Factoid Question Answering over Unstructured and Structured Web Content

Silviu Cucerzan and Eugene Agichtein
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{silviu,eugeneag}@microsoft.com

Abstract

We describe our experience with two new, built-from-scratch, web-based question answering systems applied to the TREC 2005 Main Question Answering task, which use complementary models of answering questions over both structured and unstructured content on the Web. Our approaches depart from previous question answering (QA) work in several ways. For unstructured content, we used a web-based system with novel features such as web snippet pattern matching and generic answer type matching using web counts. We also experimented with a new, complementary question answering approach that uses information from the millions of tables and lists that abound on the web. This system attempts to answer factoid questions by guessing relevant rows and fields in matching web tables and integrating the results. We believe a combination of the two approaches holds promise.

1 Introduction and Previous Work

The systems described in this paper are entirely web-based and explore two different research directions: one is to employ a web search engine to mine text web pages (*unstructured web information* henceforth), the other to employ the already *structured web information* in the form of html tables, which typically summarize various relations of interest to web users.

There has been a substantial amount of work on using web information and search engines for TREC QA, starting from the premise that a data collection such as the TREC corpus has considerably less answer redundancy than the web and thus, it is easier to match a question to the web data, to extract answers from the matching text, and then project these answers on the restricted data collection (e.g. Brill et al. [2], Radev et al. [12], Ramakrishnan et al. [13]).

Most of the published web-based QA systems focus on one language (English) and employ advanced natural language processing tools and/or extensive hierarchies of answer matching rules and answer types. From a practical perspective (i.e., having a search engine handle natural language questions in all markets in which it is deployed), such assumptions cannot be made. While previous approaches investigated how to scale current paradigms to general QA on the web (e.g. Kwok [11]), one of our main goals was to determine what performance can be achieved with a moderate annotation effort (in our case, one person-day) by a web-based QA system.

Because previous research on question answering largely ignored existing html tables and focused either on natural language text from web pages or online databases, another important goal of this work is to investigate a new way of using the existent structured information on the web to retrieve answers to factoid questions. By exploiting the explicit tabular structures created by the web document authors, we can, in principle, get natural language understanding “for free” and hence, advance the applicability and scalability of question answering.

There have been many efforts to extract structured information from the web. Previous approaches (e.g., Agichtein and Gravano [1], Etzioni et al. [6]) focused on extracting specific relationships (e.g., "*is a*"), which can then be used to answer the specific questions that these relationships support (e.g., "*who is X*"). In this work, we attempt to support any question by finding the structured table(s) on the web where this question was already answered. Unfortunately, many of the most useful tables do not contain the text patterns these systems look for. By indexing “all” potentially useful tables we are more likely to achieve high coverage of user’s questions.

In a closely related study, Hildebrandt et al. [9] used a large number of dictionaries and lists, some of which were constructed dynamically by querying sites such as Amazon. Unlike in our approach, the lists were specific and were constructed in advance for each question type. To the best of our knowledge, our study is the first attempt to integrate, index, and exploit millions of tables for question answering.

2 Systems Description

2.1 Common System Architecture

Figure 1 outlines the architecture of the two proposed systems. In this section, we describe the pre-processing and post-processing blocks common to both systems. Sections 2.2 and 2.3 present in detail the novel features of these systems.

In a first step, a question to be answered is passed through a phrase chunker (derived from the parsing system described in Heidorn, 2000) to extract information about the verbs, pronouns, and noun phrases in the question. The pronoun and noun phrase information is used to resolve the references to the question target, based on a small set of resolution heuristics. The verb information is used further by the WSQA system, as it will be described in Section 2.2.

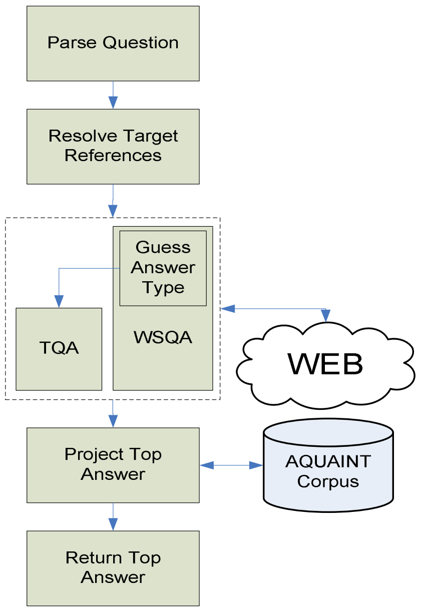

Figure 1. Overall System Architecture

Then, the WSQA system attempts to guess the answer type by using rewrite rules, as described in Section 2.2. This information is provided to both WSQA and TQA systems, which attempt to answer the question using unstructured and respectively, structured web content. Optionally, the lists of candidate answers returned by the WSQA and the TQA systems can be combined. We employed a linear mixture strategy that combines the lists using the normalized score associated with the answers by the two systems, the overlap between proposed answers, and the expected accuracy of each of the systems.

The top candidate answer is then *projected* onto the AQUAINT corpus to find document support. The projection was done by simply retrieving the document that best matched both the query and the candidate answer. Finally, the highest scoring answer and its support document are returned.

2.2 Mining Unstructured Web Content (WSQA)

For unstructured content, we used a web-based QA system inspired from the AskMSR structural design (Brill et al. [2]), which does not make use of advanced NLP tools. This system employs two novel ideas related to generic answer type matching using web counts and web snippet pattern matching. The former is proposed as an alternative to employing a predefined ontology of answer types, while the latter reduces the number of candidate answers that co-occur frequently with the question words in web search snippets but are not related to the question intent and also eliminates the need of an n-gram assembly stage, as employed in [2] and [5]. In the remainder of this section, we will focus mainly on the two novel features of the proposed system.

The WSQA system employs 198 question matching/rewriting rules, created by one human in one person day, based on the set of TREC-9 questions ([15]), a subset of the TREC 2004 questions ([14]), and another custom development set of 150 questions. The only syntactic information encoded in these rules concerns the verb identification. The system submitted to TREC 2005 uses a phrase chunker to derive the needed information, but additional experiments showed that very similar results can be obtained by employing just lexical and verb inflectional morphology information about the language.

The rules are sorted by question prefix (*when*, *where*, *what*, *which*, *who*, *how many*, *how much*, *how*, *in/on/by what*, and *name*). For each prefix, they are listed from the most particular to the most general. When presented with a question, the system tries each rewriting rule in order until a match is found or all rules are consummated. In the latter situation, a back-off strategy is applied, as described further in this section.

Each rewriting rule is composed of one Perl-like question matching pattern and one or more rewriting patterns. Each of the rewriting patterns contains a * symbol, which encodes the required position of the answer in the text with respect to the pattern. For example, the rewriting rule

```
When ~V<(is|was|are|were)> (.+) →
{
 Rewrite: $2 $1 on * AnsType: &DATE
 Rewrite: $2 $1 in * AnsType: &YEAR
}
```

transforms a question such as *When is Halloween?* into *Halloween is on* * `AnsType: &DATE` and *Halloween is in* * `AnsType: &YEAR`.

We defined 15 standard types (`DATE`, `MONTH`, `YEAR`, `PERSON`, `ORGANIZATION`, `PLACE`, `COUNTRY`, `NUMBER`, `TIME`, `DISTANCE`, `SPEED`, `WEIGHT`, `TEMPERATURE`, `CURRENCY`, and `PERCENTAGE`), each being described by a regular expression, for example (in Perl/C# notation):

```
PERCENTAGE: [0-9\.\,]+\s*(\%|percent)$
PERSON:  ^(([A-Z](\.|[a-z]+)\s+)+([a-z\-
]+)?)?(\s*[A-Z][a-z]+)+$
```

In some patterns, the answer type is represented by one of the match constituents in the regular expression instead of one of the standard types, e.g.:

```
What ~V<(is|are|was|were)> the (\S+)
(of|for) (.+) →
{
 Rewrite: $2 $3 $4 $1 * AnsType: $2
 Rewrite: * $1 the $2 $3 $4 AnsType: $2
}
```

According to the rewrite rule above, the question *What is the color of the sky?* is rewritten as *color of the sky is* * `AnsType:` *color* and * *is the color of the sky* `AnsType:` *color*. Here, *color* is a generic answer type, obtained automatically.

When the answer type is obtained in this way and is not mapped to one of the standard types, the system uses a web search engine to validate the matching of each answer candidate (e.g. *blue*, *blue blue*, *result*, *grey*, *usually*, *being*, *often*, *lake*, etc.) with the generic answer type (*color*). For each candidate *X* with an answer type *Y*, the following five quoted queries, are sent to the search engine:

```
"Y such as the X"
"Y such as X"
"X is a|an Y"
"X is the Y"
"X Y"
```

The score associated with each answer is boosted up based on the number of results returned by the search engine for each of these queries.

Additional hypernym patterns, for example, similar to those employed by Hearst [7], could be used. Unfortunately, for each additional pattern, the system has to perform a number of searches on the order of the number of candidate answers, which means that every additional pattern adds a considerable computational effort.[1]

We now describe the usage of the rewrite patterns by using as example the question *Q*: *Who killed Kennedy?* According to the first matching rule employed by our system, this question is rewritten as $R_1(Q)$: * *killed Kennedy*, $R_2(Q)$: *Kennedy was killed by* *, $R_3(Q)$: *Kennedy were killed by* *, and $R_4(Q)$: *Kennedy, killed by* *, all four rewrites having the answer type `PERSON`.[2]

The rewrites are sent as quoted queries to a search engine and the top *N* search result snippets are extracted (in our experiments, $N = 40$). Next, the system finds the location of the patterns in the snippets and hypothesizes as possible answers all word *n*-grams (in our experiments, $n = 6$) that appear in the position of the * in the pattern. For example, the rewrite $R_2(Q)$ retrieves from the snippet […] *Some people believe that Kennedy was killed by Lee Harvey Oswald , acting as a lone gunman* […] candidates such as *Lee*, *Lee Harvey*, *Lee Harvey Oswald*, *Lee Harvey Oswald acting*, *Lee Harvey Oswald acting as*, and *Lee Harvey Oswald acting as a*. We then count the occurrences of the valid candidates[3] in all snippets retrieved by the rewrite. Note that the longer candidate *Lee Harvey Oswald* will have a smaller count than its prefix *Lee Harvey,* which, in turn, will have a smaller count than the prefix *Lee*. In general, for a rewrite with the * symbol on the rightmost position (henceforth, *prefix pattern*), multi-word candidates will have smaller counts then their valid candidate prefixes. Similarly, for a rewrite with * on the rightmost position (henceforth, *suffix pattern*), multi-word candidates will have smaller counts than their valid candidate suffixes. Thus, the raw counts always favor shorter answers. To compensate for this problem, we employ two strategies. The first one is to reduce the count of an affix when the longer candidate's count is greater than a threshold, proportionally to the longer candidate's count (we name these as adjusted counts). The second one is to ensure, as much as possible, that each rewrite rule contains rewrite patterns in which the position of the * symbol varies (prefix, suffix, and infix), and thus, the correct answer receives counts from all types of rewrite patterns, while its prefixes and suffixes receive counts from only one type of patterns. For example, the rewrite $R_1(Q)$ (i.e. * *killed Kennedy*) retrieves as candidate answers *Oswald*, *Harvey Oswald*, and *Lee Harvey Oswald*,

[1] A number of searches for a given pattern can be avoided; for example, when a candidate answer contains as an affix another candidate for which no web results were found.

[2] Note that some patterns may be ungrammatical because the system does not use agreement information.

[3] In the provided example, the last three candidates are discarded immediately because they do not match the regular expression for the answer type `PERSON`.

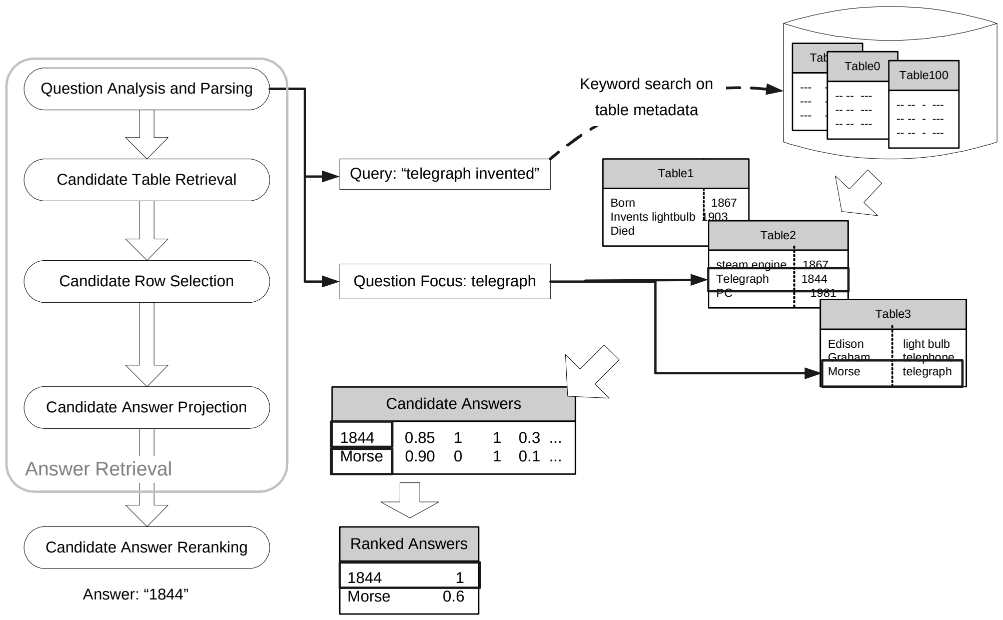

Figure 2.1: The Table Question Answering (*TQA*) framework

but not *Lee* and *Lee Harvey*, which were retrieved by the suffix pattern discussed previously. Typically, when the adjusted counts for candidate answers are aggregated from all rewrites, the complete, correct answers are obtained.

When no rewrite rule matches a question or no web search results for the matching rewrite are found, the system backs off iteratively to search queries that use bags of phrases (if phrase information is available), bags of bigrams, and finally, bags of words (in this case, the search queries do not contain quoted terms). All non-singleton *n*-grams in the web search snippets returned by the search engine that match the answer pattern are selected as candidate answers. The counts for candidates that are affixes of other candidates are discounted in the same manner as for * matches of rewrite patterns. When the answer type is derived from the question based on a rewriting rule, these counts are further adjusted using the validating strategy described earlier.

Because each question *Q* in TREC 2005 has a target *T*, we run WSQA in parallel on the original question Q and the target-resolved question Q(*T*). When using the rewrites for the original question, the target *T* is added as a quoted string to the queries sent to the search engine. The candidate answer lists are combined and the adjusted counts are added up. In this setting, the back-off system described in the preceding paragraph is employed only when the rewrites for both runs fail at retrieving a list of candidate answers. Employing a second run on the rewrites for the original query has two advantages: it may retrieve answers in which the target was referred in the same way it is referred in the question (e.g. pronoun), and it decreases the system's dependence on accurate target resolution.

2.3 Mining Structured Content on the Web (TQA)

Traditional question answering systems typically perform complex parsing and entity extraction for both queries and best matching Web pages, and maintain local caches of pages or term weights. Our approach is distinguished from these in that we explicitly target the *structured* content which presents new challenges and opportunities.

Consider the example question "*When was the telegraph invented?*" We often find that someone (perhaps, a history buff) has created a list of all the major inventions and the corresponding years. In our approach, we take this idea to an extreme, and assume that *every* query or factoid question can be answered by *some* relationship expressed in a structured content (e.g., an HTML table) on the web.

Unfortunately, HTML tables on the web rarely have associated metadata or schema. Hence, we use whatever metadata is available (e.g., surrounding text on the page, page title, and the URL of the originating page). More specifically, each structure is associated with a set of keywords that appear in the *header*, *footer*, and *title* of the page, and the first row of the table as *column headers*.
Once the structures are indexed, they can be used for answering questions. Our framework for answering factoid questions over this *implicitly structured* content is outlined in Figure 2.1.

To answer a question in our framework, we proceed conceptually as outlined in Algorithm 2.1:

```
1.  Retrieve all matching tables $R_T$ from the
    indexed tables $T_{ALL}$.
2.  For each table $t$ in $R_T$ select the rows $t_i$
    in $t$ that match the question target.
3.  Extract answer candidates $C$ from the un-
    ion of all selected rows.
4.  Assign a score and re-rank answer candi-
    dates $C_i$ in $C$ using the features associ-
    ated with $C_i$.
Return top $K$ answer candidates.
```

Algorithm 2.1: Answering factoid questions over structured web content.

For our example question, the appropriate semantic relationship is *InventionDate(Invention, Date*), and a table snippet could be one of many instances of *InventionDate* on the web. The selected rows in R_T would be those containing the question target, ``telegraph".

Initially, the question is chunked and automatically annotated with the *question target* and *answer type* as described above, and converted to a keyword query. The query is then submitted to the search engine over the metadata stored for our set of tables. The metadata information is indexed as regular text keywords, and the actual table content is stored as "non-indexed" text blob.

In this years' TREC QA evaluation, the question target was manually specified which allowed us to filter the candidate *rows* to include only those that match the target. The candidate answer strings were then filtered to match the answer type. For each surviving candidate answer string we construct a feature vector describing the source quality, the closeness of match, frequency of the candidate and other characteristics of the answer and the source tables from which it was retrieved. The answer candidates are then re-ranked using linear combination of (heuristically assigned) feature weights, and the top-scoring answer is returned as the output of *TQA*.

As a source of tables we used a random sample of 100M documents from the web, obtained from the *msnbot* (MSN Search) crawler, enhanced with the more focused crawl of likely fact-rich websites such as Wikipedia and FactMonster. Overall, more than 200 million tables were extracted and indexed for this evaluation.

3 Results and Conclusions

The systems described in this paper were developed from scratch for this year's TREC evaluation and were initially tested on the TREC 2004 factoid QA evaluation. On the development set, WSQA obtained 32.7% U-accuracy, while TQA was reached 14.9%.

The official performance of these systems in the TREC 2005 evaluation is reported in Table 3.1. As we can see, *WSQA* performs well on extracting answers from the web (U+R=21.8%) when the support for these answers in the TREC corpus is ignored. In contrast, *TQA* performs poorly, achieving only 5.5% exact answer accuracy even with no support required.

Unfortunately, the *WSQA* system timed out for almost 10% on the test questions in the official run, and a NULL answer was submitted for those questions, thereby decreasing the highest possible score achievable by our systems.

Because the proposed systems are *web*-based, we had to *project* the hypothesized web answers on the TREC data by retrieving the documents in the collection that " best support" the answers. We used a rather naïve projection strategy, and therefore, our official strict score suffered significantly due to inadequate support for otherwise correct answers. Specifically, the *WSQA* score dropped from 21.8% to 9.4% and the *TQA* score dropped from 5.8% to 1.7%. For the next TREC QA evaluation we plan to devote more effort into a more robust and effective answer projection and document-support finding component.

Tables 3.2 and 3.3 report the official performance results of WSQA and TQA broken down by question type ("wh-" word) and target type ("event" or "entity"). The accuracy of WSQA for "when" and "where" questions is substantially better than the accuracy on other types of questions.

System:	*WSQA*	*TQA*	Combined
Nr. right (R):	34	6	33
Nr. unsupported (U):	45	14	47
Nr. inexact (X):	10	1	7
Nr. wrong (W):	273	341	275
U + X + R accuracy:	24.6%	5.8%	24.0%
U + R accuracy:	21.8%	5.5%	22.1%
R accuracy:	9.4%	1.7%	9.1%

Table 3.1: Official results for the three runs submitted to the TREC 2005 QA Task.

(a)	Total	Correct (RU)		+ Partial (X)		Wrong
"When"	54	33	61.1%	2	64.8%	19
"Where"	40	12	30.0%	5	42.5%	23
"Who"	51	6	11.8%	1	13.7%	44
"What"	117	14	12.0%	1	12.8%	102
"Which"	4	0	0.0%	0	0.0%	4
"How"	60	7	11.7%	0	11.7%	53
Other	35	7	19.4%	1	22.2%	28
(b)						
Entity	266	62	23.3%	7	25.9%	197
Event	96	17	17.7%	3	20.8%	76
Overall	362	79	21.8%	10	24.6%	273

Table 3.2: Official results for *WSQA* broken down by (a) question prefix, and (b) target type.

(a)	Total	Correct (RU)		+ Partial (X)		Wrong
"When"	54	7	13.0%	0	13.0%	47
"Where"	40	3	7.5%	0	7.5%	37
"Who"	51	2	3.9%	0	3.9%	49
"What"	117	3	2.6%	1	3.4%	113
"Which"	4	0	0.0%	0	0.0%	4
"How"	60	4	6.7%	0	6.7%	56
Other	36	1	2.8%	0	2.8%	35
(b)						
Entity	266	14	5.3%	1	5.6%	251
Event	96	6	6.3%	0	6.3%	90
Overall	362	20	5.5%	10	5.8%	273

Table 3.3: Official results for *TQA* broken down by (a) question prefix, and (b) target type.

While the overall performance of TQA is low, we also analyze how this system performs for different question prefixes and target types (Table 3.3). Not surprisingly TQA performs better on the "when" questions (0.13) than on all other question types.

Finally, we analyze the accuracy of different systems for varying number of top *K* results examined. As shown in Figure 3.1, the accuracy numbers of all systems steadily improve. The Combined system (COMB) retrieved the correct answer 40% of the time in the top six and 50% of the time in the top 15. Clearly, an important direction for future progress is the improvement of answer ranking for all systems.

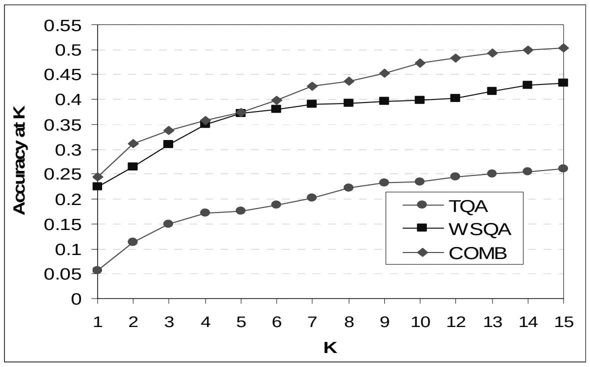

Figure 3.1: Accuracy at *K* for TQA, WSQA, and COMB.

In summary, we presented two new web-based question answering systems that were evaluated on the TREC 2005 Main Question Answering task. The systems use complementary models of answering questions over unstructured and structured content on the Web, respectively. Despite a rather poor showing in our first TREC evaluation, we believe that a combination of the two approaches holds promise.

Acknowledgments

We thank MSN Search for providing us access to their search engine. We also thank Eric Brill for extremely valuable discussions and feedback on this work.

References

[1] Agichtein, E. and L. Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections.

[2] Brill, E., J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-intensive question answering. In *Proceedings of TREC 2001*.

[3] Brill, E. and G. Ngai. 1999. Man vs. Machine: A Case Study in Base Noun Phrase Learning. In *Proceedings of ACL 1999*, pages 65-72.

[4] Caverlee, J., L. Liu, and D. Buttler. 2004. Probe, cluster, and discover: Focused extraction of qa-pagelets from the deep web. In *Proceedings of ICDE*.

[5] Dumais, S., M. Banko, E. Brill, J. Lin, and A. Ng. 2002. Web Question Answering: Is More Always Better? In *Proceedings of SIGIR 2002*.

[6] Etzioni, O., M. Cafarella, D. Downey, S. Kok, A. M. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2004. Web-scale information extraction in KnowItAll. In *Proceedings of WWW 2004*.

[7] Hearst, M. A. Automated discovery of wordnet relations. 1998. In *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

[8] Heidorn, G. 2000. Intelligent Writing Assistance. In *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*. Marcel Dekker, NY, 181-207.

[9] Hildebrandt W., B. Katz, and J. Lin. 2004. Answering Definition Questions with Multiple Knowledge Sources. In *Proceedings of HLT/NAACL 2004*.

[10] Keller, F., M. Lapata, and O. Ouriopina. 2002. Using the web to overcome data sparseness. In *Proceedings of EMNLP 2002*, pages 230-237.

[11] Kwok, C. C. T., O. Etzioni, and D. S. Weld. 2001. Scaling question answering to the web. In *Proceedings of the World Wide Web Conference 2001*.

[12] Radev, D., H. Qi, Z. Zheng, S. Blair-Goldstein, Z. Zhang, W. Fan, and J. Prager. 2001. Miningthe Web for Anwers to Natural Language Questions. In *Proceedings of CIKM 2001*.

[13] Ramakrishnan, G., S. Chakrabarti, D. Paranjpe, and P. Bhattacharyya. 2004. Is question answering an acquired skill? In *Proceedings of the World Wide Web Conference 2004*.

[14] Voorhees, E. M. 2004. Overview of TREC 2004. In *NIST Special Publication 500-261: The Thirteenth Text REtrieval Conference Proceedings (TREC 2004)*, pages 1-12.

[15] Voorhees, E. M. and D. Harman. 2000. Overview of the Ninth Text REtrieval Conference (TREC-9). In *NIST Special Publication 500-249: The Ninth Text REtrieval Conference (TREC-9)* 2000, pages 1-14.

[16] Zhu, X. and R. Rosenfeld. Improving Trigram Language Modeling with the World Wide Web. In *Proceedings of ICASSP 2001*, pages 592-597.