Scaling Question Answering to the Web

Cody Kwok
Oren Etzioni
Daniel S. Weld
{ctkwok, etzioni, weld}@cs.washington.edu
University of Washington, Seattle, WA, USA

ABSTRACT

The wealth of information on the web makes it an attractive resource for seeking quick answers to simple, factual questions such as "who was the first American in space?" or "what is the second tallest mountain in the world?" Yet today's most advanced web search services (e.g., Google and AskJeeves) make it surprisingly tedious to locate answers to such questions. In this paper, we extend question-answering techniques, first studied in the information retrieval literature, to the web and experimentally evaluate their performance.

First we introduce MULDER, which we believe to be the first general-purpose, fully-automated question-answering system available on the web. Second, we describe MULDER's architecture, which relies on multiple search-engine queries, natural-language parsing, and a novel voting procedure to yield reliable answers coupled with high recall. Finally, we compare MULDER's performance to that of Google and AskJeeves on questions drawn from the TREC-8 question track. We find that MULDER's recall is more than a factor of three higher than that of AskJeeves. In addition, we find that Google requires 6.6 times as much user effort to achieve the same level of recall as MULDER.

1. INTRODUCTION AND MOTIVATION

While web search engines and directories have made important strides in recent years, the problem of efficiently locating information on the web is far from solved. This paper empirically investigates the problem of scaling question-answering techniques, studied in the information retrieval and information extraction literature [1, 16], to the web. Can these techniques, previously explored in the context of much smaller and more carefully structured corpora, be extended to the web? Will a question-answering system yield benefit when compared to cutting-edge commercial search engines such as Google[1]?

Our focus is on factual questions such as "what is the second tallest mountain in the world?" or "who was the first American in space?" While this species of questions represents only a limited subset of the queries posed to web search engines, it presents the opportunity to substantially reduce the amount of user effort required to find the answer. Instead of forcing the user to sift through a list search engine "snippets" and read through multiple web pages potentially containing the answer, our goal is to develop a system that concisely answers "K-2" or "Alan Shepard" (along with the appropriate justification) in response to the above questions.

As a step towards this goal, this paper introduces the MULDER web question-answering system. We believe that MULDER is the first general-purpose, fully-automated question-answering system available on the web.[2] The commercial search engine known as AskJeeves[3] responds to natural-language questions but its recall is very limited (see Section **??**) suggesting that it is not fully automated.

To achieve its broad scope, MULDER automatically submits multiple queries on the user's behalf to a search engine (Google) and extracts its answers from the engine's output. Thus, MULDER is an *information carnivore* in the sense described in [13]. MULDER utilizes several natural-language parsers and heuristics in order to return high-quality answers.

The remainder of this paper is organized as follows. Section **??** provides background information on question-answering systems and consider the requirements for scaling this class of systems to the web. Section **??** describes MULDER and its architecture. Section **??** compares MULDER's performance experimentally with that of AskJeeves and Google, and also analyzes the contribution of each of MULDER's key components to its performance. We conclude with a discussion of related and future work.

2. BACKGROUND

A *question answering* (QA) system provides direct answers to user questions by consulting its knowledge base. Since the early days of artificial intelligence in the 60's,

[1] `http://www.google.com`
[2] See `http://mulder.cx` for a prototype
[3] `http://www.ask.com`

researchers have been fascinated with answering natural language questions. However, the difficulty of natural language processing (NLP) has limited the scope of QA to domain-specific expert systems.

In recent years, the combination of web growth, improvements in information technology, and the explosive demand for better information access has reignited the interest in QA systems. The availability of huge document collections (*e.g.*, the web itself), combined with improvements in information retrieval (IR) and NLP techniques, has attracted the development of a special class of QA systems that answers natural language questions by consulting a repository of documents [16]. The Holy Grail for these systems is, of course, answering questions over the web. A QA system utilizing this resource has the potential to answer questions of a wide variety of topics, and will constantly be kept up-to-date with the web itself. Therefore, it makes sense to build QA systems that can scale up to the web.

In the following sections, we look at the anatomy of QA systems, and what challenges the web poses for them. We then define what is required of a web-based QA system.

2.1 Components of a QA system

An automated QA system based on a document collection typically has three main components. The first is a *retrieval engine* that sits on top of the document collection and handles retrieval requests. In the context of web, this is a search engine that indexes web pages. The second is a *query formulation* mechanism that translates natural-language questions into *queries* for the IR engine in order to retrieve relevant documents from the collection, *i.e.*, documents that can potentially answer the question. The third component, *answer extraction*, analyses these documents and extracts answers from them.

2.2 Challenges of the Web

Although the web is full of information, finding the facts sometimes resembles picking needles from a haystack. The following are specific challenges of the web to an automated QA system:

- **Forming the right queries.** Given a question, translating it into a search engine query is not a trivial task. If the query is too general, too many documents may be retrieved and the system would not have enough resources to scan through all of them. If the query is too specific, no pages are retrieved. Getting the right set of pages is crucial in dealing with the web.

- **Noise.** Sometimes even with the right set of keywords, the system may still retrieve a lot of pages that talk about something else. For example, we issued the query `first american in space` to Google, and received results about the first American woman in space and the first American to vote from space. The QA system has to be tolerant of such noise or it will become confused and give incorrect answers.

- **Factoids.** By far the worst scenario for a QA system is finding falsehoods. Advanced NLP techniques in the future may allow us to find answers 90% of the time, but if the knowledge is incorrect then the answer is still invalid. Sometimes such factoids are human errors or lies, but not always. For example, one of the sites returned by Google stated "John Glenn was the first American in space", which is false. However, the site is labeled "Common misconceptions in Astronomy". Such sentences can cause a lot of headaches for QA systems.

- **Resource limitations.** Search engines have been improving steadily and are faster than ever, but it is still very taxing to issue a few dozen queries in order to answer each question. A QA system has to make intelligent decisions on which queries to try instead of blindly issuing combinations of words. Online interactive QA systems also have to consider the time spent by the user in waiting for an answer. Very few users are willing to wait more than a few minutes.

In the next section, we describe a fully implemented prototype web QA system, MULDER, which takes these factors into consideration.

3. THE MULDER WEB QA SYSTEM

Figure 1 shows the architecture of MULDER. The user begins by asking the question in natural language from MULDER's web interface. MULDER then parses the question using a *natural language parser* (section 3.1), which constructs a tree of the question's phrasal structure. The parse tree is given to the *classifier* (section 3.2) which determines the type of answer to expect. Next, the *query formulator* (section 3.3) uses the parse tree to translate the question into a series of search engine queries. These queries are issued in parallel to the *search engine* (section 3.4), which fetches web pages for each query. The *answer extraction* module (section 3.5) extracts relevant snippets called *summaries* from the web pages, and generates a list of possible *candidate answers* from these snippets. These candidates are given to the *answer selector* (section 3.6) for scoring and ranking; the sorted list of answers (presented in the context of their respective summaries) is displayed to the user.

The following sections describe each component in detail.

3.1 Question Parsing

In contrast to traditional search engines which treat queries as a set of keywords, MULDER parses the user's question to determine its syntactic structure. As we shall see, knowing the structure of the question allows MULDER to formulate the question into several different keyword queries which are submitted to the search engine in parallel, thus increasing recall. Furthermore, MULDER uses the question's syntactic structure to better extract plausible answers from the pages returned by the search engine.

Natural language parsing is a mature field with a decades-long history, so we chose to adopt the best existing parsing technology rather than roll our own. Today's best parsers employ statistical techniques [12, 8]. These parsers are trained on a tagged corpus, and they

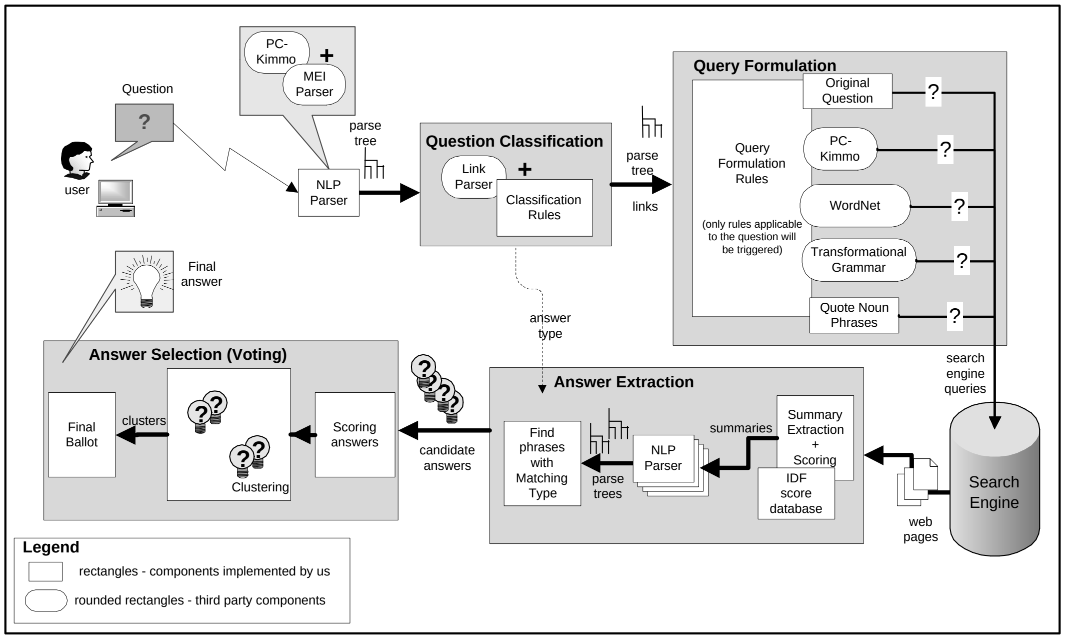


Figure 1: Architecture of Mulder

use learned probabilities of word relationships to guide the search for the best parse. The Maximum Entropy-Inspired (MEI) parser [9] is one of the best of this breed, and so we integrated it into MULDER. The parser was trained on the Penn Wall Street Journal tree-bank [21] sections $2-21$.

Our overall experience with the parser has been positive, but initially we discovered two problems. First, due to its limited training set the parser has a restricted vocabulary. Second, when the parser does not know a word, it attempts to make educated guesses for the word's part-of-speech. This increases its search space, and slows down parsing.[4] In addition, the guesses can be quite bad. In one instance, the parser tagged the word "tungsten" as a cardinal number. To remedy both of these problems, we integrated a lexical analyzer called PC-KIMMO [3] into the MEI parser. Lexical analysis allows MULDER to tag previously unseen words. For example, MULDER does not know the word "neurological", but by breaking down the word into the root "neurology" and the suffix "-al", it knows the word is an adjective. When the MEI parser does not know a word, MULDER uses PC-KIMMO to recognize and tag the part-of-speech of the word. If PC-KIMMO also does not know the word either, we adopt the heuristic of tagging the word as a proper noun.

[4]While speed is not a concern for short questions, longer questions can take a few seconds to parse. We also use the parser extensively in the answer extraction phase to parse longer sentences, where speed is a very important issue.

MULDER's combination of MEI/PC-KIMMO and noun defaulting is able to parse all TREC-8 questions correctly.

3.2 Question Classifier

Question classification allows MULDER to narrow the number of candidate answers during the extraction phase. Our classifier recognizes three types of questions: nominal, numerical and temporal, which correspond to noun phrase, number, and date answers. This simple hierarchy makes classification an easy task and offers the potential of very high accuracy. It also avoids the difficult process of designing a complete question ontology with manually-tuned answer recognizers.

Most natural questions contain a *wh-phrase*, which consists of the interrogative word and the subsequent words associated with it. The MEI parser distinguishes four main types of wh-phrases, and MULDER makes use of this type information to classify the question. Wh-adjective phrases, such as "how many" and "how tall", contain an adjective in addition to the question words. These questions inquire about the degree of something or some event, and MULDER classifies them as numerical. Wh-adverb phrases begin with single question words such as "where" or "when". We use the interrogative word to find the question type, *e.g.*, "when" is temporal and "where" is nominal. Wh-noun phrases usually begin with "what", and the questions can look for any type of answer. For example, "what height" looks for a number, "what year" a date, and "what car" a noun. We use the object of the verb in the question to determine the type. To find the object of the verb, MULDER uses the Link parser [14]. The Link parser is different from the MEI parser in that it outputs the sentence parse in the form of the relationships between words (called *links*) instead of a tree structure. MULDER uses the verb-object links to find the object in the question. MULDER then consults WordNet [22] to determine the type of the object. WordNet is a semantic network containing words grouped into sets called *synsets.* Synsets are linked to each other by different relations, such as synonyms, hypernyms and meronyms[5] To classify the object word, we traverse the hypernyms of the object until we find the synsets of "measure" or "time". In the former case, we classify the question as numerical, and in the latter case temporal. Sometimes a word can have multiple senses, and each sense may lead to a different classification, *e.g.*, "capital" can be a city or a sum of money. In such cases we assume the question is nominal. Finally, if the question does not contain a wh-phrase (*e.g.*, "Name a film that has won the Golden Bear award ..."), we classify it as nominal.

3.3 Query Formulation

The query formulation module converts the question into a set of keyword queries that will be sent to the search engine for parallel evaluation. MULDER has a good chance of success if it can *guess* how the answer may appear in a *target sentence*, *i.e.* a sentence that contains the answer. We believe this strategy works especially well for questions related to common facts, because these facts are replicated across the web on many different sites, which makes it highly probable that some web page will contain our formulated phrase. To illustrate, one can find the answer to our American-in-space question by issuing the phrase query "`"The first American in space was"`".[6] When given this query, Google returns 34 pages, and most of them say "`The first American in space was Alan Shepard`," with variations of Shepard's name.

A second insight behind MULDER's query formulation method is the idea of varying the specificity of the query. The most general query contains only the most important keywords from the user's question, while the most specific queries are quoted partial sentences. Longer phrases place additional constraints on the the query, increasing precision, but reducing the number of pages returned. If the constraint is too great, then no pages will be returned at all. *A priori* it is difficult to know the optimal point on the generality/specificity tradeoff, hence our decision to try multiple points in parallel; if a long phrase query fails to return, MULDER can fall back on the more relaxed queries. MULDER currently implements the following reformulation strategies, in order of general to specific:

- **Verb Conversion.** For questions with an auxilliary do-verb and a main verb, the target sentence is likely to contain the verb in the conjugated form rather than separate verbs. For example, in "When did Nixon visit China?", we expect the target sentence to be "Nixon visited China ..." rather than "Nixon did visit China ...". To form the verb "visited", we use PC-KIMMO's word synthesis feature. It generates words by combining the infinitive of a word with different suffixes or prefixes.[7] MULDER forms a query with the auxilliary and the main verb replaced by the conjugated verb.

- **Query Expansion.** Extending the query vocabulary via a form of *query expansion* [5, 29] improves MULDER's chances of finding pages with an answer. For example, with wh-adjective questions such as "How tall is Mt. Everest?", the answers may occur in sentences such as "the height of Everest is ...". For these questions, we use WordNet [22] to find the *attribute noun* of the adjectives. MULDER composes a new query by replacing the adjective with its attribute noun. Tight bounds are employed to ensure that MULDER will not flood the search engine with too many queries.

- **Noun Phrase Formation.** Many noun phrases, such as proper names, are atomic and should not be broken down. MULDER gets more relevant web pages if it quotes these entities in its query. For example, if one wants to find web pages on question-answering research, the query "`"question answering"`" performs better than the unquoted query. MULDER implements this idea by composing a query with quoted non-recursive noun phrases, *i.e.*, noun phrases that do not contain other noun phrases.

- **Transformation.** Transforming the question into equivalent assertions (as illustrated in the beginning of this section) is a powerful method for increasing the chance of getting target sentences. Transformational Grammar [11, 2] allows MULDER to do this in a principled manner. The grammar defines conversion rules applicable to a sentence, and their effects on the semantics of the sentence. The following are some sample transformations we have implemented in MULDER:

    1. **Subject-Aux Movements.** For questions with a single auxilliary and a trailing phrase such as the American-in-space question, MULDER can remove the interrogative and form two queries with the phrase placed in front and after the auxilliary, respectively, *i.e.*"`"was the first American in space"`" and "`"the first American in space was"`".
    2. **Subject-Verb Movements.** If there is a single verb in the question, MULDER forms a query by stripping the interrogative, *e.g.*, from "who shot JFK?" MULDER creates "`"shot JFK"`".

[5]A hypernym is a word whose meaning denotes a superordinate,*e.g.*, animal is a hypernym of dog. A meronym is a concept that is a part of another concept, *e.g.*, knob is a meronym of door.

[6]The actual query is "`"+the first American +in space +was"`", following Google's stop word syntax.

[7]Irregular verbs are handled specially by PC-KIMMO, but are only slightly trickier than the regulars.

The actual rules are slightly more complicated in order to handle complex sentences, but the essence is the same.

Since only a subset of these techniques are applicable to each question, for each user question MULDER's query formulation module issues approximately 4 search engine queries on average. We think this is a reasonable load on the search engine. Section 4.4 details experiments that show that reformulation significantly improves MULDER's performance.

3.4 Search Engine

Since MULDER depends on the web as its knowledge base, the choice of search engine essentially determines our scope of knowledge, and the method of retrieving that knowledge. The idiosyncrasies of the search engine also affect more than just the syntax of the queries. For example, with Boolean search engines one could issue a query `A AND ( B OR C )`, but for search engines which do not support disjunction, one must decompose the original query into two, `A B` and `A C`, in order to get the same set of pages.

We considered a few search engine candidates, but eventually chose Google. Google has two overwhelming advantages over others: it has the widest coverage among search engines (as of October 2000, Google has indexed 1.2 billion web pages), and its page ranking function is unrivaled. Wider coverage means that MULDER has a larger knowledge base and access to more information. Moreover, with a larger collection we have a higher probability of finding target sentences. Google's ranking function helps MULDER in two ways. First, Google's ranking function is based on the PAGERANK [17] and the HITS algorithms [19], which use random walk and link analysis techniques respectively to determine sites with higher information value. This provides MULDER with a selection of higher quality pages. Second, Google's ranking function is also based on the proximity of words, *i.e.*, if the document has keywords closer together, it will be ranked higher. While this has no effect on queries with long phrases, it helps significantly in cases when we have to rely on keywords.

3.5 Answer Extraction

The answer extraction module is responsible for acquiring candidate answers from the retrieved web pages. In MULDER, answer extraction is a two-step process. First, we extract snippets of text that are likely to contain the answer from the pages. These snippets are called *summaries*; MULDER ranks them and selects the N best. Next MULDER parses the summaries using the MEI parser and obtains phrases of the expected answer type. For example, after MULDER has retrieved pages from the American-in-space queries, the summary extractor obtains the snippet "The first American in space was Alan B. Shepard. He made a 15 minute suborbital flight in the Mercury capsule Freedom 7 on May 5, 1961." When MULDER parses this snippet, it obtains the following noun phrases, which become candidate answers: "The first American", "Alan B. Shepard", "suborbital flight" and "Mercury capsule Freedom 7".

We implemented a summary mechanism instead of extracting answers directly from the web page because the latter is very expensive. The MEI parser takes between a half second for short sentences to three or more for longer sentences, making it impractical to parse every sentence on every web page retrieved. Moreover, regions that are not close to any query keywords are unlikely to contain the answer.

MULDER's summary extractor operates by locating textual regions containing keywords from the search engine queries. In order to bound subsequent parser time and improve efficiency, these regions are limited to at most 40 words and are delimited by sentence boundaries whenever possible. We tried limiting summaries to a single sentence, but this heuristic reduced recall significantly. The following snippet illustrates why multiple sentences are often useful: "Many people ask who was the first American in space. The answer is Alan Shepard whose suborbital flight made history in 1961."

After MULDER has extracted summaries from each web page, it scores and ranks them. The scoring function prefers summaries that contain more important keywords which are close to each other. To determine the importance of a keyword, we use a common IR metric known as *inverse document frequency* (IDF), which is defined by $\frac{N}{df}$, where N is the size of a document collection and df is the number of documents containing the word. The idea is that unimportant words, such as "the" and "is", are in almost all documents while important words occur sparingly. To obtain IDF scores for words, we collected about 100,000 documents from various online encyclopedias and computed the IDF score of every word in this collection. For unknown words, we assume df is 1. To measure how close keywords are to each other, MULDER calculates the square-root-mean of the distances between keywords. For example, suppose $d_1, \ldots, d_{n-1}$ are the distances between the n keywords in summary s, then the distance is

$$D(s) = \frac{\sqrt{d_1^2 + \ldots + d_{n-1}^2}}{(n-1)}$$

If s has n keywords, each with weight w_i, then its score is given by:

$$S(s) = \frac{\sum_{i=1}^{n} w_i}{D(s)}$$

MULDER selects the best N summaries and parses them using the MEI parser. To reduce the overall parse time, MULDER runs many instances of the MEI parser on different machines so that parsing can be done in parallel. We also created a quality switch for the parser, so that MULDER can trade quality for time. The parse results are then scanned for the expected answer type (noun phrases, numbers, or dates), and the candidate answers are collected.

3.6 Answer Selection

The answer selection module picks the best answer among the candidates with a *voting* procedure. MULDER first ranks the candidates according to how close they are to keywords. It then performs *clustering* to group similar answers together, and a final ballot is cast for all clusters; the one with the highest score wins. The final answer is chosen from the top candidate in this cluster. For instance, suppose we have 3 answer candidates, “Alan B. Shepard”, “Shepard”, and “John Glenn”, each with scores 2, 1 and 2 respectively. The clustering will result in two groups, one with “Alan B. Shepard” and “Shepard” and the other with “John Glenn”. The former is scored 3 and the latter 2, thus MULDER would pick the Shepard cluster as the winner and select “Alan B. Shepard” as the final answer.

MULDER scores a candidate answer by its distance from the keywords in the neighborhood, weighted by how important those keywords are. Let k_i represent a keyword, a_i an answer word and c_i a word that does not belong to either category. Furthermore, let w_i be the weight of keyword k_i. Suppose we have a string of consecutive keywords on the left side of the answer candidate beginning at a_1, separated by m unrelated words, *i.e.*, $k_1 \ldots k_n c_1 \ldots c_m a_1 \ldots a_p$, MULDER scores the answer by

$$K_L = \frac{w_1 + \ldots + w_n}{m}$$

(the L in K_L stands for “Left”.) We compute a similar score, K_R, with the same formula, but with keywords on the right side of the answer. The score for a candidate is $max(K_L, K_R)$. For certain types of queries, we modify this score with multipliers. The answers for temporal and wh-adverb (when, where) queries are likely to occur inside prepositional phrases, so we raise the score for answers that occur under a prepositional phrase. Some transformations expect the answers on a certain side of the query keywords (left or right). For example, the search engine query “`"was the first American in space"`” expects the answer on the left side of this phrase, so MULDER rejects potential answers on the right side.

After we have assigned scores to each candidate answer, we cluster them into groups. This procedure effects several corrections:

- **Reducing noise.** Random phrases that occur by chance are likely to be eliminated from further consideration by clustering.

- **Allowing alternative answers.** Many answers have alternative acceptable forms. For example, “Shepard”, “Alan B. Shepard”, “Alan Shepard” are all variations of the same name. Clustering collects all of these dissimilar entries together so that they have collective bargaining power and get selected as the final answer.

- **Separating facts from fiction.** The web contains a lot of misinformation. MULDER assumes the truth will prevail and occur more often, and clustering embeds this ideal.

Our clustering mechanism is a simplified version of suffix tree clustering [31]. MULDER simply assigns answers that share the same words into the same cluster. While a complete implementation of suffix tree clustering that handles phrases may give better performance than our simple algorithm, our experience shows word-based clustering works reasonably well.

After the clusters are created, they are ranked according to the sum of the scores of their member answers. The member answer with the highest individual score is chosen as the *representative* of each cluster.

The sorted list of ranked clusters and their representatives are displayed as the final results to the user. Figure 2 shows MULDER's response to the American-in-space question.

4. EVALUATION

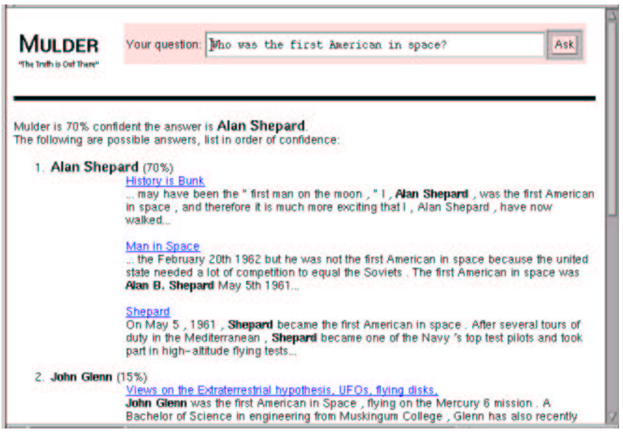
In this section, we evaluate the performance of MULDER by comparing it with two popular and highly-regarded web services, Google (the web's premier search engine) and AskJeeves (the web's premier QA system). In addition, we evaluate how different components contribute to MULDER's performance. Specifically, we study the effectiveness of query formulation, answer extraction and the voting selection mechanism.

AskJeeves allows users to ask questions in natural language. It looks up the user's question in its own database and returns a list of matching questions which it knows how to answer. The user selects the most appropriate entry in the list, and is taken to a web page where the answer will be found. Its database of questions appears to be constructed manually, therefore the corresponding web pages reliably provide the desired answer. We also compare MULDER with Google. Google supplies the web pages that MULDER uses in our current implementation and provides a useful baseline for comparison.

We begin by defining various metrics of our experiment in section 4.1, then in section 4.2 we describe our test question set. In section 4.3, we compare the performance of MULDER with the two web services on the TREC-8 question set. Finally, in section 4.4 we investigate the efficacy of various system components.

4.1 Experimental Methodology

The main goal of our experiments is to quantitatively assess MULDER's performance relative to the baseline provided by Google and AskJeeves. Traditional information retrieval systems use *recall* and *precision* to measure system performance. Suppose we have a document



Who was the first American in space?

Mulder is 70% confident the answer is **Alan Shepard**.
The following are possible answers, list in order of confidence:

1. **Alan Shepard** (70%)
   History is Bunk
   ... may have been the " first man on the moon , " I , **Alan Shepard** , was the first American in space , and therefore it is much more exciting that I , Alan Shepard , have now walked...

   Man in Space
   ... the February 20th 1962 but he was not the first American in space because the united state needed a lot of competition to equal the Soviets . The first American in space was **Alan B. Shepard** May 5th 1961...

   Shepard
   On May 5 , 1961 , **Shepard** became the first American in space . After several tours of duty in the Mediterranean , **Shepard** became one of the Navy 's top test pilots and took part in high-altitude flying tests...

2. **John Glenn** (15%)
   Views on the Extraterrestrial hypothesis, UFOs, flying disks,
   John Glenn was the first American in Space , flying on the Mercury 6 mission . A Bachelor of Science in engineering from Muskingum College , Glenn has also recently

Figure 2: Mulder output for the question “Who was the first American in space?”

collection D, there is a subset R of this collection that is relevant to a query Q. The retrieval system fetches a subset F of D for Q. Recall is defined as $\frac{|F \cap R|}{|R|}$ and precision $\frac{|F \cap R|}{|F|}$. In other words, recall measures how thorough the system is in locating R. Precision, on the other hand, measures how many relevant retrieved documents are among the retrieved set, which can be viewed as a measure of the effort the user has to expend to find the relevant documents. For example, if precision is high then, all other things being equal, the user spends less time sifting through the irrelevant documents to find relevant ones.

In question answering, we can define recall as the percentage of questions answered correctly from the test set. Precision, on the other hand, is inappropriate to measure in this context because a question is answered either correctly or incorrectly.[8] Nonetheless, we would still like to measure how much user effort is required to find the answer, not merely what fraction of the time an answer is available.

To measure *user effort* objectively across the different systems, we need a metric that captures or approximates how much work it takes for the user to reach an answer while reading through the results presented by each system. While the true "cognitive load" for each user is difficult to capture, a natural candidate for measuring user effort is time; the longer it takes the user to reach the answer, the more effort is expended on the user's part. However, reading time depends on numerous factors, such as how fast a person reads, how well the web pages are laid out, how experienced the user is with the subject, etc. In addition, obtaining data for reading time would require an extensive user study. Thus, in our first investigation of the feasibility of web QA systems, we chose to measure user effort by the number of words the user has to read from the start of each system's result page until they locate the first correct answer.

Naturally, our measure of user effort is approximate. It does not take into account page layout, users' ability to skip text instead of reading it sequentially, differences in word length, etc. However, it has two important advantages. First, the metric can be computed automatically without requiring an expensive user study. Second, it provides a common way to measure user effort across multiple systems with very different user interfaces.

We now define our metric formally. The *word distance* metric counts how many words it takes an idealized user to reach the first correct answer starting from the top of the system's result page and reading sequentially towards the bottom. Suppose $s_1, s_2, \ldots, s_n$ are the summary entries in the results page returned by the system, and d_i is the web page linked to by s_i. We define $|t|$ to be the number of words in the text t before the answer. If t does not have the answer, then $|t|$ is the total number of words in t. Finally, let a be the index of the entry with the answer, thus s_a and d_a are the *first* snippet and document containing the answer respectively. $|\ d_a\ | = 0$ if the answer is found in s_a, since we do not need to view d_a if s_a already contains the answer.

We now define $W(C, P)$, the word distance of the first correct answer C on a results page P, as follows:

$$W(C,P) = \left(\sum_{i=1}^{a} |\ s_i\ | \right) + |\ d_a\ |$$

Note that our word distance metric favors Google and AskJeeves by assuming that the user is able to determine which document d_i contains the answer by reading snippets exclusively, allowing her to skip $d_1 \ldots d_{i-1}$. On the other hand, it does assume that people read snippets and pages sequentially without skipping or employing any kind of "speed reading" techniques. Overall, we feel that this a fair, though approximate, metric for comparing user effort across the three systems investigated.

It is interesting to note that an average person reads 280 words per minute [27], thus we can use word distance to provide a rough estimate of the time it takes to find the answer.

4.2 Data Set

For our test set, we use the TREC-8 question track which consists of 200 questions of varying type, topic, complexity and difficulty. Some example questions are shown in Table 1.

In the original question track, each system is provided with approximately 500,000 documents from the TREC-8 document collection as its knowledge base, and is required to supply a ranked list of candidate answers for each question. The score given to each participant is based on where answers occur in its lists of candidates. The scoring procedure is done by trained individuals. All the questions are guaranteed to have an answer in the collection.

We chose TREC-8 as our test set because its questions are very well-selected, and it allows our work to be put into context with other systems which participated in the track. However, our experiments differ from the TREC-8 question track in four ways. First, MULDER is a QA system based on the web, therefore we do not use the TREC-8 document collection as our knowledge base. Second, because we do not use the original document collection, it is possible that some questions will not have answers. For example, we spent about 30 minutes with Google looking for the answer to "How much did Mercury spend on advertising in 1993?" and could not find an answer. Third, since we do not have the resources to score all of our test runs manually, we constructed a list of answers based on TREC-8's relevance judgment file[9] and our manual effort. Furthermore, some of TREC-8's answers had to be slightly modified or updated. For example, "How tall is Mt. Everest?" can be answered either in meters or feet; "Donald Kennedy" was TREC-8's answer to "Who is the president of Stanford University", but it is Gerhard

[8]Consider a search engine that returns one hundred results. In one case, the correct answer is the top-ranked result and all other answers are incorrect. In the second case, the first fifty answers are incorrect and the last fifty are correct. While precision is much higher in the second case, user effort is minimized in the first case. Unlike the case of collecting relevant documents, the value of repeating the correct answer is minimal.

[9]`http://trec.nist.gov/data/qrels_eng/qrels.trec8.qa.gz`

Question	Answer
Who is the President of Ghana?	Rawlings
What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)?	Tourette' s Syndrome
How far is Yaroslavl from Moscow?	280 miles
What country is the biggest producer of tungsten?	China
When did the Jurassic Period end?	144 million years ago

Table 1: Some sample questions from Trec-8 with answers provided by Mulder.

Casper at the time of writing.[10] Finally, whereas TREC-8 participants are allowed to return a snippet of fixed length text as the answer, MULDER returns the most precise answer whenever possible.

4.3 Comparison with Other Systems

In our first experiment, we compare the user effort required to find the answers in TREC-8 by using MULDER, Google and AskJeeves. User effort is measured using word distance as defined in section 4.1. At each word distance level, we compute the recall for each system. The maximum user effort is bounded at 5000, which represents more than 15 minutes of reading by a normal person. To compare the systems quantitatively, we compute *total effort* at recall r% by summing up the user effort required from 0% recall up to r%. Figure 3 shows the result of this experiment.

The experiment shows that MULDER outperforms both AskJeeves and Google. MULDER consistently requires less user effort to reach an answer than either system at every level of recall. AskJeeves has the lowest level of recall; its maximum is roughly 20%. Google requires 6.6 times more total effort than MULDER at 64.5% recall (Google's maximum recall).

[10]The updated set of questions and answers used in our experiments is publicly available at `http://longinus.cs.washington.edu/mulder/trec8t.qa`.

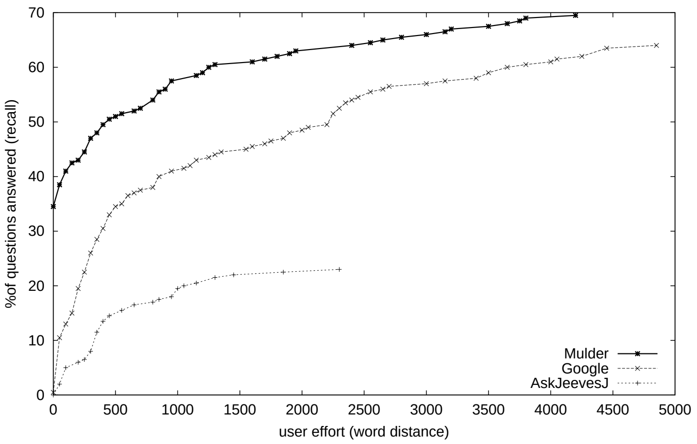


Figure 3: Recall vs. user effort: Mulder versus Google versus AskJeeves. On average, Google requires 6.6 times the total effort of Mulder, to achieve the same level of recall. AskJeeves' performance is substantially worse.

MULDER displayed an impressive ability to answer questions effectively. As figure 3 shows, 34% of the questions have the correct answer as their top-ranked result (as compared with only 1% for Google.) MULDER's recall rises to about 60% at 1000 words. After this point, MULDER's recall per unit effort grows linearly at a rate similar to Google. The questions in this region are harder to answer, and some of the formulated queries from transformations do not retrieve any documents, hence MULDER's performance is similar to that of Google. MULDER still has an edge over Google, however, due to phrased queries and answer extraction. The latter filters out irrelevant text so that users can find answers with MULDER while reading far less text than they would using Google. Finally, MULDER's curve extends beyond this graph to about 10,000 words, ending at 75%. Based on our limited testing, we speculate that of the 25% unanswered questions, about 10% cannot be answered with pages indexed by Google at the time of writing, and the remaining 15% require more sophisticated search strategies.

From our results, we can also see the limited recall of AskJeeves. This suggests that a search engine with a good ranking function and wide coverage is better than a non-automated QA system in answering questions.

4.4 Contributions by Component

Our second experiment investigates the contribution of each component of MULDER to its performance. We compare MULDER to three different variants of itself, each with a single missing component:

- **Mulder without query formulation (M-QF).** We take away the query formulation module and issue only the question to the search engine.

- **Mulder without answer extraction (M-X)** Without answer extraction, we locate answers in this system using the same approach we used for Google in the previous experiment. To linearize the results retrieved from multiple queries, we order queries in order of generality, starting from the most specific query. Then results from each query are selected in a breadth-first manner. To illustrate, suppose $r_{i,j}$ is the j^{th} result from the i^{th} query, and we have n queries. The order would be $r_{1,1}, r_{2,1}, \ldots, r_{n,1}, r_{1,2}, \ldots, r_{n,2}, \ldots$ and so on. $r_{n,i}$ would be the i^{th} result from the original unformulated query, since it is the most general.

- **Mulder without voting (M-V)** In this system,

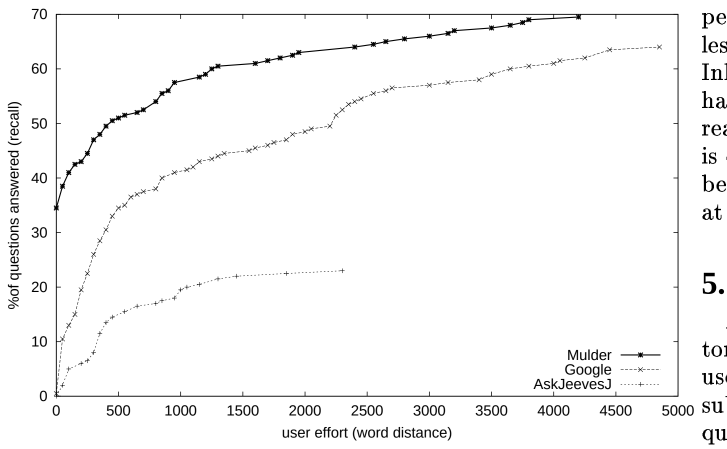


Figure 4: Recall vs. user effort: Mulder versus 3 different configurations of itself - without query formulation (M-QF), without answer extraction (M-X), and without voting (M-V). Google is the baseline.

System	$\frac{\text{total effort}}{\text{total effort MULDER}}$
MULDER	1.0
M-V	2.3
M-QF	3.0
M-X	3.8
Google	6.6

Table 2: Performances of Mulder variants and Google as multiples of total effort required by Mulder at 64.5% recall. All three Mulder variants perform worse than Mulder but better than Google. The experiment suggests that answer extraction (M-X) is the most important, followed by query formulation (M-QF) and finally voting (M-V).

we replace voting by a simple scheme: it assigns candidate answers the same score given to the summaries from which they were extracted from.

In addition, we included Google as a baseline for comparison. As in the first experiment, we compare the systems' performance on recall and user effort. Figure 4 illustrates the results.

Figure 4 shows that at all recall levels, systems with missing components perform worse than MULDER, but better than Google. Among them, M-V appears to perform better than M-QF, and M-X trails both. Table 2 shows total user effort for each system as a multiple of MULDER's at 64.5% recall, Google's maximum. The numbers suggest that answer extraction is the most important, followed by query formulation, and voting least. Answer extraction plays a major role in distilling text which saves users a lot of time. Query formulation helps in two ways: first, by forming queries with alternative words that improve the chance of finding answers; second, by getting to the relevant pages sooner. The positive effects of query formulation can also be seen with the improvement in M-X over Google; we expect this effect to be much more pronounced with other less impressive search engines, such as AltaVista[11] and Inktomi.[12] Finally, voting is the last element that enhances answer extraction. Without voting, M-V only reaches 17% recall at effort 0; with voting, the number is doubled. It is however less effective after 1500 words, because it becomes difficult to find any correct answer at that point.

5. RELATED WORK

Although question answering systems have a long history in Artificial Intelligence, previous systems have used processed or highly-edited knowledge bases (e.g., subject-relation-object tuples [18], edited lists of frequently asked questions [6], sets of newspaper articles [16], and an encyclopedia [20]) as their foundation. As far as we know, MULDER is the first automated question-answering system that uses the full web as its knowledge base. AskJeeves[13] is a commercial service that provides a natural language question interface to the web, but it relies on hundreds of human editors to map between question templates and authoritative sites. The Webclopedia project[14] appears to be attacking the same problem as MULDER, but we are unable to find any publications for this system, and from the absence of a functioning web interface, it appears to be still in active development.

START [18] is one of the first QA systems with a web interface, having been available since 1993. Focussed on questions about geography and the MIT InfoLab, START uses a precompiled knowledge base in the form of subject-relation-object tuples, and retrieves these tuples at run time to answer questions. Our experience with the system suggests that its knowledge is rather limited, *e.g.*, it fails on simple queries such as "What is the third highest mountain in the world?"

The seminal MURAX [20] uses an encyclopedia as a knowledge base in order to answer trivia-type questions. MURAX combines a Boolean search engine with a shallow parser to retrieve relevant sections of the encyclopedia and to extract potential answers. The final answer is determined by investigating the phrasal relationships between words that appear in the question. According to its author MURAX's weak performance stems from the fact that the vector space model of retrieval is insufficient for QA, at least when applied to a large corpus like an encyclopedia.

Recently Usenet FAQ files (collections of frequently asked questions and answers, constructed by human experts) have become a popular choice of knowledge base, because they are small, cover a wide variety of topics, and have high content-to-noise ratio. Auto-FAQ [30] was an early system, limited to a few FAQ files. FAQFinder [6] is a more comprehensive system which uses a vector-space IR engine to retrieve a list of relevant FAQ files from the question. After the user

[11]`http://www.altavista.com`

[12]`http://www.inktomi.com`. Inktomi powers web sites such as `http://www.hotbot.com` and `http://www.snap.com`.

[13]`http://www.ask.com`

[14]`http://www.isi.edu/natural-language/projects/webclopedia/`

selects a FAQ file from the list, the system proceeds to use question keywords to find a question-answer pair in the FAQ file that best matches the user's question. Question matching is improved by using WordNet [22] hypernyms to expand the possible keywords. Another system [25] uses *priority keyword matching* to improve retrieval performance. Keywords are divided into different classes depending on their importance and are scored differently. Various morphological and lexical transformations are also applied to words to improve matching. MULDER uses a similar idea by applying IDF scores to estimate the importance of words.

There are also many new QA systems spawning from the TREC-8 QA competition [28], the 1999 AAAI Fall Symposium on Question Answering [10] and the Message Understanding Conferences [1]. Since the TREC-8 competition provides a relatively controlled corpus for information extraction, the objectives and techniques used by these systems are somewhat different from those of MULDER. Indeed, most TREC-8 systems use simple, keyword-based retrieval mechanisms, instead focussing on techniques to improve the accuracy of answer extraction. While effective in TREC-8, such mechanisms do not scale well to the web. Furthermore, these systems have not addressed the problems of noise or incorrect information which plague the web. A lot of common techniques are shared among the TREC-8 systems, most notably a *question classification* mechanism and *name-entity tagging*.

Question classification methods analyze a question in order to determine what type of information is being requested so that the system may better recognize an answer. Many systems define a hierarchy of answer types, and use the first few words of the question to determine what type of answer it expects (such as [23, 26]). However, building a complete question taxonomy requires excessive manual effort and will not cover all possible cases. For instance, *what* questions can be associated with many noun subjects, such as *what height*, *what currency* and *what members*. These subjects can also occur in many positions in the question. In contrast, MULDER uses a simple question wh-phrase matching mechanism with combined lexical and semantic processing to classify the subject of the question with high accuracy.

Name-entity (NE) tagging associates a phrase or a word with its semantics. For example, "Africa" is associated with "country", "John" with "person" and "1.2 inches" with "length". Most of the NE taggers are trained on a tagged corpus using statistical language processing techniques, and report relatively high accuracy [4]. QA systems use the NE tagger to tag phrases in the question as well as the retrieved documents. The number of candidate answers can be greatly reduced if the system only selects answers that have the required tag type. Many systems report favorable results with NE tagging, *e.g.* [26]. We believe a NE tagger would be a useful addition to MULDER, but we also believe a QA system should not rely too heavily on NE tagging, as the number of new terms changes and grows rapidly on the web.

Relationships between words can be a powerful mechanism to recover answers. As mentioned previously, the START system uses subject-relation-object relationships. CL Research's QA system [24] parses all sentences from retrieved documents and forms *semantic relation triples* which consist of a discourse entity, a semantic relation that characterizes the entity's role in the sentence, and a governing word to which the entity stands in the semantic relation. Another system, described by Harabagiu [15], achieves respectable results for TREC-8 by reasoning with linkages between words, which are obtained from its dependency-based statistical parser [12]. Harabagiu's system first parses the question and the retrieved documents into word linkages. It then transforms these linkages into logical forms. The answer is selected using an inference procedure, supplemented by an external knowledge base. We believe mechanisms of this type would be very useful in reducing the number of answer candidates in MULDER.

6. FUTURE WORK

Clearly, we need substantially more experimental evaluation to validate MULDER's strong performance in our initial experiments. We plan to compare MULDER against additional search services, and against traditional encyclopedias. In addition, we plan to evaluate MULDER using additional question sets and additional performance metrics. We plan to conduct controlled user studies to gauge the utility of MULDER to web users. Finally, as we have done with previous research prototypes, we plan to gather data on the performance of the deployed version of MULDER in practice.

Many challenges remain ahead of MULDER. Our current implementation utilizes little of the semantic and syntactic information available during answer extraction. We believe our recall will be much higher if these factors are taken into consideration.

There are many elements that can determine the truth of an answer. For example, the *last-modified* information from web pages can tell us how recent the page and its answer is. The authority of a page is also important, since authoritative sites are likely to have better contents and more accurate answers. In future versions of MULDER we may incorporate these elements into our answer selection mechanism.

We are also implementing our own search engine, and plan to integrate MULDER with its index. We expect that an evaluation of the best data structures and integrated algorithms will provide many challenges. There are many advantages to a local search engine, the most obvious being reduced network latencies for querying and retrieving web pages. We may also improve the efficiency and accuracy for QA with analyses of a large document collection. One disadvantage of a home-grown search engine is our smaller coverage, since we do not have the resources of a commercial search engine. However, with focussed web crawling techniques [7], we may be able to obtain a very high quality QA knowledge base with our limited resources.

7. CONCLUSION

In this paper we set out to investigate whether the question-answering techniques studied in classical information retrieval can be scaled to the web. While additional work is necessary to optimize the system so that it can support high workloads with fast response, our initial experiments (reported in Section ??) are encouraging. Our central contributions are the following:

- We fielded `http://mulder.cx`, the first general-purpose, fully-automated question-answering system available on the web.
- MULDER is based on a novel architecture that combines information retrieval ideas with those of statistical natural language processing, lexical analysis, query formulation, answer extraction, and voting.
- We performed an ablation experiment which showed that each major component of MULDER (query formulation, answer extraction, and voting) contributed to the system's overall effectiveness.
- We conducted empirical experiments which showed that MULDER reduces user effort by a factor of 6.6 when compared with Google. We also demonstrated a three-fold advantage in recall when comparing our fully-automated MULDER system with the manually-constructed AskJeeves.

Our work is a step towards the ultimate goal of using the web as a comprehensive, self-updating knowledge repository that can be automatically mined to answer a wide range of questions with much less effort than is required from users interacting with today's search engines.

8. ACKNOWLEDGEMENTS

We thank David Ko and Sam Revitch for their contributions to the project. We also thank Erik Sneiders, Tessa Lau and Yasuro Kawata for their helpful comments. This research is supported by NSF Grant IIS9874759.

9. BIBLIOGRAPHY

10. REFERENCES

[1] *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Morgan Kaufmann, 1998.

[2] Adrian Akmajian and Frank Heny. *An Introduction to the Principles of Transformational Syntax*. MIT Press, Cambridge, Massachusetts, 1975.

[3] Evan L. Antworth. *PC-KIMMO: a two-level processor for morphological analysis.* Dallas, TX: Summer Institute of Linguistics, 1990.

[4] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: A High-performance Learning Name Finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

[5] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *NIST Special Publication 500-225: The Third Text REtrieval Conference (TREC-3)*, pages 69–80. Department of Commerce, National Institute of Standards and Technology, 1995.

[6] Robin Burke, Kristian Hammond, Vladimir Kulyukin, Steven Lytinen, Noriko Tomuro, and Scott Schoenberg. Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. Technical Report TR-97-05, University of Chicago, Department of Computer Science, 1997.

[7] S. Chakrabarti, M. van der Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. In *Proceedings of 8th International World Wide Web Conference (WWW8)*, 1999.

[8] E. Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18(4), Winter 1997.

[9] E. Charniak. A Maximum-Entropy-Inspired Parser. Technical Report CS-99-12, Brown University, Computer Science Department, August 1999.

[10] Vinay Chaudhri and Co-chairs Richard Fikes. Question Answering Systems: Papers from the 1999 Fall Symposium. Technical Report FS-98-04, AAAI, November 1999.

[11] Noam Chomsky. *Aspects of a Theory of Syntax.* MIT Press, Cambridge, Mass., 1965.

[12] Michael John Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the ACL, Santa Cruz*, 1996.

[13] O. Etzioni. Moving up the information food chain: softbots as information carnivores. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996. Revised version reprinted in AI Magazine special issue, Summer '97.

[14] Dennis Grinberg, John Lafferty, and Daniel Sleator. A Robust Parsing Algorithm for Link Grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies, Prague*, September 1995.

[15] Sanda Harabagiu, Marius Pasca, and Steven Maiorano. Experiments with Open-Domain Textual Question Answering. In *Proceedings of COLING-2000, Saarbruken Germany*, August 2000.

[16] David Hawking, Ellen Voorhees, Peter Bailey, and Nick Craswell. Overview of TREC-8 Question Answering Track. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Eighth Text Retrieval Conference - TREC-8*, November 1999.

[17] Monika R. Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. Measuring index quality using random walks on the Web. 31(11–16):1291–1303, May 1999.

[18] B. Katz. From Sentence Processing to Information Access on the World Wide Web. In *Natural Language Processing for the World Wide Web: Papers from the 1997 AAAI Spring Symposium*, pages 77–94, 1997.

[19] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*,

1998.

[20] Julian Kupiec. MURAX: A Robust Linguistic Approach for Question Answering Using an On-Line Encyclopedia. In Robert Korfhage, Edie M. Rasmussen, and Peter Willett, editors, *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1*, pages 181–190. ACM, 1993.

[21] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, 1993.

[22] G. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1991.

[23] Dragomir R. Radev, John Prager, and Valerie Samn. The Use of Predictive Annotation for Question Answering in TREC8. In *Proceedings of the 8th Text Retrieval Conference (TREC-8). National Institute of Standards and Technology, Gaithersburg MD*, pages 399–411, 1999.

[24] K.C. Litkowski (CL Research). Question-Answering Using Semantic Relation Triples. In *Proceedings of the 8th Text Retrieval Conference (TREC-8). National Institute of Standards and Technology, Gaithersburg MD*, pages 349–356, 1999.

[25] E. Sneiders. Automated FAQ Answering: Continued Experience with Shallow Language Understanding. In *Question Answering Systems. Papers from the 1999 AAAI Fall Symposium. Technical Report FS-99-02*, 1999.

[26] Rohini Srihari and Wei Li (Cymfony Inc.). Information Extraction Supported Question Answering. In *Proceedings of the 8th Text Retrieval Conference (TREC-8). National Institute of Standards and Technology, Gaithersburg MD*, pages 185–196, 1999.

[27] S. E. Taylor, H. Franckenpohl, and J. L. Pette. Grade level norms for the component of the fundamental reading skill. *EDL Information and Research Bulletin No. 3. Huntington. N.Y.*, 1960.

[28] E. Voorhees and D. Tice. *The TREC-8 Question Answering Track Evaluation*, pages 77–82. Department of Commerce, National Institute of Standards and Technology, 1999.

[29] Ellen Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of ACM SIGIR, Dublin, Ireland*, 1994.

[30] Steven D. Whitehead. Auto-FAQ: An experiment in cyberspace leveraging. *Computer Networks and ISDN Systems*, 28(1–2):137–146, December 1995.

[31] O. Zamir and O. Etzioni. A Dynamic Clustering Interface to Web Search Results. In *Proceedings of the Eighth Int. WWW Conference*, 1999.