Exploiting Redundancy in Question Answering

Charles L. A. Clarke Gordon V. Cormack Thomas R. Lynam

Department of Computer Science, University of Waterloo, Canada
mt@plg.uwaterloo.ca

ABSTRACT

Our goal is to automatically answer brief factual questions of the form "When was the Battle of Hastings?" or "Who wrote *The Wind in the Willows*?". Since the answer to nearly any such question can now be found somewhere on the Web, the problem reduces to finding potential answers in large volumes of data and validating their accuracy. We apply a method for arbitrary passage retrieval to the first half of the problem and demonstrate that answer *redundancy* can be used to address the second half. The success of our approach depends on the idea that the volume of available Web data is large enough to supply the answer to most factual questions multiple times and in multiple contexts. A query is generated from a question and this query is used to select short passages that may contain the answer from a large collection of Web data. These passages are analyzed to identify candidate answers. The frequency of these candidates within the passages is used to "vote" for the most likely answer. The approach is experimentally tested on questions taken from the TREC-9 question-answering test collection. As an additional demonstration, the approach is extended to answer multiple choice trivia questions of the form typically asked in trivia quizzes and television game shows.

1. QUESTION ANSWERING

Question Answering (QA) has recently received attention from both the Information Retrieval (IR) and Information Extraction (IE) communities, both as an extension to their traditional interests and as an area of intersection between the two. A typical QA task requires concise answers to short factual questions, where a large target corpus is used as the source for these answers. Answers may take the form of values, names, phrases, sentences, or brief text fragments. In contrast with traditional IR tasks, it is not acceptable for a QA system to retrieve a full document, or even a paragraph or large text fragment, in response to a question. In contrast with traditional IE tasks, no pre-specified domain restrictions are placed on the questions, which may be of any type and deal with any topic [3,9,19].

Over the past two years, work in question answering has been encouraged by the inclusion of a question answering task in the experimental work of the TREC conference series, which is sponsored annually by the U.S. National Institute of Standards and Technology (NIST) [22]. For TREC-9, the most recent conference, the task consisted of 682 questions posed over a 3GB target corpus comprised of newspaper articles, where an answer for each question was guaranteed to appear in the corpus [21]. Answers take the form of text fragments extracted from the target corpus. For each experimental run, five attempts were given to answer each question. The attempts were ranked, and the primary evaluation measure is based on the rank of the first fragment containing a correct answer. Two types of experimental runs were permitted, which differed only in the maximum allowed length of the text fragment, which was either 50 or 250 bytes.

Figure 1 provides a simplified overview of our QA system, which was used to generate our TREC-9 experimental runs. First, a parser analyzes a question to extract two types of information: 1) a query for submission to a passage retrieval component, and 2) a set of selection rules that guide the process of extracting answers from the passages. The passage retrieval component executes the query over the target corpus, retrieving a ranked list of the top k passages for further analysis by the answer selection component. The selection rules include an answer category ("person", "monetary value", "date", etc.) that identifies the general type of information that is sought by the question. The answer selection component identifies possible answers ("candidates") from the passages and then ranks these candidates using a variety of heuristics. These heuristics take into account the number of times each candidate appears in the retrieved passages, the location of the candidate in the retrieved passages, the rank of the passages in which the candidate appears, the likelihood that the candidate matches the assigned answer category, and other special-case information provided by the selection rules.

This general approach of question analysis followed by IR followed by IE is nearly ubiquitous in QA systems [1,4,8,10,13,14,16,20,23]. Using relatively simple question parsing and answer selection components our QA system provides good performance. For the TREC-9 QA task [21] the system placed in the top three for both 50- and 250-byte runs.

Our experience with the TREC QA task indicated that three specific features make the greatest contribution to the

SIGIR'01, September 9-12, 2001, New Orleans, Louisiana, USA

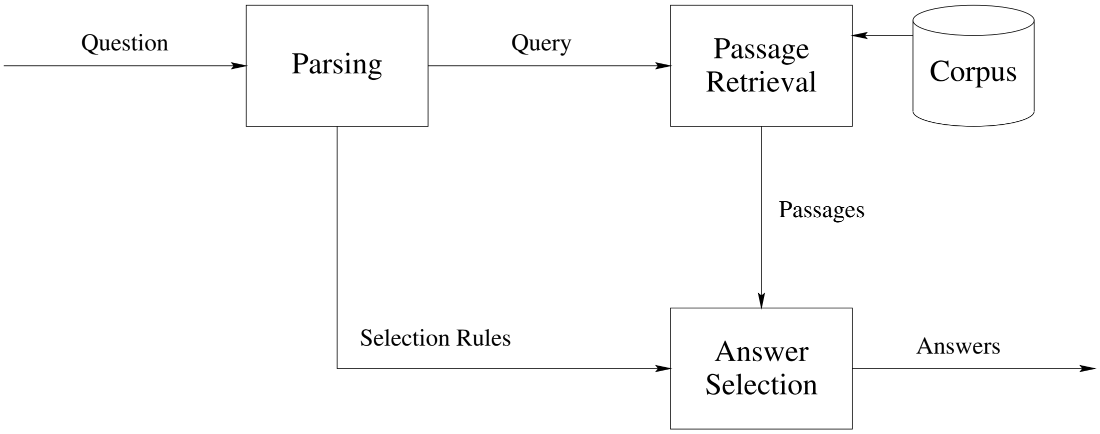

Figure 1: QA system architecture

performance of our system. One is the flexibility of the parser [5], which was able to make reasonable choices in the face of unexpected question types and formats. The second is the basic passage retrieval technique, which produces high-quality passages for further analysis. The third is the use of candidate *redundancy* in the answer selection component. By redundancy, we mean the combination of evidence from multiple passages to identify the most likely answer. The appearance of a candidate answer in several of the top ranking passages is a strong indication of the importance of a candidate. The exploitation of redundancy in question answering is the main subject of the present paper.

The remainder of the paper is organized as follows: The next section provides details of our passage retrieval technique, which forms the basis for the remainder of the work. Section 3 outlines the role of redundancy in our TREC-9 experiments and discusses its contribution to the performance of our QA system. In Section 4 we examine the impact of redundancy in a broader context by removing the guarantee that the answer appears in the corpus and by requiring an exact answer, rather than a text fragment containing the answer. A large collection of Web data is used as the target corpus. The work is further extended in Section 5, which uses trivia questions drawn from a popular game show to further test the value of redundancy. The paper concludes with an overview of additional related research and a discussion of future work.

2. PASSAGE RETRIEVAL FOR QA

Many open-domain question answering systems incorporate an IR component that is used for passage retrieval [1,8, 10,13,14,16,23]. Paragraphs, sentences or n-word segments are treated as separate documents with document-oriented retrieval techniques are applied to retrieve them. However, it is not obvious that these document-oriented IR methods are ideally suited for use in a question answering environment. Text fragments can often supply the answer to a question even when the topic of the question is incident to the topic of the document or paragraph containing the fragment. For example, evidence supporting "James Boswell" as the answer to the question "Who was Samuel Johnson's friend and biographer?" is supplied by the text fragment "...self-appointed cultural commentator Stanley Crouch played Boswell to Marsalis's Samuel Johnson..." although jazz music is the primary concern of the document from which it is drawn. In a similar fashion, the current paper itself contains answers for this and a dozen other unrelated trivia questions.

For question answering we use a passage retrieval technique that does not depend on predefined passages, but can retrieve any document substring in the target corpus. The technique was applied to question answering in both our TREC-8 and TREC-9 experiments and was outlined in the associated papers [5, 7]. This section provides additional justification for the technique and briefly discuss its efficient implementation.

Each document D in the corpus is treated as an ordered sequence of terms:

$$D = d_1\ d_2\ d_3 ... d_m.$$

A passage from D is represented by an *extent* (u, v), an ordered pair of coordinates with $1 \leq u \leq v \leq m$ that corresponds to the subsequence of D beginning at position u and ending at position v

$$d_u\ d_{u+1}\ d_{u+2} ... d_v.$$

A query Q is generated from the original question and takes the form of a term set:

$$Q = \{q_1,\ q_2,\ q_3, ...\}.$$

An extent (u, v) *satisfies* a term set $T \subseteq Q$ if the subsequence of D defined by the extent contains at least one occurrence of each of the terms from T. An extent (u, v) is a *cover* for T if (u, v) satisfies T and the subsequence corresponding to (u, v) contains no subsequence that also satisfies T. That is, there does not exist an extent (u', v') with either $u < u' \leq v' \leq v$ or $u \leq u' \leq v' < v$ that satisfies T.

Given an extent (u, v) that is a cover for a term set $T = \{t_1,\ t_2, ...\}$ we wish to compute a score for the extent with respect to T that reflects the likelihood that an answer to the question is contained in the extent or appears in its close proximity. First, we model a document as a sequence of independently generated terms and assume that there is a fixed probability p_t of a term $t \in T$ matching at any given document location. Note that this simplifying assumption allows multiple terms from T to match at a particular location.

Given an extent (u, v) with length $l = v - u + 1$, the probability $P(t, l)$ that the extent contains one or more occurrences of t is

$$\begin{aligned} P(t,l) &= 1 - (1 - p_t)^l \\ &= 1 - (1 - lp_t + O({p_t}^2)) \\ &\approx lp_t. \end{aligned}$$

The probability that an extent (u, v) contains all the terms from T is then

$$\begin{aligned} P(T,l) &= \prod_{t \in T} P(t,l) \\ &\approx \prod_{t \in T} lp_t \\ &= l^{|T|} \cdot \prod_{t \in T} p_t. \end{aligned}$$

Finally, we estimate p_t from the term frequencies within the target corpus

$$p_t = f_t / N,$$

where f_t is the total number of times t appears in the target corpus and N is the total length of all documents in the corpus. The score for an extent of length l containing the terms $T \subseteq Q$ is the self-information of $P(T, l)$

$$\sum_{t \in T} \log(N/f_t) \; - \; |T| \log(l). \qquad (1)$$

Equation 1 assigns higher scores to passages whose probability of occurrence is lower. While a higher score does not directly imply a greater likelihood that the answer will appear in close proximity, empirical evidence suggests that this relationship holds. The runs we submitted for the TREC-8 QA experiments were based solely on this passage retrieval technique [7]. Instead of parsing the questions, we simply eliminated stopwords. Instead of answer selection, we simply submitted the 250-byte fragment centered on each extent. Despite the simplicity of the technique, answers for 63% of the questions appeared in the five fragments submitted and the system was among the six best-performing systems overall [22].

Given a query Q we generate all covers for all subsets of Q and rank them using equation 1. All but the highest ranking passage from each document are discarded and the top k are used for further analysis.

Implementation of the technique depends on a fast algorithm to compute all covers of all subsets of Q. An extent (u, v) is said to *i-satisfy* a query Q if the subsequence of D defined by the extent contains exactly i distinct terms from Q. An extent (p, q) is an *i-cover* for Q if and only if it i-satisfies Q and does not contain a shorter extent that also i-satisfies Q. The appendix of reference 6 presents an algorithm to generate J_i, the set of all i-covers of Q, in time $O(|Q| \cdot |J_i| \log(N))$. The set of covers for all subsets of Q is simply the union of the i-covers

$$\bigcup_{i=1}^{N} J_i.$$

3. TREC-9 EXPERIENCE

The importance of redundancy became apparent during the post-hoc analysis of our TREC-9 QA results. Our experimental runs for TREC-9 used the overall approach outlined in Figure 1. For passage retrieval we used the technique presented in Section 2, retrieving the top $k = 10$ passages and symmetrically expanding each about its *centerpoint* $C = (u + v)/2$ to 200 terms for further analysis. A complete discussion of the question parsing and answer selection components is provided in our TREC-9 paper [5]. In the remainder of this section, we focus on the use of redundancy in the answer selection component.

Since the goal of the TREC-9 QA experiments was to select 50- and 250-byte answer fragments from the retrieved passages, the answer selection technique used to generate our experimental runs does not attempt to identify candidates that are exact or complete answers. Instead, candidates are single terms, where the nature of these terms depends on the category of the question. For example, if a question asks for a proper noun, the candidates consist of those terms that match a simple syntactic pattern for proper nouns; if a question asks for a length, the candidates consist of those numeric values that precede appropriate units; and if a question cannot be classified, the candidates simply consist of all non-query and non-stopword terms appearing in the retrieved passages. After identification, each candidate term t is assigned a weight that takes into account the number of distinct passages in which the term appears, as well as the relative frequency of the term in the database,

$$w_t \; = \; c_t \log(N/f_t),$$

where $1 \le c_t \le k$ is the number of distinct passages in which t appears and represents the redundancy associated with the candidate.

The weights of the candidates are used to select 50- and 250-byte answer fragments from the retrieved passages. Each 50- or 250-byte fragment from the retrieved passages is considered as a possible answer. A score of each fragment was computed by summing the weights of the candidate terms that appear within it. Given a text fragment F and a set of candidates K the score for a fragment is

$$\sum_{t \in F \wedge t \in K} w_t \qquad (2)$$

This score is then modified using a number of heuristics that take into account the rank of the passage in which the fragment appeared, the location of the fragment relative to the centerpoint of the passage, and the selection rules generated by the parser [5]. These heuristics had only a minor (but positive) effect on the system's performance.

Once the highest-scoring fragment is selected, the weights of the candidates appearing in that fragment are reduced to zero. All fragments are re-scored and the highest-scoring fragment is again selected. This process is repeated until five fragments have been selected.

The weight assigned to a candidate combines a redundancy factor and a term-frequency factor. The individual contributions of each can be ascertained by setting one or the other to a constant value and measuring the resulting impact on question answering performance. Figure 2 compares three different formulations for the weight w_t using the TREC-9 QA test collection. The candidate weight $w_t = c_t \log(N/f_t)$ was used for our TREC-9 experiments

w_t	50-byte answers mean reciprocal rank	number correct	percent correct	250-byte answers mean reciprocal rank	number correct	percent correct
$c_t \log(N/f_t)$	0.390	349	51.2%	0.507	444	65.1%
c_t	0.345	314	46.0%	0.471	425	62.3%
$\log(N/f_t)$	0.241	248	36.4%	0.448	414	60.7%
raw passages	0.191	209	30.6%	0.464	436	63.9%

Figure 2: TREC-9 results

and was discussed above. The weight $w_t = \log(N/f_t)$ ignores candidate redundancy, essentially as if all candidates appeared an equal number of times in the passages. The weight $w_t = c_t$ treats all candidates as having the same term frequency and takes only redundancy into account. For comparison, the final row of the figure presents results for the 50- or 250-byte fragment centered on each of the top five extents, duplicating our TREC-8 "answer selection" technique.

Figure 2 reports two effectiveness measures. *Mean reciprocal rank* (MRR) is the standard TREC effectiveness measure reported by NIST for each TREC QA run [22]. To compute MRR, each question is assigned a score that is the inverse of the rank of the first fragment that is judged to contain a correct answer. If no fragment contains an answer, the question is assigned a score of zero. The scores of the questions are then averaged to produce the MRR value for the run. In addition to MRR, the figure reports the number and percent of questions for which a correct answer was found in the five fragments. These measures were computed by an automatic judging script developed by Voorhees [22].

The value of redundancy is most apparent in the 50-byte runs. Eliminating the redundancy factor reduces the MRR by 38% and the number of questions answered correctly by 28%. Eliminating the term frequency factor has a lesser effect, reducing MRR by 12% and the number of correct questions by 10%. The raw passages have an MRR 50% lower and contain the answers to 40% fewer questions. In contrast, the raw 250-byte passages exhibit good performance, which is harmed by answer selection if either the redundancy or term frequency factors are used alone.

4. EXPLOITING REDUNDANCY

For the TREC experiments, text fragments were reported as answers. To further explore the value of redundancy in question answering we simplified most of the components in Figure 1 to eliminate the experimental confounds associated with parsing, question categorization, and the use of selection rules. We focused on a single category of question, those that require the name of a person as the answer, and attempted use redundancy as a means of isolating the required name from the top ranking passages. To identify candidate answers we used a simple syntactic pattern that matches most names in written English, but which often matches other text, including many other proper nouns. Our hypothesis was that redundancy could used as a substitute for deeper analysis; that correct answers could be distinguished from other candidates solely by their repeated occurrence in the proximity of high-ranking passages.

We identified 87 questions from the TREC-9 QA collection that required the name of a person as the answer. Typical examples are "Who is the emperor of Japan?", "Name a female figure skater," and "Who wrote the book *Huckleberry Finn*?". For each question we generated a query from the question by using the traditional IR approach of eliminating stopwords.

To provide a greater potential for redundancy and to provide an opportunity to study the impact of database size on the results, we eschewed the 3GB TREC-9 QA corpus in favor a larger corpus, the TREC 100GB VLC2 corpus [12,21]. Apart from its size, this corpus differs from the QA corpus in two important respects. First, since the QA corpus consists of newspaper articles and the VLC2 corpus of arbitrary Web pages, we might generally expect documents in the VLC2 corpus to be of lower quality. Second, it is known that answers to all TREC-9 QA questions are present in the TREC-9 QA corpus, but this is not the case for VLC2 corpus. We were curious to discover the extent to which a relatively small collection of Web data could be used to answer a set of independently created questions.

A simple syntactic pattern was used to identify candidate answers. Any sequence of at least two *name tokens* separated by whitespace or a single hyphen was considered a candidate. A name token is defined as either an uppercase letter followed by one or more lowercase letters or an uppercase letter followed by a period (an initial). In addition, a capitalized stopword is not considered to be name token, and a candidate answer cannot consist entirely of initials. Such a simple pattern will reject many valid names ("Wm. Paterson", "k. d. lang", "Samuel To", "Pius XI", "Anne of Cleaves", "Fabio") and include many inappropriate candidates. While the identification of proper names, including the names of people, is a well understood problem [3,9] we felt that the benefits of transparency would outweigh the benefits of more accurate candidate identification. Here, our purpose is not to find the answer using any and all means, but rather to specifically examine the value of redundancy, where the ability of redundancy to overcome the low precision inherent in this crude candidate-identification technique is itself of interest.

For each query we retrieved the top k passages according to Equation 1. Each passage was expanded symmetrically about its centerpoint to w bytes. For the experiment we varied the parameters k and w, which will be referred to as *depth* and *width* respectively. Candidate answers were identified in the passages, and candidates that consisted entirely of query terms were eliminated. Each remaining candidate was assigned a score that was simply a count of the number of distinct passages in which the candidate appeared (c_t). Candidates were ranked according to this score, and

depth (k)	width (w)	mean reciprocal rank	number correct	percent correct
25	250	0.407	44	51%
	500	0.431	47	54%
	1000	0.471	51	59%
	2000	0.462	50	57%
50	250	0.435	48	55%
	500	0.442	47	54%
	1000	0.463	49	56%
	2000	0.444	50	57%
100	250	0.432	53	61%
	500	0.434	50	57%
	1000	0.446	51	59%
	2000	0.418	52	60%
best 5 from top passage		0.178	20	23%
best from top 5 passages		0.299	36	41%

Figure 3: Impact of redundancy

ties were broken by applying a simple rule that takes into account the distance of each candidate from the centerpoint of the passages in which it appears. For each passage in which a candidate appears, the distance from the centerpoint of the passage to the centerpoint of the closest occurrence of the candidate is recorded. The average of these distances is used to break ties, with candidates having smaller averages ranked higher.

Preliminary experiments with TREC-8 questions suggested that a depth of $k = 50$ and width of $w = 1000$ would produce reasonable results. Using these parameters, 49 (56%) of the 87 questions are answered correctly and for 34 (39%) a correct answer is ranked first. For example, the top five candidates for the question "Who is the emperor of Japan?" are

1. Emperor Hirohito
2. World War
3. Emperor Akihito
4. Hong Kong
5. Prime Minister

where the first and third candidates are considered correct. The mixture of invalid proper nouns and the inclusion of multiple valid answers is typical of the results. The top five candidates for the question "Name a female figure skater." are

1. Kristi Yamaguchi
2. Followup Name
3. Follow Ups
4. Tonya Harding
5. Nancy Kerrigan

where the first, fourth and fifth candidates are considered correct. The second and third candidates are artifacts created primarily by the inappropriate inclusion of the term "name" in the query and appear to be generated by a series of related postings to a message board. Additional independent data might tend to reduce the rank of these candidates while increasing the rank of the remaining candidates. Finally, the top five candidates for the question "Who wrote the book, *Huckleberry Finn*?" are

1. Mark Twain
2. Tom Sawyer
3. Huck Finn
4. Samuel Clemens
5. Connecticut Yankee

which includes both the author's pen name and his real name, as well as the principal characters from the novel.

Only exact answers identifying people were counted as correct. For example, the candidate "Anne Morrow Lindbergh Foundation" was not accepted as the answer to the question "Who was Charles Lindbergh's wife?", the candidate "Time Warner" was not accepted as the answer to "Who owns CNN?", and the candidate "Prime Minister" was not accepted as the answer to "Who is the leader of India?". Since the TREC-9 judging script was not designed to identify exact answers, the determination of correct answers required a combination of automatic and manual processing. An automatic script based on the TREC-9 judging script analyzed the output of each experimental run and flagged potentially correct answers. The results of the script were manually checked to determine that the answers consisted solely of a person's name. Finally, the runs were automatically checked for consistency, verifying that answers were judged in the same way for all experimental runs.

We ran the experiment for a range of depth and width values. The results are reported in Figure 3. In addition to the runs using redundancy the figure reports two other runs for comparison. Both runs represent other simple strategies for extracting candidate answers from the retrieved passages. The first, labeled "best 5 from top passage", consists of the five distinct candidates appearing closest to the centerpoint of the top passage retrieved for each question, with the candidates ranked according their distance from the centerpoint. This run is equivalent to using parameters $k = 1$ and $w = \infty$. The second, labeled "best from top 5 passages" consists of a single candidate selected from each of the the top five passages. For each passage, the candidate closest to the centerpoint is selected. If a candidate has already been selected from a higher-ranking passage, the next closest candidate is selected instead.

Within the range shown, changes to k and w produce only minor changes in the results, with a slight tendency to improve as k and w are increased. This stability might be considered surprising given the range of data sizes represented by the retrieved passages. With $k = 25$ and $w = 250$, less than 7KB of data is analyzed per question; with $k = 100$ and $w = 2000$ over 195KB of data is analyzed per question.

We originally choose to use a 100GB target corpus to in-

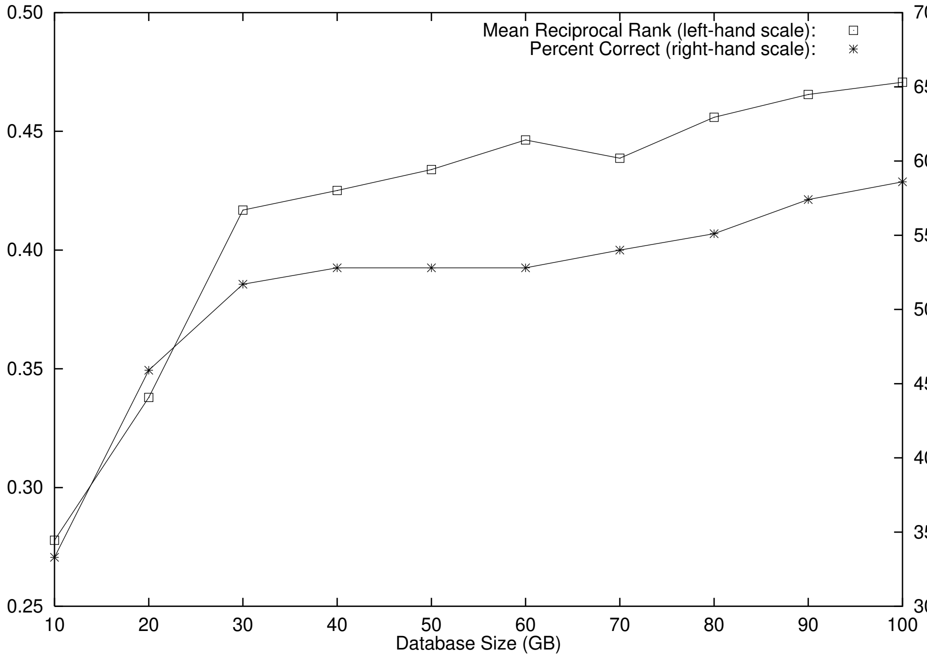

Figure 4: Influence of database size ($k = 25$, $w = 1000$)

crease the changes of finding multiple occurrences of answers in contexts that could be retrieved by the unmodified question keywords. In order to ascertain the impact that database size has on the results we ran the questions over various subsets of the full corpus. The documents in the database were ordered randomly and increasingly larger subsets of the database were used as the target corpus. The results are presented in Figure 4.

The potential remains for addition improvement if the database size were further increased. For 76% of the questions, a correct answer was reported in the top five by at least one of the runs reported in this section, and additional data may force more of these correct answers into the top five ranks. As expected, several of the questions required answers that the candidate selection pattern failed to match ("Dr. Suess", "Brutus", "William the Conqueror"). While the crude techniques used for query generation and candidate identification could easily be improved, we expect that the effects of these improvements would be largely orthogonal to the effects of redundancy and would be reflected in improvements to all runs.

5. PHONE A FRIEND

Who wants to be a Millionaire? is currently a popular game show airing in prime time on American television and duplicated locally in several other countries. As an additional application, the approach of the previous section has been extended to answer trivia questions posed to contestants on the show.

These questions are presented in a multiple choice format. For example, the question "Who was the first Prime Minister of Canada?" might be given with candidate answers "A) Pierre Elliot Trudeau", "B) John Graves Simcoe", "C) John A. MacDonald" and "D) Louis Riel". The prize money awarded for correctly answering a question ranges from \$100 to \$1,000,000. Contestants start with a \$100 question, and while they continue to answer questions correctly the value roughly doubles until the final \$1,000,000 question is reached. Contestants are eliminated if they answer a question incorrectly. They may also quit voluntarily before attempting to answer a question, keeping as prize money the value of the highest question correctly answered.

The existence of a closed set of candidate answers simplifies the voting process but complicates the passage retrieval process. While the question continues to be treated as a "bag of words" for retrieval purposes, each candidate answer is matched as a phrase and the terms in the answer do not contribute to the score of a passage. The retrieval engine was modified to return the highest scoring passages (according to Equation 1) that also contain a phrase match to one of the answers. For these questions, it is not unlikely that the weights of answer terms will be higher for incorrect answers, since some of the choices are deliberately bizarre, and are intended as jokes or red herrings. The answers can be treated as phrases, since it is not necessary to compute weights for their terms and the phrasing is usually the "common" form appearing in everyday speech and writing.

The top $k = 20$ passages were retrieved for each question. Since each passage is guaranteed to contain a match to at least one of the candidate answers, the passages were not expanded in width. A passage that contained a match for a candidate answer was counted as a vote for that candidate. Candidates were then ranked in terms of the number of votes each received.

We applied the approach to the 108 questions asked on the programs first broadcast on the ABC television network in the United States on October 31, November 1 and November 9, 2000. The results are shown in Figure 5. The figure

Question value	total questions	number correct	percent correct
\$100-\$1,000	48	36	75%
\$2,000-\$8,000	29	20	69%
\$16,000-\$64,000	22	15	68%
\$125,000-\$1,000,000	9	5	56%
OVERALL	108	76	70%

Figure 5: *Who Wants to be a Millionaire?*

splits the questions into four categories according to their value. Lower-valued questions are intended to be easier than higher-valued questions for human contestants, and this relationship held for our system as well.

Overall the system answered 76 (70%) of the questions correctly. For an additional 17 questions (16%) the correct answer was ranked second or tied for first. Three of the remaining questions essentially asked "Which of these things is not like the other?" where the correct answer was the only selection that did not exhibit the characteristics described in the question. In all three of these cases, the correct answer was ranked last. In one case the question referred to events that occurred after the 1997 Web crawl that created the corpus and it was answered in the context of that year. Three other questions had single-digit numeric values as answers. These values were treated as stopwords when the corpus was indexed and the questions actually caused the system to crash when executed. This behavior was treated as an incorrect answer.

6. CONCLUDING DISCUSSION

Redundancy can be exploited as a method for answer validation in question answering systems. An analysis of our TREC-9 QA experimental results indicates that redundancy played an important role in the process of selecting answer fragments from the top passages retrieved for each question. When exact answers are required, further experiments with simple voting algorithms demonstrate that redundancy is an effective method of ranking candidates answers.

The benefits gained from exploiting redundancy are largely independent of benefits gained from improvements to the other components of a QA system. Without redundancy, even the best QA system might be misled by a passage containing an inaccurate (or ironic) answer. Voorhees and Tice [22, page 203] provide an illustration of this problem in their discussion of responses to the TREC-8 question "Who was the first American in space?", where the corpus contained the passage "As for Wilson himself, he became a senator by defeating Jerry Brown, who has been called the first American in space."

Redundancy is available to be exploited. The experiments reported in Sections 4 and 5 used a modest-sized 100GB corpus of arbitrary Web data, where we did not know in advance how many questions had answers contained in this corpus. Apparently a corpus of this size is sufficient to answer roughly 70% of general-knowledge trivia-style questions using redundancy as the main answer selection technique. Web data is often viewed as being of low quality, and identifying high-quality pages is a major problem for web search systems [2, 24]. In the case of question answering, the quality of individual pages becomes less important since results from many low-quality pages can be combined to produce a consensus answer, and quantity can be used as an effective substitute for quality.

One of the few published answer selection algorithms to make explicit use of redundancy is the Werlect algorithm [17, 18] which was studied in the context of the GuruQA system [16]. The overall design of GuruQA is similar to that of our own system, and the Werlect algorithm shares some features with the algorithm presented in Section 3. Interestingly, the developers of the GuruQA system did not identify redundancy as a major contributor to the performance of GuruQA. In their reported experience, Werlect did not perform as well as a second answer selection algorithm, AnSel, which did not exploit redundancy. At TREC-9, GuruQA was among the five best-performing systems for both the 50- and 250-byte runs.

The MURAX QA system, described by Kupiec [15], incorporated a related form of answer validation that uses secondary queries to confirm relationships between candidates and question terms. Kupiec gives as an example the question "What film pits Humphrey Bogart against gangsters in the Florida Keys?". Text returned by an initial query suggests *The Big Sleep*, *Key Largo*, *The Maltese Falcon*, *Casablanca* and *The Treasure of the Sierra Madre* as candidates but does not indicate the setting or plot of these films. A secondary query using the phrases "Key Largo" and "Florida Keys" verifies a relationship between them that does not exist for the other candidates.

A very different form of answer validation is the abductive inference technique described by Harabagiu et al. [10] and incorporated in FALCON, the top-performing QA system at TREC-9 [11]. Given a question, a candidate answer, and the text fragment from which the candidate was extracted, abduction uses semantic knowledge to give a logical justification for its correctness. The candidate is rejected if no justification can be given.

In the experiments reported by this paper, we measure redundancy using a simple count of the number of passages in which a candidate appears. It may be possible to extend this measure to encompass other information. For each passage containing the candidate, this information includes the score of the passage, the rank of the passage and the minimum distance of the candidate from the passage's centerpoint. For TREC-9, our answer selection component took some of this information into account while making final adjustments to the scores of text fragments, but these adjustments were strictly *ad hoc* in nature. In future, we hope to undertake a theoretical and empirical examination of the use and value of this information. We continue to improve all aspects of our QA system for participation in TREC during 2001 and beyond, where a 5-year plan calls for the extension of the QA experiments in a variety of challenging directions [21].

7. REFERENCES

[1] Eric Breck, John Burger, David House, Marc Light, and Inderjeet Mani. Question answering from large document collections. In *1999 AAAI Fall Symposium on Question Answering Systems*, North Falmouth, MA, 1999.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh World Wide Web Conference*, Brisbane, Australia, April 1998.

[3] Claire Cardie. Empirical methods in information extraction. *AI Magazine*, 18(4):65–79, Winter 1997.

[4] Claire Cardie, Vincent Ng, David Pierce, and Chris Buckley. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering system. In *Sixth Applied Natural Language Processing Conference*, pages 180–187, 2000.

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

[6] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 36(2):291–311, 2000.

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

[8] Anne Diekema, Xiaoyong Liu, Jiangping Chen, Hudong Wang, Nancy McCracken, Ozgur Yilmazel, and Elizabeth D. Liddy. Question answering: CNLP at the TREC-9 question answering track. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[9] Ralph Grishman and Beth Sundheim. Design of the MUC-6 evaluation. In *6th Message Understanding Conference*, Columbia, MD, 1995.

[10] Sanda M. Harabagiu and Steven J. Maiorano. Finding answers in large collections of texts: Paragraph indexing + abductive inference. In *1999 AAAI Fall Symposium on Question Answering Systems*, pages 63–71, North Falmouth, MA, 1999.

[11] Sanda M. Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. FALCON: Boosting knowledge for answer engines. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[12] David Hawking, Nick Craswell, Paul Thistlewaite, and Donna Harman. Results and challenges in Web search evaluation. In *8th World Wide Web Conference*, pages 243–252, Toronto, Canada, May 1999.

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

[14] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi. IBM's statistical question answering system. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[15] Julian Kupiec. MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In *16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 181–190, Pittsburgh, 1993.

[16] John Prager, Eric Brown, Amni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191, Athens, August 2000.

[17] John Prager, Dragomir R Radev, Eric Brown, Amni Coden, and Valerie Samn. The use of predictive annotation for question answering in TREC-8. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

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

[19] Alan F. Smeaton. Information Retrieval: Still butting heads with natural language processing. In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, pages 115–138, Frascati, Italy, July 1997. Reprinted as Springer-Verlag Lecture Notes in Artificial Intelligence 1299.

[20] Rohini Srihari and Wei Li. Information extraction supported question answering. In *8th Text REtrieval Conference*, Gaithersburg, MD, 1999.

[21] Ellen M. Voorhees and Donna Harman, editors. *Proceedings of the Ninth Text REtrieval Conference*, Gaithersburg, MD, 2000. See `trec.nist.goc`.

[22] Ellen M. Voorhees and Dawn Tice. Building a question answering test collection. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207, Athens, August 2000.

[23] W. A. Woods, Stephen Green, Paul Martin, and Ann Houston. Halfway to question answering. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[24] Xiaolan Zhu and Susan Gauch. Incorporating quality metrics in centralized/distributed information retrieval on the WWW. In *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 288–295, Athens, August 2000.