Pauchok: A Modular Framework for Question Answering

by

Stefanie Tellex

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

Author .....
Department of Electrical Engineering and Computer Science
May 9, 2003

Certified by .....
Boris Katz
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Pauchok: A Modular Framework for Question Answering

by

Stefanie Tellex

Submitted to the Department of Electrical Engineering and Computer Science
on May 9, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

An ideal question answering system would be able to answer any question that a human could answer. Currently this dream is unattainable without major advances in artificial intelligence. However, we can advance towards this dream today by focusing on factoid questions that can be answered by short phrases. The Question Answering tracks at recent Text REtrieval Conferences (TREC) provide a forum for evaluating and comparing factoid question answering systems. Many systems entered in TREC have the same general architecture, but they are evaluated only as black box systems. This thesis describes Pauchok, an extensible playground that enables question answering components to be quickly implemented and tested. I implemented a number of passage retrieval algorithms within Pauchok's framework. I compared the performance of these passage retrieval algorithms, and found three important things. Boolean querying schemes perform well in the question answering task. The performance differences between various passage retrieval algorithms vary with the choice of document retriever, which suggests significant interactions between document retrieval and passage retrieval. The best algorithms in our evaluation employ density-based measures for scoring query terms. These results reveal future directions for passage retrieval and question answering.

Thesis Supervisor: Boris Katz
Title: Principal Research Scientist

Acknowledgments

This document originated from a paper to appear in SIGIR 2003 [29], co-authored with Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes.

I could not have completed this thesis without help. Here is an incomplete list of those to whom I owe thanks.

- Boris, for his guidance over the past two years.
- Jimmy and Greg for all the admonishments, discussions, and advice.
- Aaron for helping build the system.
- Everyone who read drafts.
- Eric, for putting up with all the stressful days and late nights.
- Alisa, Amy, Carie, Dave, Lin, and Piotr for keeping me sane.
- My family.

Dedication

To my cat Licorice.

"If man could be crossed with the cat it would improve man, but it would deteriorate the cat." - Mark Twain

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A huge amount of data is accessible on the World Wide Web. Google indexes over 3 billion web pages — so many that it would take nearly 1000 years to view each one for 10 seconds. Yet this information is useless if it cannot be retrieved when users need it. Keyword searching has emerged from the past forty years of research into this information retrieval (IR) problem. In this model, users formulate a query, send it to the IR system, and receive a collection of documents matching the query. Users then read document summaries to filter the resulting collection. Finally they must read the chosen documents in order to find the needed information.

Keyword searching has serious drawbacks when the user is searching for an answer to a factoid question such as "When did Hawaii become a state?" Although factoid questions can be answered by a short phrase, users must scan through lists of search results and must read entire documents to find the few sentences that matter.

A question answering interface addresses these problems. Users ask a question, and the system tries to find an exact answer to the question in a large corpus of documents. Asking questions does not require training; people ask questions all the time. An ideal question answering system would be able to answer any question that a human could answer. Currently this dream is unattainable without full text natural language understanding. However, we can advance towards this dream today by focusing on factoid questions that can be answered by short phrases.

This thesis describes Pauchok, a system that captures some of the abstractions

common to many factoid question answering systems. This work advances the state of the art in factoid question answering by providing a framework for quickly developing and evaluating new question answering components, and by comparing a number of algorithms used to retrieve passages from a corpus in response to a natural language question.

1.1 Motivation

The IR community has developed systems that seek to answer natural language questions from an unstructured corpus of natural language documents [9]. These systems focus on factoid questions, or questions that can be answered by a short phrase, often a noun phrase. For example, the question "Who was the first woman killed in the Vietnam War?" can be answered by the noun phrase "Sharon Lane". The Text REtrieval Conference (TREC) competition [31, 32] is a forum where many factoid question answering systems are evaluated and compared.

Although many of the systems evaluated at TREC have a similar architecture, they are evaluated as black box systems. When a system performs well, we often do not know which component is responsible for its good performance. Improving the state of the art in question answering requires good evaluations for each component, so that better components can be recognized and used to build better systems.

1.2 Approach

Pauchok is a modular question answering architecture that captures some of the key abstractions shared across top performing TREC systems. Pauchok provides a framework in which question answering components from diverse systems can be directly compared.

Pauchok instantiates a version of the standard question answering architecture [32, 31, 24, 9, 19]. As shown in Figure 1-1, Pauchok contains modules for question analysis, document retrieval, passage retrieval, and answer extraction. Question analysis

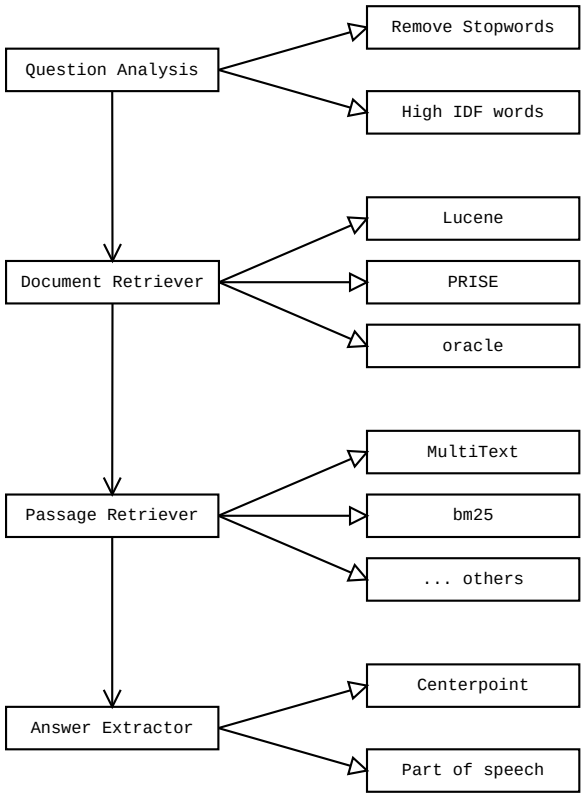


Figure 1-1: Pauchok's compile time structure. There can be many different implementations of each module on the left; this diagram only shows ones currently implemented.

analyzes the question for use by other modules. Document retrieval finds documents from the corpus that contain an answer. Passage retrieval searches for blocks of several sentences that contain an answer. Finally answer extraction identifies exact answer phrases in retrieved passages.

I evaluated a number of passage retrieval algorithms used in TREC 2001 [31] systems. This thesis presents the results of the evaluation and suggests research directions that could lead to improvements in the performance of passage retrieval algorithms.

I chose to evaluate passage retrieval over other key abstractions such as document retrieval and answer extraction for several reasons. Many passage retrieval techniques have been described in the context of improving document retrieval performance (e.g., [27, 4]), but I am not aware of any attempts to systematically study the performance of passage retrieval for question answering. Because most answer

extraction algorithms are currently computationally infeasible at the level of entire documents, current systems run them only on passages; as a result, if passage retrieval does not perform well, answer extraction is impossible. Furthermore, passages themselves form a very natural unit of response for question answering systems; Lin *et al.* [21] showed that users prefer passages over exact phrase answers in a real-world setting because paragraph-sized chunks provide context. For example, although the question "Who was the first woman killed in the Vietnam War" can be answered by "Sharon Lane", users may prefer the entire passage:

> A piece of steel from a rocket that landed between Ward 4A and Ward 4B of the 312th Evacuation Hospital had ripped through Sharon Lane's aorta. She bled to death less than a month before her 26th birthday, the first woman killed by hostile fire in Vietnam.

1.3 Contributions

This work contributes to the field of question answering in several ways.

- I synthesized a generic question answering architecture that serves as a playground for future research.
- I created an evaluation framework for components in the standard question answering architecture.
- I identified successful passage retrieval techniques.
- I suggest directions for the improvement of passage retrieval.

1.4 Organization

Chapter 2 reviews the standard question answering architecture and various other architectures for question answering systems.

Chapter 3 describes Pauchok's major abstractions: Question analysis, document retrieval, passage retrieval, and answer extraction.

Chapter 4 outlines the evaluation procedure. A number of passage retrieval algorithms were implemented in Pauchok's framework. I tuned them on the TREC 2000 data and tested on the TREC 2001 data.

Chapter 5 details the passage retrieval algorithms implemented for this evaluation.

Chapter 6 presents the results of the evaluation.

Chapter 7 discusses the results. Boolean querying schemes perform well in the question answering task. The performance differences between various passage retrieval algorithms vary with the choice of document retriever, which suggests significant interactions between document retrieval and passage retrieval.

Chapter 8 suggests future directions for passage retrieval and question answering.

Chapter 2

Related Work

This chapter discusses the standard approach to question answering and several alternate approaches. Then it describes initial work in passage retrieval.

2.1 Standard Question Answering Architecture

This section reviews various instantiations of the standard question answering architecture. Pauchok is a concrete implementation of this architecture. Pauchok's design is shown in Figure 1-1.

The TREC question answering track [32, 31] is a forum where question answering systems are evaluated and compared. A TREC system tries to find an answer to a factoid question in a large corpus of newspaper articles. Voorhees [32, 31] describes a generic question answering architecture that is implemented by most systems submitted to TREC. This architecture consists of question type classification, document retrieval, and named entity matching. First, systems identify the expected answer type of the question, in order to characterize the answer. For example, the answer type of "Who was the first governor of Alaska?" might be "person." Systems range from computing very broad to very specific answer types; there is no standard answer type ontology. Next, systems use some algorithm to retrieve passages from the corpus that are likely to contain an answer. Finally, they search the passages for entities matching the answer type of the question.

Hirschman and Gaizauskas [9] describe a generic architecture in order to frame their discussion of actual question answering systems. While Voorhees surveys existing question answering systems and generalizes an architecture, they describe a typical architecture in order to compare it to existing systems. Their system begins by analyzing the question. Subsequent processing modules have access to the results of this analysis. Document collection preprocessing and candidate document selection modules retrieve documents from the corpus likely to contain an answer. Candidate document analysis creates a useful representation of the document for answer extraction. Pauchok's passage retrieval module is a type of candidate document analysis. The answer extraction module retrieves an exact answer to the questions from the analyzed documents.

Light *et al.* [19] describe a more abstract model used to frame a component level analysis of the performance of various modules. Their model consists of document retrieval, sentence retrieval, and short answer extraction. Sentence retrieval maps to Pauchok's passage retrieval. They compare various kinds of word overlap scoring functions.

2.2 Other Question Answering Architectures

Other architectures for question answering systems have been proposed that approach the problem differently from the standard question answering architecture

Chu-Carroll *et al.* [5] describe a system consisting of question analysis agents, answering agents, and an answer resolution module. There can be multiple question analysis and answering agents that try to answer the question using different strategies. For example, one answering agent might answer questions by searching the corpus, while another searches for answers on the World Wide Web. The answer resolver combines answers from multiple systems and tries to determine the best one. Pauchok could be integrated into this architecture as an answering agent.

InsightSoft's TREC 2001 system [28] uses indicative patterns to identify answers. This system cuts the retrieved documents into passages around query terms, and

analyzes each passage for the presence of pattern. It associates each question type with human-generated patterns that signal the presence of answers.

The START [14, 17] system retrieves answers to questions from the World Wide Web. START makes use of Omnibase [15], an abstraction over knowledge sources. Omnibase provides a structured query interface to documents on the World Wide Web. It also contains natural language annotations describing the data in these documents. START parses the user's question and matches it against the annotations on the documents in order to find an answer through Omnibase.

Aranea [20] also searches for an answer to a question on the World Wide Web. It consists of two major modules: knowledge annotation and knowledge mining. The knowledge annotation module has a database of annotated web pages that contain answers to classes of TREC questions. To answer these questions, Aranea identifies the question as one that can be answered by one of its annotated pages, and then uses the annotations to extract an answer from the web page. This technique yields high precision but low recall. The knowledge mining module searches for an answer in text snippets retrieved via a web search engine. It tries to exploit the redundancy of the World Wide Web by searching for phrases repeated across multiple web pages. Because the TREC competition requires systems to find support for answers by returning a document from the provided corpus, Aranea also contains an answer projection module. Given an answer obtained from the web, the answer projection module tries to find a document supporting it in the corpus. Lin *et al.* [20] used Pauchok as Aranea's answer projection module in TREC 2002.

2.3 Stand-alone Approaches to Passage Retrieval

Salton *et al.* [27] describes an IR system that retrieves passages if the passage has a higher relevance score than the document; it returns a smaller snippet only if the similarity of the passage to the query is greater than the similarity of the entire document to the query. He describes several variations of word overlap term scoring with stemming to rank passages.

Callan [4] describes experiments in which he added passage retrieval to INQUERY, an IR system. INQUERY's overall goal is document retrieval; passages are used as evidence that a document is relevant. The scoring metric ranked passages based on a term frequency and inverse document frequency (*tf.idf*) term score.

A variety of passage retrieval techniques have been used in past TREC conferences. The ones chosen for this evaluation are described in Chapter 5.

Chapter 3

Pauchok's Design

I tried to design Pauchok to support any module from any question answering system that instantiates the standard question answering architecture. The system currently supports question analysis, document retrieval, passage retrieval, and answer extraction, four major components found in many TREC systems.

3.1 Design Overview

Data flow in Pauchok is shown graphically in Figure 3-1. Each module tries to identify the location of an answer at successively finer levels of granularity. The question analyzer analyzes the question to generate a query; this query is used to retrieve documents from the corpus. A passage retrieval algorithm analyzes the documents to extract passages from the corpus. (A different query can be used for document and passage retrieval.) Finally an answer extractor searches for an exact answer to the question in the retrieved passages.

3.2 Question Analysis

The question analysis module converts natural language questions into queries for the document retriever. This process can range in sophistication from simply returning the user's question as the query to employing sophisticated question analysis to gener-

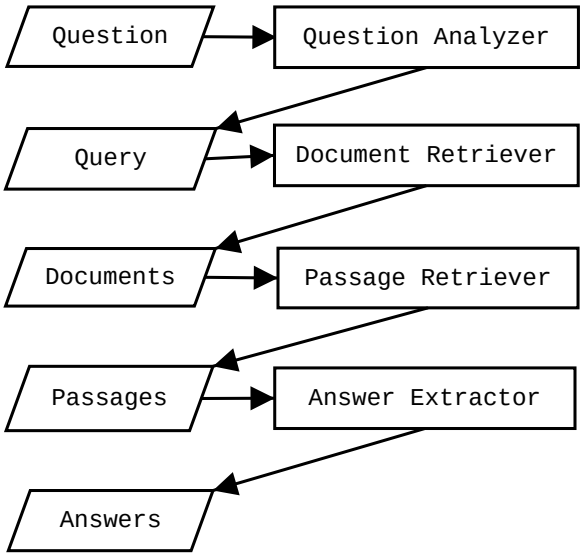


Figure 3-1: Data flow in Pauchok

ate complex, structured queries. Often, this module also detects the expected answer type of a question, e.g., the expected answer type of "When was Albert Einstein born?" is *date*. This information helps guide the answer extraction process.

3.3 Document Retrieval

The document retrieval module retrieves documents from the corpus that are likely to contain answers to the user's question. It consists of a query generation algorithm and a text search engine. The query generation algorithm takes as input the user's question and creates a query containing terms likely to appear in documents containing an answer. This query is passed to the text search engine, which uses it to retrieve a set of documents.

Pauchok currently supports the Lucene search engine [2], a freely available open-source IR engine. In addition, it can use the list of documents provided by NIST for each TREC question, generated using the PRISE engine [3]. It also supports an "oracle" document retriever that returns the documents NIST identified as relevant for each TREC question. Every document returned by the oracle document retriever is guaranteed to contain an answer to the question.

Lucene is the only Pauchok document retriever that may be used for generic questions; the other document retrievers rely on files provided by NIST for questions used in TREC. However, Pauchok supports the capability to add more document retrievers. I have begun adding support for Lemur [1].

3.4 Passage Retrieval

Passage retrieval algorithms take a document and a question and try to find passages from the document that contain an answer. Typical passage retrieval algorithms break the document into passages, compute a score for each passage, and return the passage with the highest score. The system abstracts this mechanism so that passage tokenizing algorithms and passage scoring algorithms can be modularized as well. Passage retrieval algorithms that cannot be broken down in this way are also supported.

Pauchok supports a large number of passage retrieval algorithms that are described in Chapter 5.

3.5 Answer Extraction

Finally, the system must extract an answer to the question from the passages. This module takes as input a passage from the passage retrieval component and tries to retrieve an exact phrase to return as an answer. This functionality typically requires parsing and detailed question analysis. Many systems extract answers by identifying entities in the passage that match the expected answer type of the question.

Two baseline answer extraction algorithms have been implemented within Pauchok. The identity answer extractor returns the center-point of the passage, stripping words from either end until it fits within the specified answer window. The part of speech answer extractor analyzes the question to determine the part of speech of the answer, and returns the first entity from the passage matching that part of speech.

Chapter 4

Procedure for Passage Retrieval Evaluation

In order to evaluate Pauchok, passage retrieval algorithms chosen from top performing TREC systems were implemented within its framework. The relative performance of these algorithms was evaluated and compared in order to suggest improvements for passage retrieval in question answering. The success of this implementation effort shows that Pauchok's architecture captures an abstraction shared across many question answering systems.

4.1 Implementation

Chapter 5 describes the algorithms used in detail. The algorithms were tested and trained using the data set provided by NIST for the TREC 2000 conference. All evaluation results are presented over the TREC 2001 data set. TREC 2000 and TREC 2001 used the same corpus of newspaper articles. NIST data for each conference included a list of questions, and for each question a list of answer patterns and a list of supporting documents. [31, 32]

For each answer, Pauchok outputted the score assigned by the passage retrieval algorithm, the supporting document number, and the answer text itself. Algorithms ran on the first 200 documents returned by the document retrieval module. For each

question, algorithms returned up to twenty passages of 1000 bytes each. In an end-to-end question answering system evaluation, an answer extraction algorithm would run on these passages to find an exact answer. The answer extractor would limit the results returned by the system; in TREC 2001, systems returned five answers of 50 bytes each.

Pauchok expanded or contracted the initial passage returned by the algorithm to fit within the space allotted, so that all evaluated passages had the same average length. If a passage retrieval algorithm returned an answer shorter than this limit, Pauchok expanded the passage by adding words surrounding the passage in the document. Similarly, if the algorithm returned a longer passage, Pauchok trimmed words from either end of the passage until it fit within the limit. This way the evaluation does not give an algorithm that happens to return longer passages an unfair advantage.

I present results using both the Lucene IR engine and the PRISE document retriever. Documents were retrieved by the Lucene system using a query generated by stripping a predefined list of stopwords from the question.

Many of the algorithms had a number of constant factors as part of their scoring function. A script tuned the algorithms for the TREC 2000 data set by varying each parameter around an initial value obtained from the algorithm's description. The algorithms ran on the TREC 2001 data set using the values that performed best on the TREC 2000 data. All results presented are for the TREC 2001 questions.

4.2 Scoring

As in TREC 2001 [31], both strict and lenient scores are presented. For lenient scoring, an answer is judged correct if it matches one of the regular expressions in the answer patterns obtained from NIST. For strict scoring, the answer also had to be supported; the document that contained the answer had to appear in the list provided by NIST containing correct documents for that question. I present both the percentage of missed questions for each algorithm and the mean reciprocal rank (MRR). MRR is

computed by averaging the inverse of rank of the first passage to correctly answer a question over all questions. I measured MRR over all twenty response passages, as opposed to the usual five in formal TREC evaluations.

Chapter 5

The Passage Retrieval Algorithms

For this study, eight different passage retrieval algorithms were implemented in the Pauchok framework. Most of the algorithms were chosen from top-performing TREC 2001 systems that had well-described passage retrieval algorithms.

I did not include passage retrieval algorithms from two notable TREC 2001 systems. The top performing TREC 2001 system, InsightSoft [28], cuts the retrieved documents into passages around query terms, returning all passages from all retrieved documents. The use of human-generated indicative patterns makes answer extraction on such large amounts of text viable. The second best system, LCC [8], retrieves passages that contain keywords from the question based on the results of question analysis. Their passage retrieval algorithm requires the expected answer type of the question or a bridging inference between the expected answer type and the question. However, neither the answer type ontology nor the bridging inference mechanisms were described well enough to be implemented.

The following sections provide an overview of the surveyed algorithms.

5.1 MITRE

MITRE's word overlap algorithm [19] assigns the passage a point for every word it has in common with the question. The version that Light describes in his work makes use of stemming when scoring the sentence. I implemented both a stemming and non-

stemming version in order to explore the effects of stemming on the baseline system performance. This algorithm represents one of the simplest reasonable techniques to use for passage retrieval.

5.2 bm25

The well known Okapi *bm25* weighting scheme [26, 25, 13] represents the state of the art in document retrieval. It takes a set of query terms and sums the score of each term with respect to the document. I implemented a passage retrieval algorithm based on this formula to serve as another baseline for comparison.

The *bm25* formula follows:

$$\sum_{T \in Q} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf} + k_2|Q|\frac{avdl-dl}{avdl+dl}$$

Q is a query, containing terms T.

$w^{(1)}$ is $\log \frac{N-n+0.5}{n+0.5}$; this formula is the Robertson-Sparck Jones weight without relevance information.

n is the number of documents containing the term.

N is the number of documents in the corpus.

tf is the frequency of the term within the document.

qtf is the frequency of the term in the corpus.

k_1, k_3, and k_3 are constants.

K is a constant equal to $k1((1-b)+b.dl/avdl)$

$avdl$ is the average document length in the corpus.

dl is the length of the document being scored.

The weight for the first term in the summation ($w^{(1)}$) is an estimate of this formula: $\log \frac{p_i(1-\overline{p_i})}{\overline{p_i}(1-p_i)}$. Here p_i is the probability that T appears in the document given that the document is relevant, and $\overline{p_i}$ is the probability that T appears in the document given that it is not relevant. Intuitively, terms that have a high probability of appearing

in relevant documents and a low probability of appearing in non-relevant documents have higher weights.

The second term in the summation, $\frac{(k_1+1)tf}{K+tf}$, is zero when the term does not occur in the passage, increases monotonically with the number of occurrences of the term, and has an asymptotic limit.

The third term in the summation, $\frac{(k_3+1)qtf}{k_3+qtf}$, is smaller the more frequently the term appears in the corpus.

The final term in the formula is a constant factor that normalizes for document length. Intuitively, a long document should not gain a higher score for containing more instances of the term than a shorter document. Since all passages in this experiment were the same length, this term could be dropped. However Pauchok's architecture supports scoring passages of different lengths with *bm25*, so this term was still used for these results.

5.3 MultiText

Clarke's MultiText algorithm [6, 7] gives weight to short passages containing many terms with high inverse document frequency (*idf*) values. The algorithm effectively slides a window across the document. Each window starts and ends with a term in the scoring query, and the score of each window is based on the number of query terms that appear in it as well as the length of the window. Short windows with many query terms are favored. Once the highest scoring passage has been identified, a 250 byte window of text around the center-point is returned as the answer. This window may be longer or shorter than the window actually scored; it depends on the density of query terms in the document.

The MultiText algorithm's scoring metric uses an *idf*-like measure rather than the actual *idf* because of the structure of the index. Pauchok's implementation uses the actual *idf*.

5.4 IBM

IBM's passage retrieval component [12, 11] computes a series of distance measures for the passage. The "matching words" measure sums the *idf* of words that appear in both the query and the passage. The "thesaurus match" measure sums the *idf* of words in the query whose WordNet synonyms appear in the passage. The "mismatch words" measure sums the *idf* of words that appear in the query and not in the passage, and reduces the score. The "dispersion" measure is the number of words in the passage between matching query terms, and the "cluster words" measure is the number of words that occurred adjacently in both the question and the passage. For each passage, it computes these distances, assigns weights to them, and sums them, yielding the score for that passage. The highest scoring passages are considered in the answer extraction module.

5.5 SiteQ

The SiteQ scorer [18] computes the score of a passage by summing the weights of the sentences. Sentences with more query terms have a higher score. Sentences with query terms close together in the text are scored higher as well. SiteQ weights query terms based on their part of speech. However, the version implemented in Pauchok uses the *idf* weight of the word for this term.

5.6 Alicante

The Alicante scorer [30, 22] computes the non-length normalized cosine similarity between the query and the passage. It takes into account the number of appearances of a term in the passage and in the query, along with *idf* values. The Alicante system reported an optimal passage size of twenty sentences, but my experiments show highest performance using a six sentence window.

5.7 ISI

The ISI passage retrieval algorithm [10] ranks sentences based on their similarity to the question. This system gives weight to exact match of proper names, term match, and stemmed word match. It also uses the CONTEX parser to check for QSUBUMED words in order to discount them, so they do not contribute to the score in both passage retrieval and answer extraction. A QSUBUMED word is one that the CONTEX parser identified as matching the answer type of the question. Since Pauchok only tested passage retrieval, its implementation ignored this factor.

5.8 Voting

I designed a new passage retrieval algorithm by combining the results from the implemented collection of algorithms. I implemented a simple voting scheme that scored each passage based on its initial rank and also based on the number of answers the other algorithms returned from the same document. More precisely, given the results from various passage retrieval algorithms, the algorithm calculates the score for each passage as follows:

$$
\begin{aligned}
A &= \text{number of algorithms} \\
R &= \text{number of passages returned} \\
docids &= A \times R \text{ matrix of document ids} \\
&\quad \text{returned by each algorithm} \\
docscore(doc) &= \sum_{a=1}^{A} \sum_{r=1}^{R} \begin{cases} 1/r & \text{if } docids[a,r] = doc \\ 0 & \text{otherwise} \end{cases} \\
score(a,r) &= \frac{1}{r} + \frac{1}{2} docscore(docids[a,r])
\end{aligned}
$$

I ran this voting algorithm using IBM, ISI, and SiteQ, the best performing algorithms under the PRISE IR engine.

Chapter 6

Results of Passage Retrieval Evaluation

First, this section presents the results of the algorithms compared to the end-to-end performance in TREC 2001. Then it compares the performance of all the algorithms using the Lucene and PRISE document retrievers. It presents similar results for the oracle document retriever. Finally it explores differences among top performing algorithms.

6.1 Comparison With End-to-end Systems

Table 6.1 presents the percentage of unanswered questions in this evaluation compared to the end-to-end results for each system in TREC 2001 in order to show that Pauchok's implementation of the passage retrieval algorithms was reasonable. Both

	Strict			**Lenient**		
Algorithm	Lucene	PRISE	TREC 2001	Lucene	PRISE	TREC 2001
IBM	49.2%	39.6%	44.3%	39.6%	30.8%	43.1%
ISI	48.8%	41.8%	41.7%	40.2%	32.2%	39.8%
SiteQ	48.0%	40.4%	56.1%	40.2%	32.6%	52.8%
MultiText	46.4%	41.6%	43.1%	38.6%	34.8%	40.7%
Alicante	50.0%	42.6%	60.4%	41.8%	35.2%	59.6%

Table 6.1: Performance showing percentage of unanswered questions over all documents using two different IR engines, compared to the end-to-end TREC 2001 results for each algorithm. Algorithms not taken from TREC 2001 systems are not shown.

	Strict				Lenient			
	Lucene		PRISE		Lucene		PRISE	
Algorithm	MRR	% Incorrect	MRR	% Incorrect	MRR	% Incorrect	MRR	% Incorrect
IBM	0.326	49.20%	0.331	39.60%	0.426	39.60%	0.421	30.80%
ISI	0.329	48.80%	0.287	41.80%	0.413	40.20%	0.396	32.20%
SiteQ	0.323	48.00%	0.358	40.40%	0.421	40.20%	0.435	32.60%
MultiText	0.354	46.40%	0.325	41.60%	0.428	38.60%	0.398	34.80%
Alicante	0.296	50.00%	0.321	42.60%	0.380	41.80%	0.391	35.20%
BM25	0.312	48.80%	0.252	46.00%	0.410	40.80%	0.345	38.00%
stemmed MITRE	0.250	52.60%	0.242	58.60%	0.338	44.20%	0.312	39.20%
MITRE	0.271	49.40%	0.189	52.00%	0.372	42.20%	0.265	42.00%
Averages	0.309	49.15%	0.297	45.33%	0.399	40.95%	0.370	35.60%
Voting with IBM, ISI, and SiteQ	0.350	39.80%	0.352	39.00%	0.410	31.00%	0.430	30.00%

Table 6.2: Performance on TREC 2001 using both Lucene and PRISE, ordered by the lenient percentage incorrect under PRISE. Mean reciprocal rank (MRR) and percentage incorrect (% Inc.) are shown for both the strict and lenient scoring conditions. The performance of the associated TREC 2001 systems is also provided.

were run on the same set of questions. NIST assessors manually scored the TREC 2001 systems; NIST created the answer patterns used to score Pauchok based on the assessors' scoring. The Pauchok runs returned twenty 1000 byte passages instead of five 50 byte answers as in TREC 2001. As a result there are generally fewer unanswered questions in the Pauchok runs. I do not compare the MRR numbers because answer extraction will likely reorder the results in the end-to-end system.

6.2 Overall Performance

Table 6.2 shows the overall performance of the passage retrieval algorithms with the Lucene and PRISE document retrievers, under strict and lenient conditions. ANOVA (over all runs except for voting) revealed that the performance difference for the PRISE set of results was statistically significant under both strict and lenient scoring (lenient: $F(7, 3992) = 3.25$, $p = 0.001$; strict: $F(7, 3992) = 3.71$, $p = 0.0005$). However, the differences in the performance of passage retrieval algorithms with Lucene was not significant under both strict and lenient scoring, as demonstrated by ANOVA over all runs except for voting (lenient: $F(7, 3696) = 0.71$, *ns*; strict $F(7, 3696) = 0.68$, *ns*). (Questions for which Lucene returned no documents are excluded from the ANOVA, although they are not excluded from Table

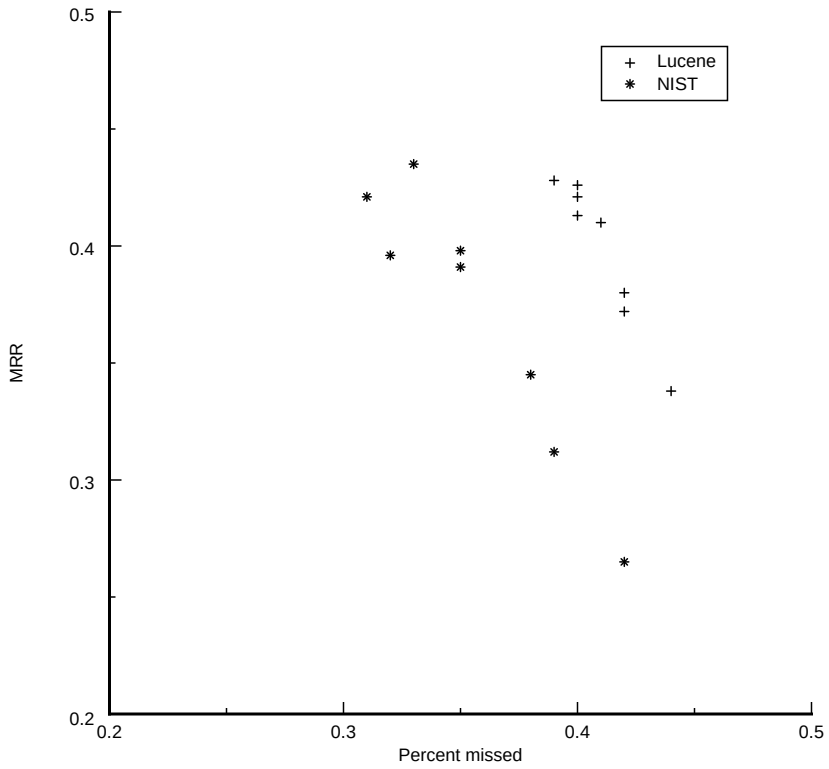


Figure 6-1: Missed questions versus MRR for algorithms using both document retrievers.

6.2.) Overall, most of the passage retrieval algorithms achieved a higher MRR with Lucene as the document retriever, but passage retrievers using the PRISE engine had fewer questions with no answers. Intuitively, passage retrievers using the PRISE IR engine answer more questions, but passage retrievers using Lucene rank correct answers higher.

Figure 6-1 visualizes the performance differences among the algorithms using Lucene and PRISE. Under PRISE the algorithms missed fewer questions. The PRISE data points are more spread out, because the performance differences between the algorithms using PRISE was statistically significant. Under both document retrievers, algorithms with higher MRR scores also missed fewer questions.

Figure 6-2 visualizes the precision/recall tradeoff between Lucene and PRISE. IBM's passage retriever is shown because it performed well, and MITRE's passage

Algorithm	# Incorrect	% Incorrect	MRR
IBM	31	7.18%	0.851
SiteQ	32	7.41%	0.859
ISI	37	8.56%	0.852
Alicante	39	9.03%	0.816
MultiText	44	10.19%	0.845
BM25	45	10.42%	0.810
MITRE	45	10.42%	0.800
stemmed MITRE	63	14.58%	0.762

Table 6.3: Performance of different passage retrieval algorithms using the oracle document retriever.

retriever is shown as a baseline. IBM using Lucene has more correct answers at ranks 1 and 2 than PRISE, even though IBM using PRISE answers more questions overall.

6.3 Performance of Voting

Although voting resulted in a slight performance boost, the improvement was not statistically significant. For Lucene, ANOVA over all runs including voting was not significant (lenient: $F(8, 4158) = 0.79$, *ns*; strict: $F(8, 4158) = 0.69$, *ns*). For the PRISE results, although the ANOVA over all runs was significant (lenient: $F(8, 4990) = 3.27$, $p = 0.0006$; strict: $F(8, 4991) = 3.64$, $p = 0.0003$), a t-test between IBM and the voting algorithm was not. (lenient: $t(499) = -0.67$, $p = 0.51$; strict: $t(499) = -0.54$, $p = 0.59$)

Table 6.3 shows the results using the oracle document retriever: every document returned contains an instance of the answer.[1] This condition tests the performance of passage retrieval algorithms under optimal document retrieval. ANOVA revealed that the difference in performance between the algorithms is statistically significant ($F(7, 3448) = 2.71$, $p = 0.008$).

	t-test results		
Algorithm	PRISE	Lucene	Oracle
ISI	$t(499) = 0.94$, $p = 0.35$	$t(462) = 0.44$, $p = 0.66$	$t(431) = 1.23$, $p = 0.22$
SiteQ	$t(499) = 1.15$, $p = 0.25$	$t(462) = 0.45$, $p = 0.66$	$t(431) = 0.19$, $p = 0.85$

Table 6.4: t-test results for ISI and SiteQ with IBM, the algorithm that found an answer for the most questions, over different IR engines. It only shows results for lenient scoring; the t-tests for strict scoring were also not statistically significant.

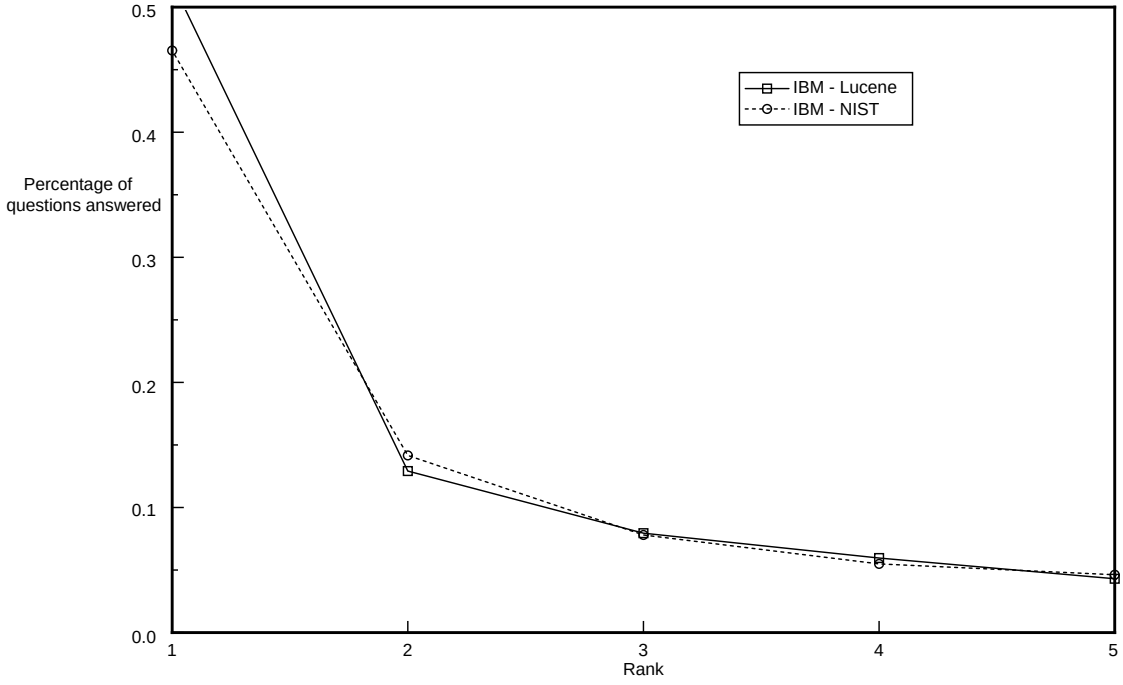
6.4 Differences Among Top Algorithms

The performance difference of the three passage retrievers that answered the most questions correctly (IBM, ISI, and SiteQ) was not significant. The results of our pairwise t-tests are shown in Table 6.4.

6.5 Scoring

Neither strict nor lenient scoring was perfect. Strict scoring displayed many false negatives, i.e., valid answers scored as incorrect, because the list of relevant documents supplied by PRISE was not exhaustive. Conversely, lenient scoring displayed many false positives, i.e., wrong answers scored as correct, because many of the answer patterns were not discriminating enough. However, both scoring conditions are useful in establishing realistic upper and lower bounds on performance. Both scoring metrics relied on the list of answer patterns provided by NIST for each TREC question. Some questions had correct answers in the corpus not matched by any of the NIST answer patterns. However the error introduced by this factor is probably not large, because the answer patterns were generated based on answers returned by TREC systems.

[1]Note that the strict and lenient measures are identical under the oracle document retriever.



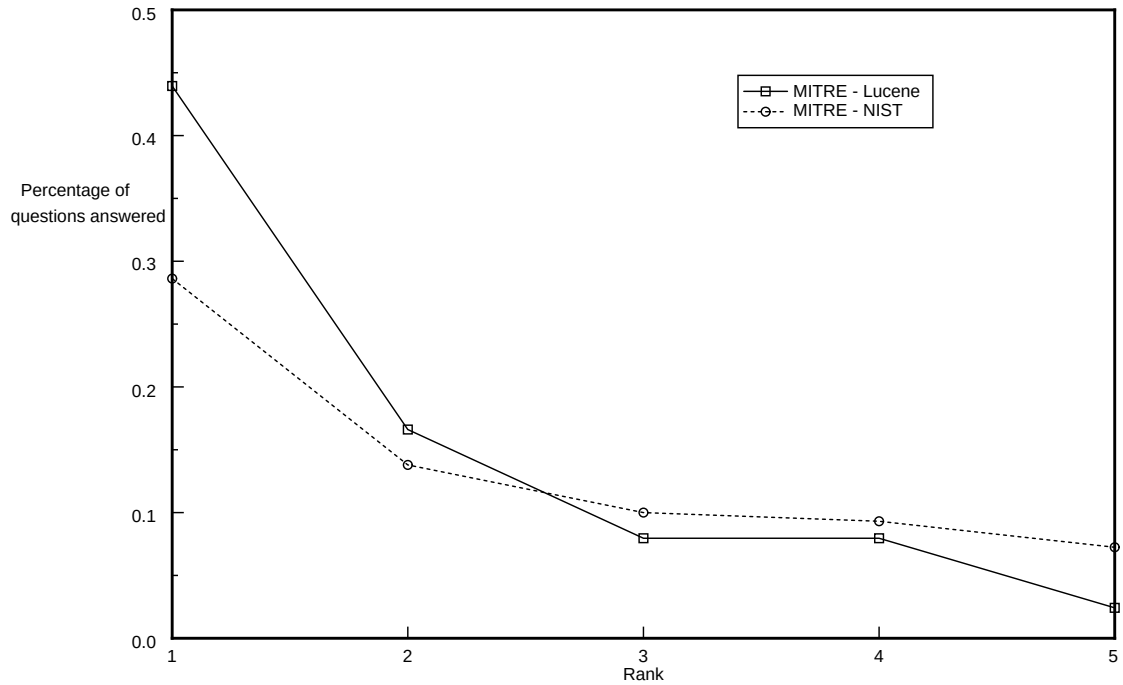


Figure 6-2: Performance at ranks 1 through 5 for two passage retrieval algorithms using both document retrievers.

Chapter 7

Discussion of Passage Retrieval Evaluation

This section discusses implications of the differences in performance of the algorithms and presents the results of an error analysis of questions that no passage retrieval algorithm was able to answer.

7.1 Density-Based Scoring

The differences in performance of the IBM, ISI, and SiteQ algorithms were not significant. All these algorithms use a density based scoring metric; they look for high scoring query terms that appear close together in the passage. The fact that all the best systems use a density based scoring metric suggests that this is a good indicator of the presence of an answer.

7.2 Boolean Querying

An immediate conclusion from this study is that in terms of passage retrieval, the performance obtained using the Lucene document retriever is comparable to the performance obtained using the PRISE document retriever. In fact, passage retrieval algorithms using Lucene actually achieve a higher MRR on average. This result is

surprising because in terms of pure document retrieval, boolean query models have been consistently outperformed by more modern approaches. Yet, for the passage retrieval task, the effectiveness of these different document retrievers is quite similar. This result confirms the intuition of many in the question answering community: boolean queries can supply a reasonable set of documents for down-stream components in a question answering system. Many of the top-performing systems in the TREC competitions (e.g., [10, 23]) employ simple boolean queries for this reason, and because a boolean model allows for finer-grained control over query expansion.

7.3 Passage Retrieval Differences

The results with Lucene show that the performance differences between passage retrieval algorithms were not statistically significant. It may be possible to attribute the differences in performance to pure chance. Based on this experience, systems utilizing boolean keyword querying schemes should focus more on improving document retrieval. Compared to the PRISE results, Lucene appears to have lower recall, consistent with conventional wisdom; as such, methods for boosting recall (e.g., query expansion techniques) should be a priority in the development of question answering systems built on boolean keyword search engines. In fact, at least one system [23] has implemented "feedback loops" which expand or contract boolean queries to improve the quality of a hit list.

In contrast, the results with the PRISE document retriever show that the performance differences attributed to the passage retrieval algorithms are statistically significant; here the passage retriever does make a difference. However, passage retrieval algorithms using PRISE tend to place relevant passages at lower ranks compared to Lucene. Generalizing from this result yields the insight that different passage retrieval techniques are still worth exploring, with a focus on better confidence ranking in the context of an IR engine like PRISE.

Finally, the results with the oracle document retriever reveal that performance differences among the various algorithms were still statistically significant even when

document retrieval was perfect. This observation suggests that document and passage retrieval technology can be developed independently and still be combined for better end-to-end performance.

7.4 Error Analysis for Missed Questions

Some questions could not be answered by any of the passage retrieval algorithms. For these questions, subsequent processing by answer extraction modules is useless. Error analysis of these questions yields suggestions for future work.

A significant number of the missed questions required definitions as an answer. NIST plans to require fewer definition questions in future tracks because users have better mechanisms for finding the definition of a term, such as WordNet [31]. However, the system could still be confronted with definition questions by users ignorant of those better mechanisms, so it is still reasonable to expect systems to answer these questions. Definition questions are difficult for keyword based passage retrieval algorithms because there is essentially one keyword available in the question. For example, for the question "What is an ulcer?" the only term specific to the question is "ulcer." Here, synonymy expansion and parsing could help to improve performance. A parser could look for a set of relations or patterns that denote the presence of the definition of a term. Synonymy expansion and recognition could allow the algorithms to recognize additional keywords that might be found near the answer. This strategy might help if the definition contains synonyms of the term. Similarly, expanding the query by using WordNet hypernyms or terms in a dictionary definition might increase the precision for these questions.

Another significant fraction of missed questions asked for the first instance of something. For example, for "What was the first satellite to go into space?", many answers had the terms "first" and "satellite" co-occur, but "first" did not modify "satellite" as in "The *first* country to launch a *satellite*". Katz and Lin [16] identified this phenomenon as *ambiguous modification*, where words in a query are found in the candidate answers, but in the wrong modification relationship (in this case,

adjective-noun modification). In this specific example, "first" is the ambiguous modifier, because it could potentially modify many head nouns that co-occur with satellite, e.g., "first country". Katz and Lin's solution extracted syntactic relations from both candidate answers and queries in order to ensure that the proper relations existed in both. The integration of linguistic processing techniques into passage retrieval algorithms could increase their overall precision.

Chapter 8

Contributions

Pauchok has instantiated a generic question answering architecture. The architecture is robust enough to support many top performing passage retrieval algorithms from TREC 2001. Lin *et al.* [20] used Pauchok as the answer projection component for Aranea in TREC 2001. Current work is being done using this architecture to explore new question answering components.

Pauchok's architecture allows components from diverse systems to be implemented in a common framework so they can be directly evaluated. This thesis empirically evaluated a number of passage retrieval algorithms. This evaluation showed that top performing algorithms use density based scoring measures. Error analysis from this evaluation suggested that future work in passage retrieval should focus on definition questions and using a parser to disambiguate word relations.

8.1 Future Work

A similar comparison of various answer extraction algorithms would be interesting, but would present more challenges than passage retrieval. The best performing answer extraction algorithms work via large databases of patterns. Replicating those databases from scratch is non-trivial.

Pauchok's framework enables the exploration of new question answering components. It would be interesting to explore ways of using a natural language parser

to improve passage retrieval. Combining the existing passage retrieval algorithms through a voting mechanism might yield an algorithm that performs better overall.

Pauchok should be entered into a future TREC competition both to evaluate its architecture as an end-to-end system and to stimulate the development of better question answering components.

Bibliography

[1] Lemur. http://www-2.cs.cmu.edu/~lemur/.

[2] Lucene. http://jakarta.apache.org/lucene/docs/index.html.

[3] Prise. http://www.itl.nist.gov/iaui/894.02/works/papers/zp2/.

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

[5] Jennifer Chu-Carroll, John Prager, Christopher Welty, Krzysztof Czuba, and David Ferrucci. A multi-strategy and multi-source approach to question answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, 2002.

[6] Charles Clarke, Gordon Cormack, Derek Kisman, and Thomas Lynam. Question answering by passage selection (Multitext experiments for TREC-9). In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.

[7] Charles Clarke, Gordon Cormack, and Elizabeth Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 36:291–311, 2000.

[8] Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, Paul Morărescu, and Răzvan Bunescu. Answering complex, list, and context questions with LCC's Question-

Answering Server. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[9] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: The view from here. *Journal of Natural Language Engineering, Special Issue on Question Answering*, Fall–Winter 2001.

[10] Eduard Hovy, Ulf Hermjakob, and Chin-Yew Lin. The use of external knowledge in factoid QA. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[11] Abraham Ittycheriah, Martin Franz, and Salim Roukos. IBM's statistical question answering system—TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[12] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi. IBM's statistical question answering system. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, 2000.

[13] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: Development and status. Technical Report 446, University of Cambridge Computer Laboratory, 1973.

[14] Boris Katz. Annotating the World Wide Web using natural language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97)*, 1997.

[15] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002)*, 2002.

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

[17] Boris Katz, Jimmy Lin, and Sue Felshin. The START multimedia information system: Current technology and future directions. In *Proceedings of the International Workshop on Multimedia Information Systems (MIS 2002)*, 2002.

[18] Gary Geunbae Lee, Jungyun Seo, Seungwoo Lee, Hanmin Jung, Bong-Hyun Cho, Changki Lee, Byung-Kwan Kwak, Jeongwon Cha, Dongseok Kim, JooHui An, Harksoo Kim, and Kyungsun Kim. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[19] Marc Light, Gideon S. Mann, Ellen Riloff, and Eric Breck. Analyses for elucidating current question answering technology. *Journal of Natural Language Engineering, Special Issue on Question Answering*, Fall–Winter 2001.

[20] Jimmy Lin, Aaron Fernandes, Boris Katz, Gregory Marton, and Stefanie Tellex. Extracting answers from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, 2002.

[21] Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. What makes a good answer? The role of context in question answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT-2003)*, 2003.

[22] Fernando Llopis and Jose L.Vicedo. IR-n: A passage retrieval system at CLEF-2001. In *Proceedings of the Second Workshop of the Cross-Language Evaluation Forum (CLEF 2001)*, 2001.

[23] Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system.

In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, 2002.

[24] John Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. Use of WordNet hypernyms for answering what-is questions. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[25] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, 1995.

[26] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, 1994.

[27] Gerard Salton, James Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-1993)*, 1993.

[28] Martin M. Soubbotin and Sergei M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[29] Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2003)*, 2003.

[30] José L. Vicedo and Antonio Ferrández. University of Alicante at TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[31] Ellen M. Voorhees. Overview of the TREC 2001 question answering track. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001.

[32] Ellen M. Voorhees and Dawn M. Tice. Overview of the TREC-9 question answering track. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.