Probabilistic Question Answering on the Web

Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, Amardeep Grewal
The University of Michigan
Ann Arbor, MI, 48105
USA

{radev, wfan, hqi, harriswu, agrewal}@umich.edu

ABSTRACT

Web-based search engines such as Google and NorthernLight return documents that are relevant to a user query, not answers to user questions. We have developed an architecture that augments existing search engines so that they support natural language question answering. The process entails five steps: query modulation, document retrieval, passage extraction, phrase extraction, and answer ranking. In this paper we describe some probabilistic approaches to the last three of these stages. We show how our techniques apply to a number of existing search engines and we also present results contrasting three different methods for question answering. Our algorithm, probabilistic phrase reranking (PPR) using proximity and question type features achieves a total reciprocal document rank of .20 on the TREC 8 corpus. Our techniques have been implemented as a Web-accessible system, called NSIR.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous; D.2 [**Software**]: Software Engineering; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Algorithms, Design, Experimentation, Performance, Languages

Keywords

question answering, information retrieval, natural language processing, search engines, answer extraction, answer selection, query modulation.

1. INTRODUCTION TO WEB-BASED Q&A

Given the amount of information that is available on the Web, it is not surprising that it is an ideal source of answers to a large variety of questions. The problem is that existing front ends to the Web such as NorthernLight, Google, and AlltheWeb's FAST search engine are designed to retrieve *documents* that are relevant to a user *query*, posed in an idiosyncratic language, and not to a *question* formulated in a *human language* such as English.

We claim that it is much more natural for a user to type a question such as "Who wrote King Lear?" or "What is the largest city in Northern Afghanistan?" rather than queries such as "(wrote OR written OR author) AND ('King Lear')". We also believe that when a user is looking for an answer, rather than a document, then the search engine should return an actual answer, possibly in the context of the document where it occurs. The process of retrieving answers from questions is known as Natural Language Question Answering (NLQA).

In this paper, we will introduce a method for Web-based question answering called Probabilistic Phrase Reranking (PPR). It is fully implemented at the University of Michigan as a Web-accessible system, called NSIR (pronounced "answer"). The existing NLQA systems that are closest to PR are Ask Jeeves[1], Mulder[2] [1], and Ionaut[3] [2]. Ask Jeeves accepts questions but doesn't actually return answers and rather engages in a menu-driven dialogue with the user. Mulder is no longer available on the Web for a comparison. Ionaut is fast and interactive, but it is based on a local cache of files and doesn't provide access to the full Web.

We should note that all search engines allow a user to enter a natural language question instead of a query. The search engines then remove certain frequent stop words such as "is" or "where" and treat the rest of the question as a query. Thus search engines behave as if they can handle the first stage of NLQA (input in the form of a question). However, they still provide documents rather than answers as their output results. For example, when we asked the question "What is the largest city in Northern Afghanistan?" in the Google[4] search engine, we got the following results back (the full list has been truncated for reasons of space).

> Yahoo! Full Coverage - Afghanistan
> ... within three miles of the airport at Mazar-e-Sharif, the **largest city** in **northern Afghanistan**, held since 1998 by the Taliban. There was no immediate comment ...
> uk.fc.yahoo.com/photos/a/afghanistan.html - 13k - Cached - Similar pages
>
> washingtonpost.com: World
> ... died in Kano, **northern** Nigeria's **largest city**, during two days of anti-American riots led by Muslims protesting the US-led bombing of **Afghanistan**, according to ...
> www.washingtonpost.com/wp-dyn/print/world/ - Similar pages

[1]http://www.ask.com
[2]http://mulder.cx
[3]http://www.ionaut.com:8400/
[4]http://www.google.com

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

The result consists of short summaries of all relevant documents plus pointers to the documents themselves. It is clear that a document returned by the search engine as relevant to the input question is likely to contain the answer. If the very first returned document doesn't contain the answer, it is still possible for another top-ranked document to contain it. The problem becomes then how to identify the correct answer within the top n relevant documents returned by a search engine.

A similar output, produced by AlltheWeb is shown in Figure 1.

After reviewing relevant previous work, this paper will describe our new method for Web-based NLQA. The PPR technique (Probabilistic Phrase Reranking) relies on an existing search engine to return documents that are likely to contain the answer to a user question. PPR goes through several stages until it extracts and ranks the most likely answers to the question. These stages are query modulation, document retrieval, passage (or sentence) retrieval, phrase (answer) extraction and answer ranking. These are described in more detail in the following Sections.

All experiments described in this paper were performed by the authors using AlltheWeb, Google, and Northern Light in the period September – November 2001.

2. RELATED WORK

START [3] is one of the first QA systems. However, it focuses only on questions about geography and the MIT InfoLab. A pre-compiled knowledge base is used to answer questions. Another earlier system, MURAX [4], uses an encyclopedia as a knowledge base to answer trivia questions. Given a question, MURAX uses a shallow parser to extract potential answers from sections of the encyclopedia based on the phrasal relationships between words in the question.

A large number of QA systems have emerged recently. Primarily, they follow two directions: one direction is to use the TREC Q&A [5] data as the test corpus and develop their own search engines and answer extraction techniques on top of the corpus; the other direction is to use the WWW as the potential answer source and use generic search engines, such as Google, to retrieve information related to the question and do further post-processing to extract the answers for the questions. We will review some recent work in this section.

2.1 Related work from TREC

The TREC question answering evaluation [5] is the motivating force behind a recent surge in question answering research. Systems participating in TREC have to identify short passages from a 2-GB text corpus that contain answers to a list of factual questions. [6, 7] introduced the technique of predictive annotation, a methodology for indexing texts for fact-seeking question answering. The idea of this approach is that texts in documents are annotated with labels anticipating their being targets of certain kinds of questions. Given a question, their system retrieves a set of passages that may contain the answers. The potential answers are then extracted from these passages and ranked using a linear ranking function of various heuristics, which is learned by logistic regression.

[8] developed a Q&A system called Webclopedia based on IR and NLP techniques. A given question is first parsed to create a query to retrieve the top ranked documents. These top-ranked documents are then split into segments and further ranked. Potential answers are then extracted and sorted according to a ranking function involving the match with the question type and patterns. These question patterns are manually constructed. The classification of each question and potential answer to these patterns is done using rules learned from a machine-learning based grammar parser.

Abney et al. [2] described a method based on named entity identification techniques. For each question, a set of relevant passages that mostly contain the answers are first identified. A candidate set of entities are extracted from these retrieved passages. Both the question and these extracted entities are classified into a pre-defined set of categories. Only those entities that match the category required by the question are retained and re-ranked using the frequency and other position related information.

Clarke et al. [9] also applied the passage retrieval techniques for initial preprocessing. The passage ranking utilized semantic match information between the query type and term, the IDF-liked term weighting information of each term and also the coverage of these query related terms in the passage itself. A statistical context free grammar parser based on WordNet is used to determine the question category. Those top ranked passages were then scanned for patterns matching the answer category and the potential answers were extracted and ranked using various quality heuristics.

A different technique named boosting, which integrates different forms of syntactic, semantic and pragmatic knowledge, is presented in [10]. For example, question reformulation is used to construct a query that contains more semantic information based on WordNet and named entity recognition techniques are employed to ensure high quality passage retrieval. Potential answers are extracted from the semantically rich passages that match the question type. These candidate answers are further justified by using abductive reasoning and only those that pass the test are retrieved. The system, named Falcon, scored very high in the recent TREC Q&A evaluation contest [5].

2.2 Related work on the Web

The TREC conference offers an exciting environment for competitive research on Question-Answering. However, the questions that can be answered from the fixed text corpus as in TREC are limited. Various efforts are now under way and try to port existing Q&A techniques to a much larger context — the World Wide Web. Our earlier study [11] already shows that the current WWW search engines, especially those with very large index like Google, offers a very promising source for question answering.

Agichtein et al. [12] presented a technique on how to learn search engine specific query transformations for question answering. A similar transformation technique also appeared in [13]. The idea is that the current query interfaces of most generic search engines, such as Google, etc., does not provide enough capability for direct question answering in natural language mode. By transforming the initial natural questions into a certain format which include more domain specific information can dramatically improve the chances of finding good answers at the top of the search hit lists. A set of transformation rules are learned from a training corpus and applied to the questions at the search time and the experiments have shown promising results. Their work,

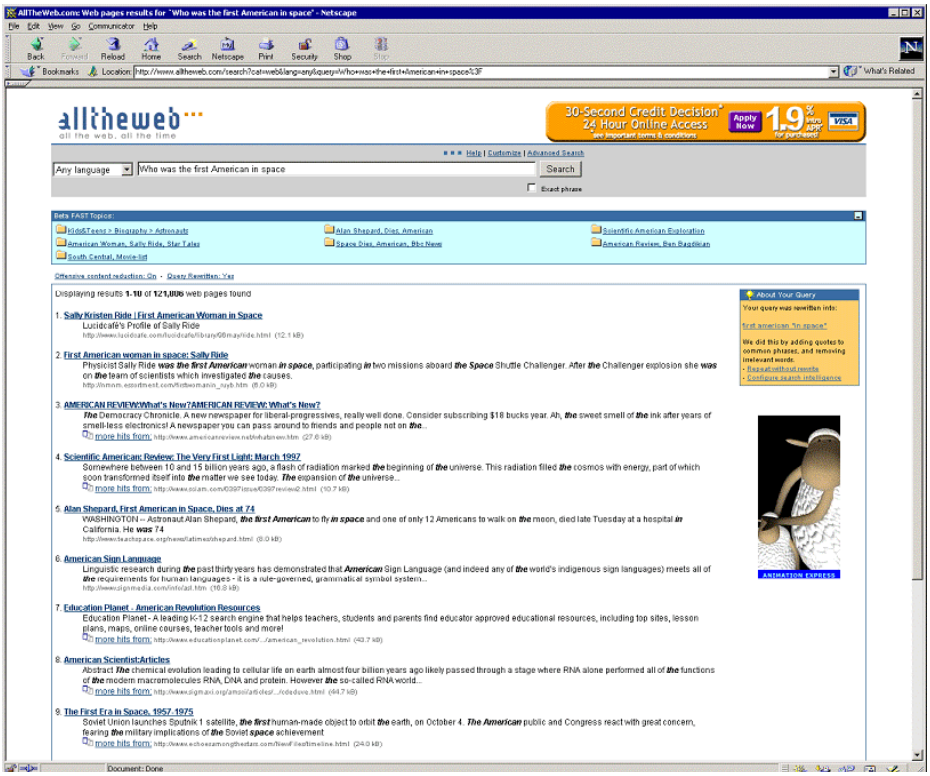

Figure 1: Sample output from AlltheWeb.

however, is focused on improving the chances of getting high quality documents from a search engine. It does not provide any mechanism to identify the true answers from the search results. A even more general framework on query transformation in discussed in [14]. A total of 15 query modulation operators are identified and trained using the EM algorithm for a wide range of training questions. The best operators tailored to different questions are then identified and applied for later online question answering using the web search engines.

A relatively complete, general-purpose, web-based Q&A system is discussed recently in [1]. Similar techniques as in TREC conferences are applied to the web context. For example, the user question is processed by a parser to learn its syntactic structure. In order to help extract the answers and make the classification task easier, the questions are classified into three categories: nominal, numerical and temporal. Similar to the techniques used in [12, 7], various query modulation techniques, such as query expansion, noun phrase formation, transformation are applied to the initial questions to get high quality results for later answer extraction. Their answer extraction module utilize both IDF (inverse document frequency, a measure of the spread of a word or phrase among documents in a corpus) information and word distance to extract answers. These extracted answers are then assigned scores based on their contexts in documents and then further clustered into groups. The member with the highest score in each group is selected as representative for that group and is presented to the user as a potential answer.

Our system is different from [1] in that we did not use a deep natural language parser. As the authors admit on the Mulder Web page, such approaches are very slow and are not usable on the Web until faster parsers and/or hardware becomes available. In fact, the Mulder system runs on 100 workstations in parallel (Oren Etzioni and Daniel S. Weld, Personal Communication, 2002) to make it scalable. Instead, we replace these time-consuming parts with approaches based on rule-based classifiers and probabilistic phrase reranking and still achieve reasonable performance on a single Sun workstation. As will be shown later, these new additions offer a much more scalable approach for the web context than previous work.

3. A PROBABILISTIC APPROACH TO PHRASE RERANKING

3.1 The general architecture of our system

Our general approach includes the following stages:

- query modulation — that is the process of converting a question to an appropriate query. This is an optional stage. We have already described an approach for query modulation in [14]. In this paper we will present an approach that doesn't need query modulation to answer natural language questions.
- question type recognition — in this stage questions are subdivided depending on the type of answer that they expect: e.g., person, place, organization, distance, etc.
- document retrieval — the process of returning documents that are likely to contain, among other things, the answer(s) to the input question.

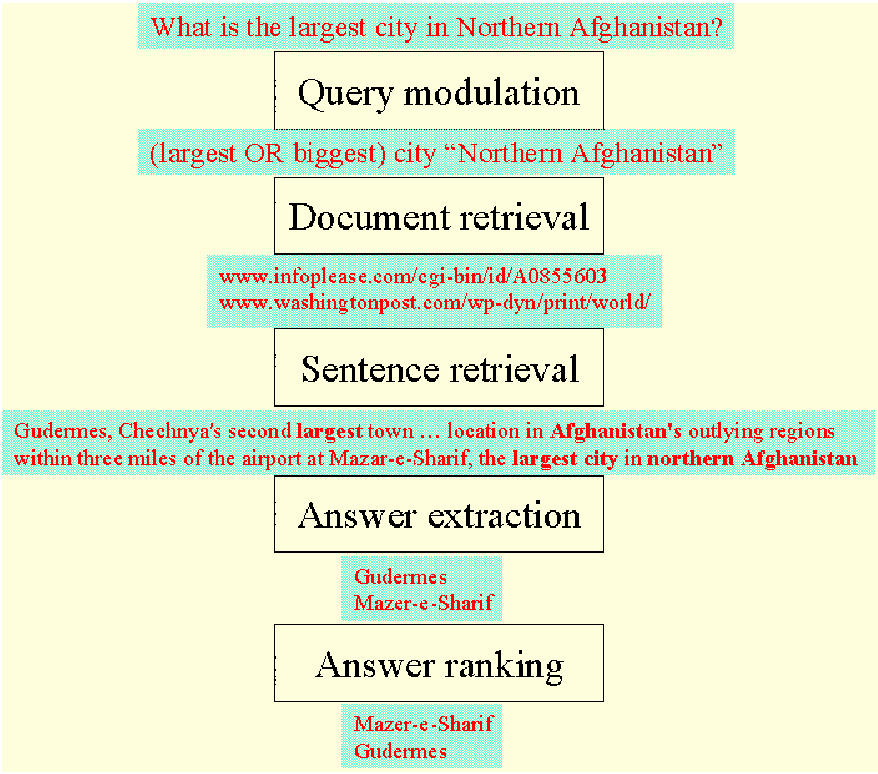

Figure 2: The architecture of the NSIR system.

- passage (or sentence) retrieval — the process of identifying in the retrieved documents the textual units that contains the answers.
- answer extraction — that is where the relevant sentences or passages are split up into constituent phrases, each of which is a potential answer candidate.
- phrase (answer) ranking — these phrases extracted in the previous stage are ranked with the goal of getting the right answer near the top of the phrase-level hit list.

3.2 Evaluation metric

As an evaluation metric, we use total reciprocal document rank ($TRDR$). The value of $TRDR$ is the sum of the reciprocal values of the rank of all correct documents among the top n (arbitrary) extracted by the system:

$$TRDR = \sum_{i}^{N} \frac{1}{rank_i} \quad (1)$$

where N is the number of returned documents (resp., passages or phrases). For example, if the system has retrieved 10 documents, of which three: the second, eighth, and tenth, contain the correct answer, $TRDR$ for that given paraphrase is $\frac{1}{2} + \frac{1}{8} + \frac{1}{10} = .725$. TRDR can be defined similarly for the sentence level or the phrase level. In these cases, it is based on the rank of the correct sentences (resp., phrases). The TRDR metric is similar to the metric used in the TREC evaluation. The difference between the two is that in TREC, only the top-ranked answer counts whereas, in our case, we want to be able to tell apart two runs, one of which only gets a correct answer in second place and another, which gets correct answers not only in second, but also in eighth and tenth places. Using TRDR rather than the metric employed in TREC, we are able to make such finer distinctions in performance. Another departure from the TREC scheme is our inability to perform a manual evaluation due to the large number of judgments that are needed to evaluate the different methods described in this paper. We need to rely instead on an automatic evaluation scheme in which a document (resp., sentence or phrase) is checked by a program for the presence (even unjustified) of the expected answer. This sort of evaluation is similar to the "lenient" evaluation described in [5]. In the next subsection, we will show that it is a reasonable compromise to use an automatic evaluation mechanism to measure performance in a large system. We should note here that for all experiments described in this paper we have used the TREC 8, 9, and 10 corpora from NIST [5]. They contain a total of more than 1,000 question/answer pairs. Unless otherwise indicated, we have evaluated our system on the 200 questions form TREC8. The Mulder system uses a different metric for evaluation, namely *user effort* which measures the total number of words shown on a page before the first correct answer is identified. We chose to use a metric closer in spirit to the established metrics in TREC.

3.3 Manual vs. automatic evaluation consistency

There are some cases where the document does match the answer patterns but does not actually support the answer. For example, "1987" is one of the answers for the question "When was London's Docklands Light Railway con-

structed"; automatic evaluation cannot tell if a document containing 1987 is really about when the railway was constructed or about other events happened in 1987, which a human can judge correctly. The question that arises is to what extent the automatic evaluation is consistent with human judges. So we did an experiment to examine the consistency between the results of automatic and manual evaluation.

We ran the first 100 TREC8 questions on the AlltheWeb search engine and evaluate the first 5 hits automatically and manually. Document-level TRDR performance scores are computed for each question and for both methods. We use the formula to get the Pearson correlation between the two data sets,

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{\left(n\sum x^2 - \left(\sum x\right)^2\right)\left(n\sum y^2 - \left(\sum y\right)^2\right)}} \qquad (2)$$

where x and y are the 100 reciprocal performance scores of manual evaluation and automatic evaluation, respectively. The Pearson correlation score derived from this formula is .538 which shows reasonably high correlation between the manual and automatic performance scores and, as a result, justifies the use of automatic evaluation when manual evaluation is too expensive (e.g., on tens of thousands of question-document pairs). Note that this automatic method for evaluation contrasts with the small-scale manual evaluation described in [12].

4. QUESTION TYPE IDENTIFICATION

To identify the semantic type of the question is an important step before extracting the actual answer. For example, a question like "Who was the tallest U.S. president?" expects a person as answer. Currently, we have 17 question types, listed in Table 4. Two methods have been implemented to categorize questions: decision rule induction using Ripper [15] and a heuristic rule-based algorithm. 1200 questions from TREC8, TREC9 and TREC10 are used in our experiment. The automatically identified question types are then compared against manually annotated question types.

Table 1: List of question types

PERSON	PLACE	DATE
NUMBER	DEFINITION	ORGANIZATION
DESCRIPTION	ABBREVIATION	KNOWNFOR
RATE	LENGTH	MONEY
REASON	DURATION	PURPOSE
NOMINAL	OTHER	

"Description" is used for questions seeking a description of some person, such as "Who was Whitcomb Judson?"; while questions like "Who invented the paper clip?" should be labeled as "Person" type. "Nominal" describes questions which have nominal phrase as answers but cannot be assigned to other specific categories such as person or organization. Questions not belonging to any of the above types fall in "Other".

4.1 Machine learning approach

Ripper is the machine learning tool used for question categorization. In our experiment, each question is represented by 13 features, 9 of which are semantic features based on WordNet. For instance, one of the semantic features is "ifNounIsMoney" which checks if the hypernyms of the first noun in the sentence contains money related words such as "monetary, economic, liability, and etc. The questions have also been processed by LTCHUNK [16], which yields the NumberOfNounPhrases features. All the features are listed in Table 4.1.

Table 2: Features for question representation

QuestionWords	Whword
WordBesideWhwords	ifWordBesideIsWealthy
ifNounIsMoney	ifNounIsLength
ifNounIsDuration	ifNounIsPerson
ifNounIsLocation	ifNounIsGroup
ifNounIsRate	ifNounIsNumber
NumberOfNounPhrases	

Several experiments have been done using Ripper for question type identification. Questions of TREC9, TREC8 and TREC10 have been incrementally added to the training data set. In addition, we manually added 37 data points to the training data set, which helps produce more robust rules. For example, ", how, many, n, , y, , , , , , , , LENGTH:4." means that questions which have wh-word "how many" followed by a "Length" noun should be categorized as a "Length" type. These manual data points are very helpful when Ripper uses training data set to produce the hypotheses. For the example data point, Ripper generates a hypothesis which can be represented as "LENGTH 5 0 IF (ifNounIsLength LIKE y) AND (wordbeside LIKE many)". So when Ripper predicts question types on testing data set, it will know that the questions such as "How many miles is it from London, England to Plymouth, England" are expecting a length as its answer. The Results of using Ripper to identify question types are listed in Table 3.

Table 3: Results (in error rate) of using Ripper to identify question types

Train	Test	Train Error	Test Error
TREC9	TREC8	22.4%	24%
TREC8,9	TREC10	17.03%	30%
TREC8,9,10	-	20.69%	-

4.2 Heuristic algorithm

The second method for type identification that we use to categorize questions is heuristic in nature. Question categorization seems trivial since the question word is often a good indication of the semantic type of the answer. However, this is not true even for the "who" questions. 484 TREC9 questions containing wh-words are examined with respect to the mapping between wh-words and question types.

As it can be from Table 4, only a small number of these wh-words can determine the question types, such as "when" and "why". "what" questions can cover almost all the different types. For the exceptions of who and where questions, we use the simple rules like listed in Table 5.

But for what/which questions, syntactic and semantic analysis is needed. In our system, Brill's transformation-based POS tagger is used to tag questions [17]. What/which can either be tagged as a WDT (determiner) as in "What state in the United States covers the largest area?", or as a

Table 4: Analysis of Wh-words and their corresponding types

Wh-word	Types
who (102)	PERSON(77) DESCRIPTION(19) ORG(6)
where(60)	PLACE(54) NOMINAL(4) ORG(2)
when(40)	DATE(40)
why(1)	REASON(1)
what /which(233)	NOMINAL(78) PLACE(27) DEFINITION(26) PERSON(18) ORG(16) NUMBER(14) ABBREVIATION(13) DATE(11) RATE(4) KNOWNFOR(8) MONEY(3) PURPOSE(2) REASON(1) TRANSL(1) LENGTH(1) DESCOTHER(10)
how(48)	NUMBER(33) LENGTH(6) RATE(2) MONEY(2) DURATION(3) REASON(1) DESCOTHER(1)

Table 5: Sample rules

Template	Types
who is <Person Name>	Description
who (manufacture|produce|grow|provide) ...	Organization

WP (wh-phrase) as in "What is the population of Japan?". The base noun phrase right after a WDT "what" can often be used to determine the question type. We select the last noun of this noun phrase as the informative noun which will be further used to determine the question type with semantic analysis. For example, in the question "What/WDT card/NN company/NN sells/VBZ Christmas/NNP ornaments/NNS ?/.", the first base noun phrase is "card company", then "company" becomes the informative noun for this question; we then categorize this question as "ORGANIZATION". Compared to WDT "what", questions with a WP what are more complicated for this task. First, the verbs in questions are used to categorize questions like "What caused the Lynmouth floods?" in which the verbs indicate the types; then questions are typed as "DEFINITION" if the number of question words is one or two excluding wh-words or any determiners; for the remaining questions, the system needs to extract the informative noun. Our general heuristics for finding the informative noun is to locate the last noun of the first base noun phrase. For example, in the question "What's the average salary of a professional baseball player?", we get two base noun phrases which are "the average salary" and "a professional baseball player", and we then use "salary" as the informative noun for this question. Different heuristics are used for questions like "What was the name of the first Russian astronaut to do a spacewalk?" or "What person's head is on a dime?". These heuristics have also been applied to the questions containing no wh-word, like "Name a film in which Jude Law acted".

The decision of some questions' types are left to the informative nouns. This is implemented through a lexicon built manually which maps nouns to their corresponding categories, such as length|circumference|diameter|diam|radius –> LENGTH. So if the informative noun is "diameter", then the question will be categorized as LENGTH type. WordNet has been used to build the mapping lexicon. Table 6 shows the result of using our heuristics to determine the question types. The first line is the results of the heuristics generated by only looking at TREC9 questions; the second is including both TREC8 and TREC9 questions; the last line shows the results of the heuristics modified based on all the three TREC question sets.

Table 6: Results (in error rate) of using heuristics to identify question types

Train	Test		
	TREC9	TREC8	TREC10
TREC9	7.8%	15%	18%
TREC8,9	7.4%	6%	18.2%
TREC8,9,10	4.6%	5.5%	7.6%

The accuracy has been greatly improved by using heuristics. When using Ripper, the training error rate is around 20% and the test error rate is even higher which goes to 30% when trained on TREC8,9 and tested on TREC10. As can be seen from Table 6, training error rate never goes above 8% and testing error is around 18%. It should be noted that the training error rate exists because some questions are really hard to categorize without any additional information. For example, questions like "Who won..." could expect a person as its answer such as "Who won the Nobel Peace Price in 1991", but it also could expect an organization such as "Who won the Superbowl in 1982".

5. DOCUMENT RETRIEVAL

We use the offline interfaces to three of the major search engines, AlltheWeb, Northern Light, and Google. The input to these interfaces is a query (or a question in our case). The output is a list of the URLs of the top matching documents. We use the Perl LWP::Download module to retrieve the actual documents before we split them up into sentences and/or phrases. We retrieve the top 40 documents from the search engine. In the evaluation section we will show our method's performance on the document level as well as the passage and phrase levels. Note: all retrievals used in this paper were performed in late October and early November of 2001.

6. SENTENCE RANKING

The purpose of sentence ranking is to reduce the computational complexity of the later phrase ranking stage. Producing phrases directly from the downloaded documents requires substantial computation.

To perform sentence ranking, two models were used in our system. One is based on an N-gram model. The other is based on the traditional Vector Space model.

For the N-gram model, the question submitted by a user is parsed to generate unigrams, bigrams, and trigrams. Various lexical statistics about these grams are used for sentence ranking. The formula for scoring sentences (passages) by proximity to the words of the query using the N-gram model is as follows:

$$Score = \frac{w_1 \sum_{i=1}^{N_1} tf_i * idf_i + w_2 \sum_{j=1}^{N_2} tf_j + w_3 \sum_{k=1}^{N_3} tf_k}{Normalized\_Factor} \quad (3)$$

where $N_i (i = 1, 2, 3)$ is the total number of occurrences of unigram, bigram, and trigram in a sentence. $w_i (i = 1, 2, 3)$ is the linear combination weight. We set the weights as 1, 1.5, and 4, respectively in our experiments. tf_i is the term frequency of i-gram. idf is the inverse document frequency,

which measures the rarity of a unigram. $Normalized\_Factor$ is defined as follows:

$$\begin{cases} 1 & \text{if } Sentence\_Length < 40 \\ Sentence\_Length/40 & \text{if } Sentence\_Length > 40 \end{cases}$$

Another sentence ranking function is designed based on modification of the Okapi ranking function used for document ranking [18]. It s defined as follows:

$$Score(S) = \sum_{T \in Q} \frac{3 \times tf \times idf}{0.5 + 1.5 \times \frac{Sentence\_Length}{Sentence\_Length_{avg}} + tf} \quad (4)$$

where $Sentence\_Length$ is the length of a sentence in words and $Sentence\_Length_{avg}$ is the average sentence length in the top 20 documents returned from a search. tf, idf is the same meaning as above in the linear combination formula. In the evaluation section we show the performance of NSIR on the sentence level.

7. PHRASE EXTRACTION AND RANKING

7.1 Potential answer identification

In our study, we convert all retrieved documents from the Web into chunks using an off-the-shelf chunker [16]. For a typical question, after downloading the top 40 hits from a given search engine, the chunker produces several tens of thousands of phrasal chunks. Here is an example. Given the question (from TREC 8) *Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"?*, 10,360 phrases are returned. Of these 10,360 phrases, 10 contain the correct answer "Hugo Young". However, these 10 phrases are scattered among the 10,360 phrases and need to be identified automatically. To address this issue, our current algorithm utilizes the following feature: proximity between the text and the question — that is, a phrase that contains most query terms gets a high score, however a phrase that is *near* a phrase that contains most query terms will get a slightly lower score. Figure 3 shows the effect of proximity on the score of a phrase. Phrases that are next to a large number of query words get higher scores than phrases that are further away.

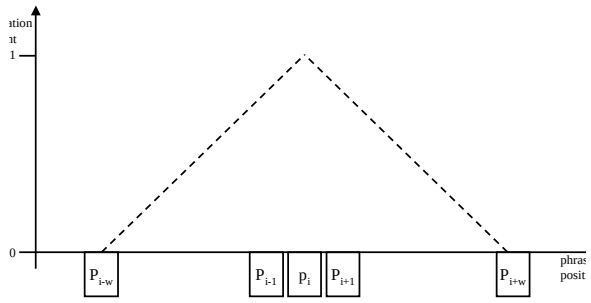

Figure 3: Proximity feature.

7.2 Phrase ranking

Answer Extraction and Ranking is the module that looks at additional sources of information to improve the performance of the previous module. We have been experimenting with the so-called part-of-speech "phrase signatures" which associate with a particular question or answer type. For example, the expected answer type for the question about Margaret Thatcher's biography is a person. We can compute the probabilities P(PHRASETYPE|SIG) for all possible signatures and phrase types. For example, the phrase signature for the phrase "Hugo Young" is determined by the chunker as "NNP NNP". The probability P(PERSON|NNP NNP) is .458. We will call this probability the *signature score*. For a given phrase, we can then multiply two numbers, the proximity score from the previous subsection and the signature score from the current subsection. Figure 4 shows some of the most common part of speech (POS) signatures and the phrase types that correspond to each of them. The part of speech symbols are generated by the text chunker [16]. For instance, VBD indicates the past participle form of a verb, JJ is an adjective, DT is a determiner (e.g., "the"), and "NNP" is a proper noun. Other Q&A systems use commercial grade named entity taggers. Our approach could also benefit from such taggers although phrase signatures seem to work quite well.

Note that the step of answer identification and ranking can be done based on sentences instead of the original downloaded documents. We will compare the performances of these approaches in the experiments section.

8. EXAMPLE

Let's consider question 21 from TREC8: *Who was the first American in space?*.[5] The expected correct answer is "Alan Shepard".

We submit this question to the Google search engine and get the following results back (see Figure 5). Note that a number of person names appear in the hit list. These include several occurrences of the correct answer as well as a many other names. The top 40 documents returned by Google for this question contain a total of 14,717 phrases, of which approximately 2% are names of people for a total of 300 names. Among these names, our system needs to pick out the instances of "Alan Shepard".

It is not difficult to see from Figure 5 that articles in the top 2 of the hit list are about the first American woman, not the first American in general. The correct answer appears in the third hit.

The output of our system at the sentence level is shown in Figure 6.

Figure 7 shows the top phrases retrieved by our system directly from the documents and using the proximity feature only. Figure 8 reflects the use of both features (proximity and qtype) but without taking into account the top-ranked sentences. Figure 9 shows how these phrases are re-ranked after taking into account the top sentences from the returned documents and also the qtype feature. As we will show in the following sections, such combination of features improves overall system performance by a factor of at least 2.

Let's consider an example. The top-ranked phrase in Figure 7 is "the Space Flight Operations contractor". Its sig-

[5]This question was used as a running example in the Kwok et al. paper as well.

Signature	Phrase Types
VBD	NO (100%)
DT NN	NO (86.7%) PERSON (3.8%) NUMBER (3.8%) ORG (2.5%)
NNP	PERSON (37.4%) PLACE (29.6%) DATE (21.7%) NO (7.6%)
DT JJ NN	NO (75.6%) NUMBER (11.1%) PLACE (4.4%) ORG (4.4%)
NNP NNP	PLACE (37.3%) PERSON (35.6%) NO (16.9%) ORG (10.2%)
DT NNP	ORG (55.6%) NO (33.3%) PLACE (5.6%) DATE (5.6%)

Figure 4: Common part-of-speech signatures and their corresponding phrase types. A NO means that this particular signature cannot be the answer to any type of question. The frequencies are derived from a manually annotated corpus of text unrelated to the TREC corpus.

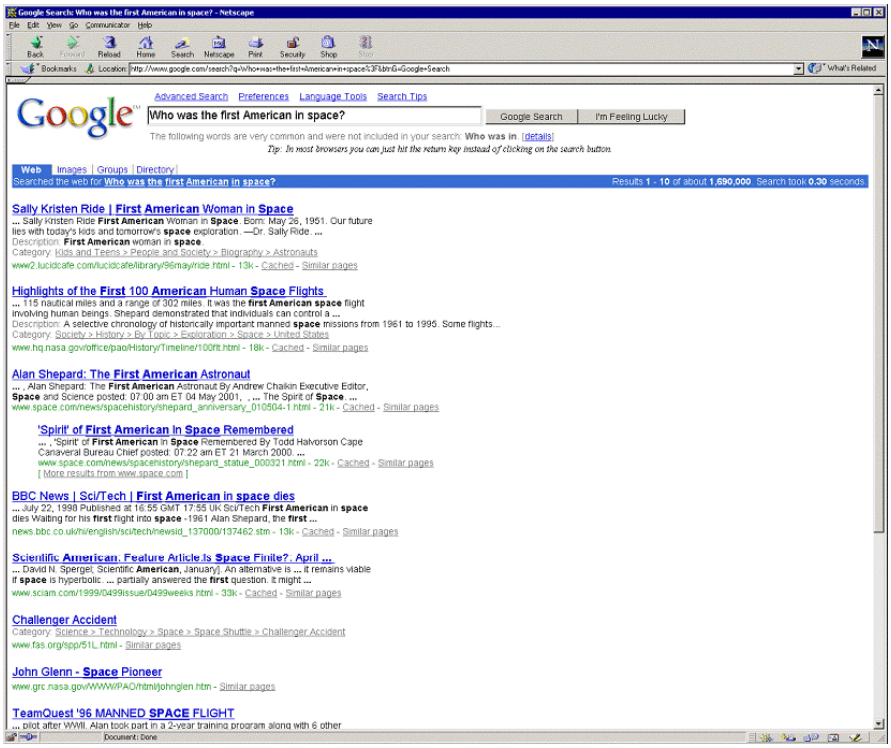

Figure 5: Google result given TREC8 Question 21.

American in space **Alan Shepard** Becomes the First American in Space 1961 On May 5 1961 **Alan Shepard** Jr. was launched into space.

Keep in mind that **Alan Shepard** was the first American in space not the first man.

First American woman in space: **Sally Ride** First American woman in space: **Sally Ride** **Sally Ride** was the first American woman to travel into space.

Submitted by Anonymous Contributor Title: First American woman in space: **Sally Ride** Description: Physicist **Sally Ride** was the first American woman in space participating in two missions aboard the Space Shuttle Challenger.

It was the first American space flight involving human beings.

Privacy Copyright Disclaimer First American in Space May 5 In 1961 I lived in a small town in Florida called Neptune Beach.

As a Mercury astronaut he was chosen to be the first American in space.

He was the first American to dare the unknown and enter space.

Figure 6: The top ranked sentences from our system (names of PERSONs are shown in bold face, the correct answer is boxed).

Rank	Probability and phrase
1	0.599862 the_DT Space_NNP Flight_NNP Operations_NNP contractor_NN ._.
2	0.598564 International_NNP Space_NNP Station_NNP Alpha_NNP
3	0.598398 International_NNP Space_NNP Station_NNP
4	0.598125 to_TO become_VB
5	0.594763 a_DT joint_JJ venture_NN United_NNP Space_NNP Alliance_NNP
6	0.593933 NASA_NNP Johnson_NNP Space_NNP Center_NNP
7	0.587140 will_MD form_VB
8	0.585410 The_DT purpose_NN
9	0.576797 prime_JJ contracts_NNS
10	0.568013 First_NNP American_NNP
11	0.567361 this_DT bulletin_NN board_NN
12	0.565757 Space_NNP :_:
13	0.562627 'Spirit_NN '_" of_IN
14	0.561702 space_NN
15	0.561064 February_NPN
...	
41	**0.516368 Alan_NNP Shepard_NNP**

Figure 7: Document + Phrase, proximity only, Question 21 from TREC 8. The highest-ranking correct answer is in position 41 out of 14,717 phrases when phrases are ranked solely based on proximity.

nature is "DT NNP NNP NNP NN". That signature is not associated with the expected question types "PERSON" or "PLACE". In other words,

$$\begin{aligned} & P(\text{"PERSON"}|\text{"DT NNP NNP NNP NN"}) \\ = \; & P(\text{"PLACE"}|\text{"DT NNP NNP NNP NN"}) \\ = \; & 0 \end{aligned}$$

As a result, the combined probability is 0. On the other hand, the highest-ranking "Alan Shepard" is in forty-first place out of 14,717 phrases with a proximity score of .516368.

When taking into consideration the qtype feature in addition to proximity, that phrase moves up to tenth place with a score of .376336. That score is the product of the proximity score .516368 and the qtype score .728814. How was the qtype score computed? It is equal to

$$P(\text{"PLACE"}|\text{"NNP NNP"}) + P(\text{"PERSON"}|\text{"NNP NNP"})$$

or in terms of actual numbers, .37288 + .35593. We should note here that we have modified the output of Ripper to specify the top two candidate categories for each question — that is, why we are getting both PERSON and PLACE as candidate question types for this question.

Finally, the phrase "Alan Shepard" moves even higher in the list (to seventh place) when the list of phrases is limited to the 1,935 phrases in the highest ranking 50 sentences returned by our sentence ranking component. Overall, our TRDR for this question is .18 (.14 + .03 + .01).

9. EVALUATION

9.1 Performance by question length

We ran an experiment, trying to compare performance based on the length of the question in words. We noticed a "V"-shaped behavior. The results are based on AlltheWeb, D+P (Document+Phrase, not going through extraction of sentences). For the shortest third of the questions, our TRDR is .13; for the middle third, it is .06; while for the longest third, it is .10. Our explanation is that really long questions contain a large number of content words that help identify the right documents and passages while it is easier to guess the question type for short questions.

9.2 Robustness in question type guessing

We should note that our performance depends dramatically on whether we get the question type right on our first or second guess. Out of 173 questions for which we guess the question type right on either guess, our performance (AlltheWeb, D+P) is .120 while for the 27 questions whose types we don't guess correctly, the performance is .007, or 17 times lower.

10. EXPERIMENTAL COMPARISON

We performed several experiments at different levels of granularity to study how we can effectively improve our chances of finding good answers from the search results. As we mentioned earlier, we have several different ways of pruning the search space to get the answers at different levels: document, sentence, and phrase.

We will next compare each of these ranking schemes from searches of all three search engines.

10.1 Document level performance

The performance at the document level from all three engines is summarized in Table 7, where "Avg." is the average of all total reciprocal scores. "# > 0" is the number of times (of 200 questions) that correct answers are identified in top 40 documents. Using the results from AlltheWeb as the baseline, we can see that both NorthernLight and Google did better overall than AlltheWeb in getting good answers, with Google completely dominant in this category. Using it improves chances of finding correct answers by almost 60%. Moreover, Google returns documents containing correct answers for 164 questions out of 200, 10% improvement over AlltheWeb. Google is the best engine as the potential answers source for web-based question answering.

10.2 Sentence level performance

Table 8 summarizes the performance results for sentence ranking using results from various search engines. The sentence ranking results using the linear combination formula is listed in the column under "Linear", and Similar results using the modified Okapi formula is listed under "Okapi".

In each cell of the row for "Avg", three values reported are: total reciprocal value for the top 50 sentences of each question, improvement over the upper-bound for a particular engine, and the improvement using one engine over the

Rank	Probability and phrase
1	0.465012 Space_NNP Administration_NNP ._.
2	0.446466 SPACE_NNP CALENDAR_NNP _.
3	0.413976 First_NNP American_NNP
4	0.399043 International_NNP Space_NNP Station_NNP Alpha_NNP
5	0.396250 her_PRP$ third_JJ space_NN mission_NN
6	0.395956 NASA_NNP Johnson_NNP Space_NNP Center_NNP
7	0.394122 the_DT American_NNP Commercial_NNP Launch_NNP Industry_NNP
8	0.390163 the_DT Red_NNP Planet_NNP ._.
9	0.379797 First_NNP American_NNP
10	**0.376336 Alan_NNP Shepard_NNP**
11	0.375669 February_NNP
12	0.374813 Space_NNP
13	0.373999 International_NNP Space_NNP Station_NNP
14	0.372289 Als_NNPS
15	0.371194 The_NNP Spirit_NNP

Figure 8: Document + Phrase, proximity + qtype, Question 21. The highest-ranking correct answer moves up to 10th place based on proximity and question type.

Rank	Probability and phrase
1	0.478857 Neptune_NNP Beach_NNP ._.
2	0.449232 February_NNP
3	0.447075 Go_NNP
4	0.437895 Space_NNP
5	0.431835 Go_NNP
6	**0.424678 Alan_NNP Shepard_NNP**
7	0.423855 First_NNP American_NNP
8	0.421133 Space_NNP May_NNP
9	0.411065 First_NNP American_NNP woman_NN
10	0.401994 Life_NNP Sciences_NNP
11	0.385763 Space_NNP Shuttle_NNP Discovery_NNP STS-60_NN
12	0.381865 the_DT Moon_NNP International_NNP Space_NNP Station_NNP
13	0.370030 Space_NNP Research_NNP A_NNP Session_NNP
14	0.366714 First_NNP American_NNP
15	0.359058 Sally_NNP Ride_NNP Sally_NNP Ride_NNP

Figure 9: Document + Sentence + Phrase, proximity + qtype, Question 21. The correct answer moves to 6th place when a sentence filter is used before extracting answer phrases.

Table 7: The performance comparison among search engines at the document level. A score of 1 means that either the top ranked document is the single correct answer or that the correct hits have a sum of reciprocal ranks equal to 1 (e.g., 2, 4, 5, 20).

Search engine	AlltheWeb	NorthernLightn	Google
Avg.	0.8355	1.0495	1.3361
Compared with AlltheWeb	-	25.61%	59.92%
#>0	149	163	164
Compared with AlltheWeb	-	9.40%	10.07%

Table 8: Sentence ranking comparison for TREC 8 questions

Engine	AlltheWeb			NorthernLight			Google		
	Upper	Linear	Okapi	Upper	Linear	Okapi	Upper	Linear	Okapi
Avg	2.1313	0.3128	0.2622	2.5258	0.4765	0.4397	2.5528	0.5400	0.4932
Compared with Upper Bound	-	-85.32%	-86.40%	-	-81.13%	-82.59%	-	-78.85%	-80.68%
Compared with AlltheWeb	-	-	-	-	52.33%	51.73%	-	72.63%	70.19%
#>0	148	99	99	159	121	119	159	137	135
Compared with Upper Bound	-	-33.11%	-33.11%	-	-23.90%	-25.16%	-	-13.84%	-15.09%
Compared with AlltheWeb	-	-	-	-	22.22%	20.20%	-	38.38%	36.36%

baseline engine — "AlltheWeb". The upper-bound performance is obtained by calculating the total reciprocal score for all those sentences containing the answer with the assumption that these sentences are ranked at the top of list. For example, if there are total 5 sentences containing the answer, then the upper-bound for the sentence ranking is (1+1/2+1/3+1/4+1/5). An ideal phrase ranking algorithm would achieve the upper bound. Giving upper bounds at each stage (document, passage, phrase) allows for the performance at each stage to be measured separately. Similar upper-bound definition is used for phrase ranking evaluation discussed in next subsection.

As can be seen from the TREC 8 results (shown in Table 8) that Linear combination sentence ranking, using sentences extracted from top 20 search results from Google, gives the best results. Similarly, the row for "#>0" reports

the number of questions where the correct answers have been found in the sentences produced by the sentence ranking formula, along with the comparison with the upper-bound and that of the baseline engine of "AlltheWeb". Again using the linear combination formula with Google engine gives the best result: 137 out of 200. Note that we lose some of the correct answers for 22 (159–137) questions. This is due to the fact that some of the sentences which contain the answers are not ranked at the top according to the sentence ranking formula. Our later manual evaluation of these results show that some of these problems are caused by our simple sentence segmentator while others simply reflect the existence of spurious answers by coincidence.

10.3 Phrase-level performance

The phrase level performance is shown in Table 9. "D+P" means phrases are generated and ranked from the original downloaded documents. "D+S+P" means phrases are created and ranked from top 50 sentences after sentence ranking. Four different types of performances are reported. The upper-bound definition is similar to the one used in sentence ranking. "Appearance Order" means that the phrases are ordered based on the appearance position of these phrases in original documents. "Proximity" utilizes only the proximity information, i.e., the overlap between a phrase and a user query, to rank phrases. "Proximity and qtype" uses both the proximity and the signature information to do the phrase ranking, which is essentially our proposed PPR (probabilistic phrase ranking) approach.

As can be seen in Table 9, Google again is the best in D+P (document + phrase) performance category among the three search engine examined. A further test using D+S+P (document + sentence + phrase) on Google shows that the performance is dramatically improved over the "D+P" approach, which demonstrates that the application of sentence ranking can indeed improve the performance of question answering. The final TRDR performance of NSIR is 0.199. As a comparison, the value of MRDR would be 0.151 which is significantly lower than TREC (the top performance at TREC 8 was around .40). There are multiple reasons for this discrepancy. First, TREC 8 questions were pulled directly from the 2-GB TREC corpus by rephrasing existing sentences. Second, all questions were guaranteed to have answers in the corpus. Third, the TREC corpus consists only of clean news articles while the Web contains significantly more heterogeneous and dirty texts.

11. CONCLUSIONS AND FUTURE WORK

We presented a probabilistic method for Web-based Natural Language Question Answering. It has been implemented in a robust system and has been tested on a realistic corpus of questions.

One thing we didn't address in this paper is the scalability issue. Even though the current system performs relatively faster than other web-based question answering system, the current system's performance for real-time question answering remains to be improved. One thing deserves further attention is that, after extensive testing and manipulation on top of various search engines, we found out that many pro-processing steps such as page downloading, sentence segmentation, part of speech tagging, etc., take most of the response time. Even though, parallel processing can be used to speed up the downloading phase, the dependence of existing web search engines as answer sources is really the bottleneck of our system. The performance could be expected to significantly improve if we have a pre-built snapshot of a search engine's content.

NSIR currently takes between 5 and 30 seconds per question depending on the (user-specified) number of documents to be downloaded from the Web and on the (again user-specified) number of phrases to extract. The current version of NSIR doesn't include query modulation [14] (the process of converting a question to the best query for a given search engine).

Our future work is to add query modulation to the system, fine tune various phrases of the system, and experiment with additional heuristics in answer selection. Currently, the PPR approach is not working as well as what we expected mostly due to the simple sentence segmentation and POS tagging and text chunking. We plan to combine the PPR approach with other efficient heuristics to further improve the final performance of our system.

12. ACKNOWLEDGMENTS

We would like to thank Bojan Peovski and Michael Gimbel for help with data annotation, and Airong Luo for proofreading.

Author biographies

Dragomir Radev is Assistant Professor of Information, Computer Science and Engineering, and Linguistics at the University of Michigan. He has a Ph.D. in Computer Science from Columbia University. Before joining Michigan, he worked at IBM's T.J. Watson Research Center. Dragomir's interests are in Information Retrieval and Natural Language Processing, more specifically Text Summarization and Question Answering.

Weiguo Fan will join Virginia Tech as an Assistant Professor of Information Systems and of Computer Science starting in August 2002. He is currently finishing his Ph.D. in the Department of Computer and Information Systems, Business School, University of Michigan. He has a M.S. degree in Computer Science from the National University of Singapore. His research interests include Information Retrieval, Data and Text Mining, Knowledge Management, Information Integration and Question Answering.

Hong Qi is a Ph.D. Candidate at the School of Information, University of Michigan. She holds an M.S. degree in Information Science from Beijing University. She is interested in Natural Language Processing and Question Answering.

Harris Wu is a Ph.D. student in the Department of Computer and Information Systems, Business School, University of Michigan. He has a M.S. degree in Computer Science from Florida State University. His interests include Information Retrieval and Question Answering.

Amardeep Grewal is an undergraduate student at the University of Michigan pursuing his Bachelor's in Computer Science. He attended the Bronx High School of Science in New York City. He is currently working on the analysis of the results yielded by the Q&A project in order to determine the weak points in the system.

Table 9: Phrase ranking performance

	AlltheWeb D+P	NorthernLight D+P	Google D+P	Google D+S+P
Upper bound	2.176	2.652	2.698	1.941
Default (Appearance Order)	0.026	0.048	0.068	0.0646
Proximity	0.038	0.054	0.058	0.0646
Proximity and qtype	0.105	0.117	0.157	0.199

13. REFERENCES

[1] Cody Kwok, Oren Etzioni, and Daniel S. Weld. Scaling question answering to the web. In *the Proceedings of the 10th World Wide Web Conference (WWW 2001)*, Hong Kong, 2001.

[2] Steven Abney, Michael Collins, and Amit Singhal. Answer extraction. In *the Proceedings of ANLP 2000*, 2000.

[3] 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.

[4] Julian Kupiec. Murax: A robust linguistic approach for question answering using an on-line encyclopedia. In *the Proceedings of 16th SIGIR Conference*, Pittsburgh, PA, 2001.

[5] Ellen Voorhees and Dawn Tice. The TREC-8 question answering track evaluation. In *Text Retrieval Conference TREC-8*, Gaithersburg, MD, 2000.

[6] J. Prager, D. Radev, E. Brown, and A. Coden. The use of predictive annotation for question answering in trec8. In *NIST Special Publication 500-246:The Eighth Text REtrieval Conference (TREC 8)*, pages 399–411, 1999.

[7] Dragomir R. Radev, John Prager, and Valerie Samn. Ranking suspected answers to natural language questions using predictive annotation. In *the Proceedings of 6th Conference on Applied Natural Language Processing (ANLP)*, Seattle, Washington, 2000.

[8] E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C-Y Lin. Question answering in webclopedia. In *NIST Special Publication 500-249: The Ninth Text REtrieval Conference (TREC 9)*, pages 655–664, 2000.

[9] C. L. A. Clarke, G. V. Cormack, D. I. E. Kisman, and T. R. Lynam. Question answering by passage selection (multitext experiments for trec-9). In *NIST Special Publication 500-249: The Ninth Text REtrieval Conference (TREC 9)*, pages 673–683, 2000.

[10] S. Harabagiu, D. Moldovan, R. Mihalcea M. Pasca, R. Bunescu M. Surdeanu, R. Gîrju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *NIST Special Publication 500-249: The Ninth Text REtrieval Conference (TREC 9)*, pages 479–488, 2000.

[11] D. R. Radev, K. Libner, and W. Fan. Getting answers to natural language queries on the web. *Journal of the American Society for Information Science and Technology (JASIST)*, 53(5), 359–364, 2002.

[12] Eugene Agichtein, Steve Lawrence, and Luis Gravano. Learning search engine specific query transformations for question answering. In *the Proceedings of the 10th World Wide Web Conference (WWW 2001)*, Hong Kong, 2001.

[13] Eric J. Glover, Gary W. Flake, Steve Lawrence, William P. Birmingham, Andries Kruger, C. Lee Giles, and David M. Pennock. Improving category specific web search by learning query modifications. In *The Proceedings of Symposium on Applications and the Internet, SAINT 2001*, San Diego, California, 2001.

[14] Dragomir R. Radev, Hong Qi, Zhiping Zheng, Sasha Blair-Goldensohn, Zhu Zhang, Weiguo Fan, and John Prager. Mining the web for answers to natural language questions. In *the Proceedings of ACM CIKM 2001: Tenth International Conference on Information and Knowledge Management*, Atlanta, GA, 2001.

[15] William W. Cohen. Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 709–716, Menlo Park, August 1996. AAAI Press / MIT Press.

[16] Andrei Mikheev. Document centered approach to text normalization. In *Proceedings of SIGIR'2000*, pages 136–143, 2000.

[17] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–566, December 1995.

[18] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-4. In D. K. Harman, editor, *Proceedings of the Fourth Text Retrieval Conference*, pages 73–97. NIST Special Publication 500-236, 1996.