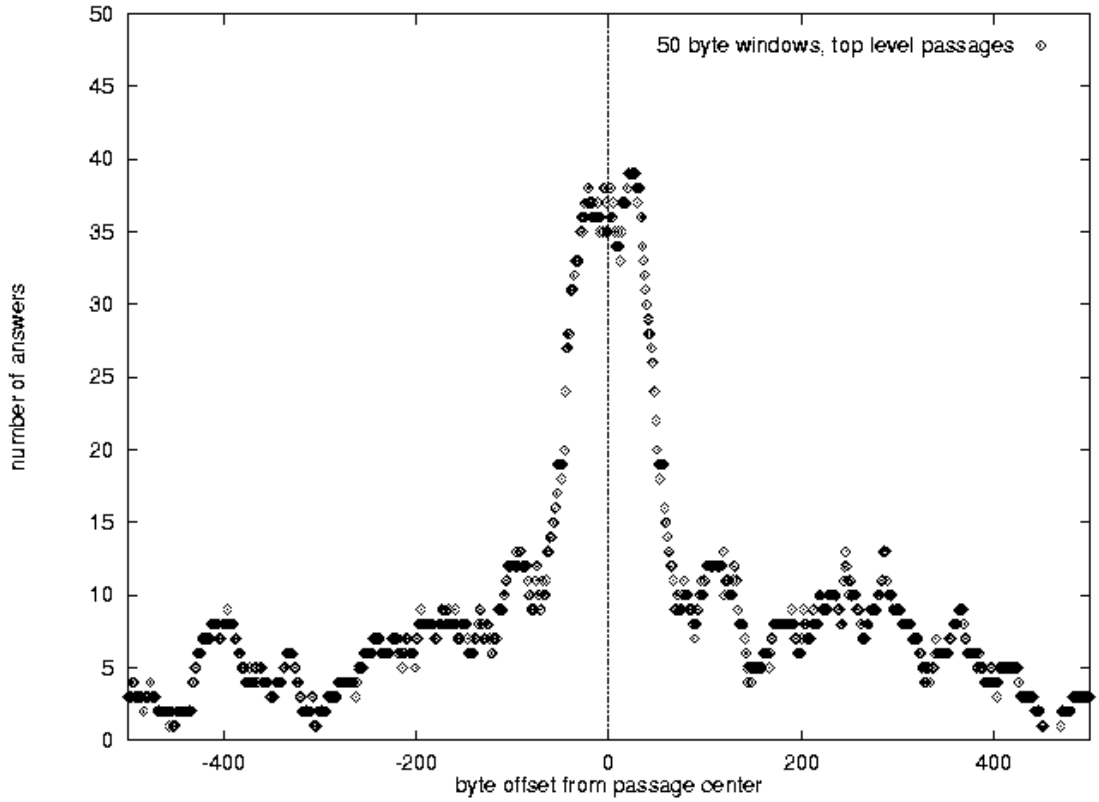

Figure 4.4: Answer Position Relative to Passage Centre[4]

Position Heuristic

It has been shown that far more answers come from the centre of passages then the ends. This can be seen in Figure: 4.4 where answer location makes a standard

distribution around the centre of the passages. The evaluation of position heuristic is done by modifying β in the following formula.

$$PosHeur_t = 1 - \frac{1}{\beta - d_t}$$

Figure 4.5 shows the results for the determination of the best value of β. In the system evaluated for TREC-9 $\beta = 1000$. The system improves slightly when β

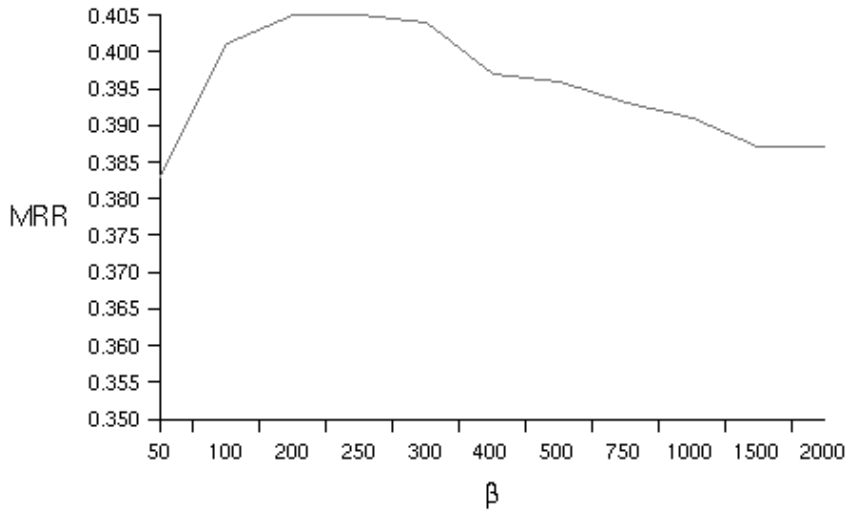

Figure 4.5: Position Heuristic Analysis

ranges from 100 to 300. Though there is only a small improvement, it is enough that β should be set to 250 for future use.

Rank Heuristic

All information retrieval systems use some method of ranking the retrieved information. Ranking the information is important because it allows a system to

determine what information is more valuable. It follows the principle that, in general, the higher the rank the better the information. Under this simple premise, one would presume that during the answer extraction process the rank of the passages a candidate answer comes from would make a substantial difference. This does not seem to be true based on the fact that the rank heuristic only improves the system 1.8%. The rank heuristic is the following formula:

$$RankHeur_t = 1 - \frac{1}{\beta - r_t}$$

where for TREC-9 evaluation $\beta = 1000$.

It is possible that heuristic does not have enough effect on the term's weight. An analysis can be done by modifying β. One would assume that a small value of β would improve the system

The Figure: 4.6 was created by evaluating the system using different values of β.

Notably, the more effect the rank heuristic had on the system the lower the overall performance. However, the performance hardly changed demonstrating that the rank a candidate answer is taken from is not important. There are a few reasons for a candidate term's passages rank not making a major difference. Many of the passages will have the same rank for a question. A lower ranked passage might contain two candidate terms that are needed to answer the question but the rank heuristic decreases the term weight too much. The answer extraction component is better at finding answers to the question then an information retrieval component.

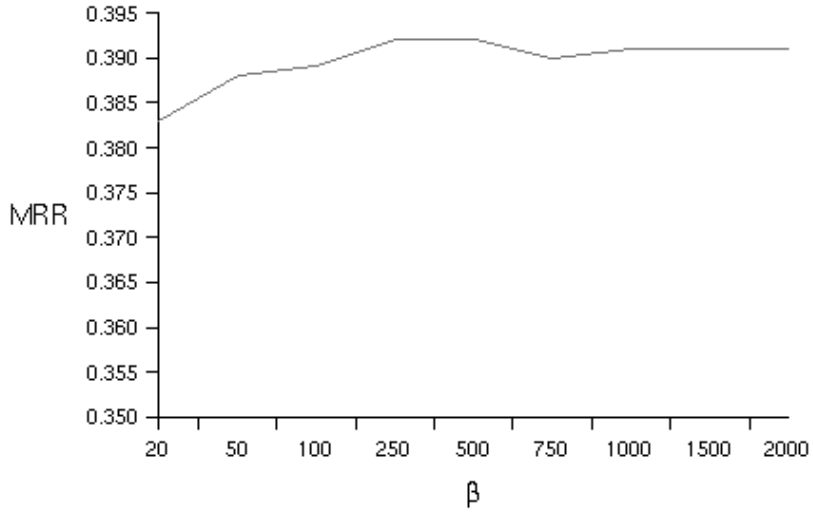

Figure 4.6: Rank Heuristic Analysis

The rank heuristic makes a very small contribution to the system but it is unclear that any changes to it would improve the system.

Category Heuristics

The category heuristics made no significant improvement to the system. These heuristics were derived from training on the TREC-8 question. The heuristics did make a fair amount of improvement on the training questions. For TREC-9, together, they made no difference. It is likely that the heuristics were over-trained to the TREC-8 questions. The only category heuristic that contributed a noticeable difference was the PLACE heuristic improving question of that type by 9.6%. This heuristic worked on a different principle than the others. It used an external database to determine if the term was a known place. The database contained

major world cities, countries, and US states. The other category heuristics were based on writing practices (i.e. places are more likely capitalized). Future heuristics should be based further on exploiting information from external sources. Many systems use WordNet[18] to improve their performance. Due to the little effect that category heuristics had, all but the one applying to PLACE should be eliminated or improved.

4.6 Cover Expansion

It has been determined that answers tend to fall closer to the centre of a passage. This means that information further from the passage's centre is of less importance than information closer to it. A passage is a cover that has been expanded by adding text to either side. The cover expansion experiment is used to determine the effect of adding different lengths of text to either size of the cover. Figure: 4.7 shows the system's performance using different cover expansion lengths.

The information just outside the cover is very important. Extending the cover by 25-bytes changes the MRR from 0.183 to 0.372. Larger cover expansion does not make as great an improvement. The graph reveals that for TREC-9 questions the best cover expansion is between 150 and 200 bytes. Extending the cover by a large value does not significantly reduce performance.

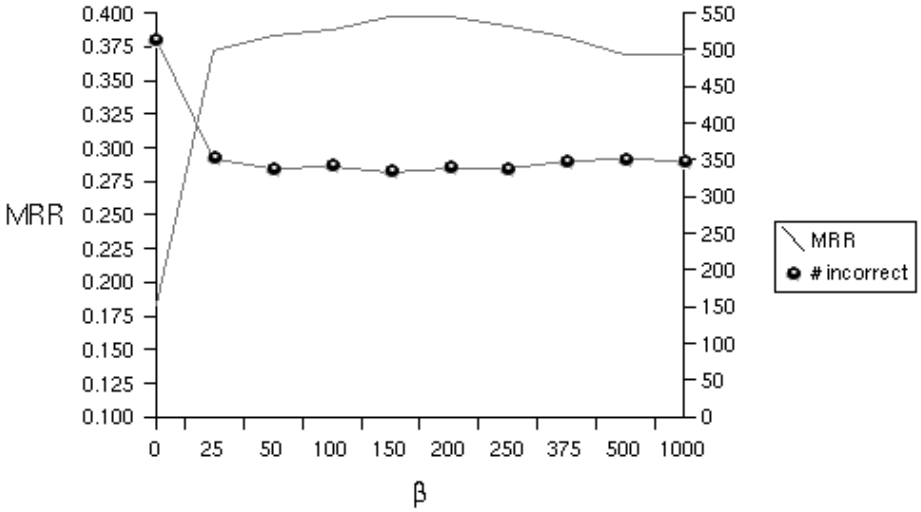

Figure 4.7: Cover Expansion Analysis

4.7 Depth of Passages

Tests were also complete to determine if the number of passages used to find the answer effected the system. Figure: 4.8 shows the effect of using 1 to 1000 passages, these passages are not exactly the same as the ones used for TREC-9 though the process of retrieval was similar. When only a small number of passages were used, system performance decreased. This is expected because there would be fewer redundant terms. The use of too many passages also decreases system performance. For future purposes, between 10 and 20 passages should be used.

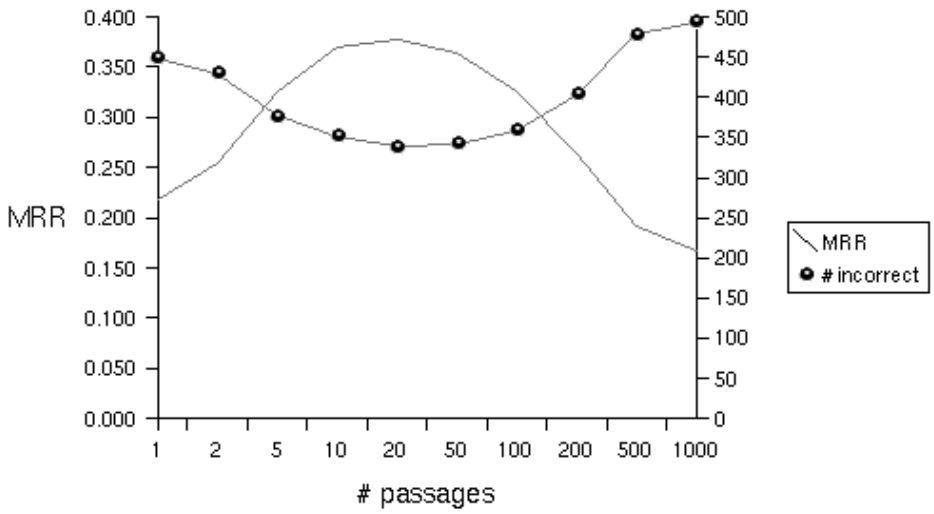

Figure 4.8: Depth of Passages Analysis

4.8 Misclassification

Classification of a question can have a largely impact on the system. Table: 4.7 shows the effect of misclassification. Each row is the classification given by the parser for TREC-9. The number of question for each classification depends on the parser. There was no questions classified as DISTANCE make misclassification not applicable NA. Columns show the effect of reclassification from the row to the column classification. The system performs well with no classification achieving an overall MRR of 0.344 by defining all questions as OTHER. However, when questions are misclassified the system can perform as poorly as a MRR of 0.065. Misclassification has the largest impact when the correct answer is a word but the system is looking for candidates that are numbers.

TREC-9	MRR							
PROP	0.502	0.463	0.116	0.047	0.048	0.060	0.061	0.501
PLACE	0.479	0.547	0.085	0.096	0.099	0.087	0.100	0.494
DATE	0.093	0.073	0.357	0.050	0.072	0.110	0.134	0.081
MEAS	0.000	0.000	0.028	0.198	0.210	0.174	0.214	0.125
DIST	NA	NA	NA	NA	0.000	NA	NA	NA
MON	0.000	0.000	0.250	0.125	0.250	0.250	0.000	0.000
NUM	0.049	0.047	0.028	0.238	0.257	0.219	0.273	0.090
OTHER	0.205	0.195	0.031	0.040	0.052	0.055	0.061	0.286
ALL	0.307	0.299	0.096	0.065	0.075	0.078	0.088	0.344
Classified	PROP	PLACE	DATE	MEAS	DIST	MON	NUM	OTHER

Table 4.7: Misclassification Analysis

4.9 Results Summary

Extensive analysis was done to determine what each element contributed to the overall system. Testing showed that the key improvement to the system was the Redundant Inverse Term Frequency formula. Though many different term evaluation formulas were examined, RITF formula significantly outperformed them all. The location and category heuristics also contributed to the performance of the system. Cover expansion tests show that the quality of information obviously makes a difference in how the answer extraction component performs. The addition of poor quality information can negatively impact the system. Results reveal that the Redundant Inverse Term Frequency formula creates an excellent basis for a question answering system.

Chapter 5

Conclusion

Overall, the addition of the answer extraction component improves the question answering system. Not only does the AE component achieve solid results, it is also a good basis for a more advanced question answering system.

The QA system is similar to other top performing systems in its approach to solving the question answering problem. First the questions are analyzed to determine question type and focus. This information is then used to generate queries for a passage retrieval component. Passages returned from the retrieval engine are passed to a answer extraction component. Next, candidate answers are extracted from the passages and are then evaluated using the RITF formula and other heuristics. Finally, a string containing the best candidate answer is outputted.

A major difference that the MultiText QA system has from other question answering systems is in its answer extraction methods. MultiText QA uses only

statistical approaches for answer extraction.

The idea behind the RITF component is to find and evaluate potential answers in the passages retrieved using the RITF Formula: 3.1. The RITF formula's use of corpus term frequency in conjunction with a voting scheme allows it perform well.

The RITF component still performs well without question analysis.The RITF algorithm does not need a question classification because it can treat all questions as OTHER. Without question classification the system can achieve a mean reciprocal rank of 3.38, which would out perform all but a few QA systems at TREC-9. This creates a very robust method to extract answers; though having question categories does improve the system's mean reciprocal rank considerably. With the addition of question classification, the system can apply specialized candidate finding patterns. Also, category based heuristics can be exploited. The overall system can achieve MRR of 3.91.

Notably, because the RITF algorithm does not require information regarding the structure or grammar of a natural language, the algorithm should prove useful in many natural languages. The RITF algorithm can even extract answers when the question's meaning is completely unknown. Having an elementary and reliable way to evaluate each term in a set of passages is the key to the answer extraction component.

Preliminary testing indicated that a correct answer location tends to be close to or within the query cover. Preliminary testing also showed that candidate answers from higher ranking passages are usually more likely to contain an answer than

lower ranking passages. For this reason, rank and position heuristics were created. These heuristics made less of an impact on the system then expected. There was enough performance gain to justify the use of term location heuristics; though more research should be done to increase the performance of these heuristics.

Candidate term evaluation using question category specialized heuristics made only a small contribution. Over-training is the main reason these category heuristics do not perform better. Category heuristics based on external knowledge did improve the system. Performance improvements can be seen with the addition of a known place database. The PLACE category heuristic is based on knowledge and out performs those base on simple writing style based rules. Therefore these writing based rules should be replaced with heuristics that are more knowledge based.

The MultiText QA system achieved second place overall at TREC-9. Although the RITF statistical approach we employed did not perform as well as Southern Methodist University the top performing system which combined knowledge based and NLP techniques, it is still very useful. When no information about the question is known, a statistical approach can perform well; it can be language independent. Combining statistical, knowledge based and NLP techniques should greatly enhance the question answering process.

The RITF algorithm is also scaleable because in theory, as the corpus size expands, the performance of the system should increase as more duplicate information will become available. Finally, the initial value of the redundant inverse term frequency algorithm is beneficial to the overall system and future applications of

question answering.

Chapter 6

Future Work

This system creates a strong basis on which question answering can continue to develop. It is the basis for the system submitted to TREC-10 where, once again, it was one of the top performing systems.

A major addition to the system includes finding answers in web data; this was the most prevalent improvement to the overall system. Currently, researchers in the group are examining the use of a tera-byte of data for question answering.

Next, the exploitation of WordNet to assist in finding definition answers in the corpus was added. It is also used for knowledge based category heuristics. Once again, better patterns were developed which also improved the system.

Another feature that was implemented was a feedback loop that included adding highly weighted terms to the original query. This was not successful because the system returned lists of possible answers as its output substrings; this made it

impossible to recognize the correct answer. However, it did prove promising for many of the questions but further research in this area is required.

There are a number of areas in which the system can still be expanded to improve performance. Name Entity tagging must be incorporated further into the system[11]; this is a major goal of future work. Also, a process to relate question parses with candidate passage parsing is needed in the system. The use of more external knowledge sources could also prove beneficial.

A research goal is to develop a standard question answering platform available to the public that is derived from the MultiText QA system.

Bibliography

[1] Richard Beckwith, George Miller, and Randee Tengi. Design and implementation of the WordNet lexical database and searching software. Included in WordNet software distribution. See http://www.cogsci.princeton.edu/ wn.

[2] Chris Buckley, Mandar Mitra, Janet A. Walz, and Claire Cardie. SMART high precision: TREC 7. In *Text REtrieval Conference*, pages 230–243, Gaithersburg, MD, 1998.

[3] Nancy A. Chinchor. Overview of MUC-7. In *Seventh Message Understanding Conference (MUC-7)*, 1996.

[4] C. L. A. Clarke, G. V. Cormack, D. I. E. Kisman, and T. R. Lynam. Question answering by passage selection. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[5] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *SIGIR Conference 2001*, New Oreans, Louisiana, 2001.

[6] Micheal Collins. A new statistical parser based on bigram lexical dependencies. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco, 1996.

[7] C.L.A. Clarke G.V. Cormack. Interactive substring retrieval (MultiText Experiments for TREC-5). In *5th Text REtrieval Conference*, Gaithersburg, MD, 1996.

[8] G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. Fast automatic passage ranking. In *8th Text REtrieval Conference*, Gaithersburg, MD, November 1999.

[9] D. Harman E. Voorhees. Overview of the Ninth Text REtrieval Conference (trec-9). In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[10] J. Earley. An efficient context-free parsing algorithm. Communications of the ACM, 13(2):94–102, 1970.

[11] M. Fuller, M. Kaszkiel, S. Kimberley, J. Zobel (RMIT), C. Ng (RMIT, Sharp Laboratories of Europe Ltd.), R. Wilkinson (CSIRO), M. Wu (RMIT, and CSIRO). The RMIT/CSIRO ad hoc, q&a, web, interactive, and speech experiments at TREC 8. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[12] Sanda M. Harabagiu and Steven J. Maiorano. Finding answers in large collections of texts: Paragraph indexing + abductive inference. In *1999 AAAI Fall*

Symposium on Question Answering Systems, pages 63–71, North Falmouth, MA, 1999.

[13] D. Harman. Overview of the First Text REtrieval Conference (TREC-1). In *First Text REtrieval Conference (TREC-1)*, Gaithersburg, MD, 1992.

[14] Ulf Hermjakob and Raymond J. Mooney. Learning parse and translation decisions from examples with rich context. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–489, Somerset, New Jersey, 1997. Association for Computational Linguistics.

[15] Eduard Hovy, Ulf Hermjakob, Chin-Yew Lin, Mike Junk, and Laurie Gerber. Question answering in Webclopedia. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[16] D. Hull. Xerox TREC-8 question answering track report. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[17] T. R. Lynam, C. L. A. Clarke, and G. V. Cormack. Information extraction with term frequencies. In *Human Language Technology Conference 2001*, San Diego, Ca, 2001.

[18] G.A. Miller. WordNet: A lexical database. In *Comunication of the ACM, vol 38: No11*, pages 39–41, 1995.

[19] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. Lasso: A tool for surfing the answer net. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[20] T. Morton. Using coreference in question answering. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[21] Marius Pasca and Sanda Harabagiu. High-performance question/answering. In *SIGIR Conference 2001*, New Oreans, Louisiana, 2001.

[22] J. Prager, D. Radev, E. Brown, and A. Coden. The use of predictive annotation for question answering in TREC8. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[23] John Prager and Eric Brown. One search engine or two for question-answering. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[24] Dragomir R Radev, John Prager, and Valerie Samn. Ranking suspected answers to natural language questions using predictive annotation. In *6th Conference on Applied Natural Language Processing*, Seattle, May 2000.

[25] P. Morarescu S. Harabagiu D. Moldovan M. Pasca R. Mihalcea M. Surdeanu R. Bunescu R. Grju V. Rus. Falcon: Boosting knowledge for answer engines. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[26] A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle, and F. Pereira.

AT&T at TREC-8. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[27] R. Srihari and W. Li. Information extraction supported question answering. In *8th Text REtrieval Conference*, Gaithersburg, MD, November 1999.

[28] E. Voorhees. The TREC-8 question answering track evaluation. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[29] E. Voorhees. The TREC-8 question answering track report. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[30] E. Voorhees. Overview of the TREC-9 question answering track. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[31] E. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[32] E. Voorhees and D. Tice. Building a question answering test collection. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 2000.

[33] Ian H. Witten, Alisair Moffat, and Timothy C. Bell. *Managing Gigabytes Compressin and Indexing Documents and Images.* Morgan Kaufmann, 1999.