Xerox TREC-8 Question Answering Track Report

David A. Hull

Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan France
hull@xrce.xerox.com

Abstract

This report describes the Xerox work on the TREC-8 Question Answering Track. We linked together a few basic NLP components (a question parser, a sentence boundary identifier, and a proper noun tagger) with a sentence scoring function and an answer presentation function built specifically for the TREC Q&A task. Our system found the correct 50-byte answer (in the top 5 responses) to 45% of the questions, a quite respectable performance, but with considerable room for improvement. Based on the failure analysis presented in this paper, we can conclude that the system would benefit from having access to a broad range of other NLP technologies, including robust parsing and coreference analysis, or some good heuristic approximations thereof. The system also has a clear need for some semantic resources to help with certain difficult problems, such as finding answers that match the semantic class X in *What X?* questions.

1 Introduction

Natural language processing (NLP) techniques have not had a large impact on the information retrieval community in recent years. While NLP techniques are extremely useful for stemming and phrase extraction, linguistic techniques can be easily approximated by adhoc methods which run faster and work about as well, at least for the English language. However, there is growing evidence that this will change as the IR community moves towards methods which analyze document content in more depth.[1] The Question Answering task is an important step in that direction. Question Answering is an interesting challenge for NLP researchers because it is new application for many traditional NLP techniques, such as parsing, coreference analysis, and proper name recognition.

Xerox has a long history of research in information retrieval and natural language processing. The TREC Q&A track gives us a nice opportunity to apply the techniques in our NLP toolbox to a new problem. Unfortunately, the author was only able to devote two weeks of time to the TREC-8 task, so expectations were necessarily modest. Our primary goal this year was to test our proper name tagger, called ThingFinder, for this task. Given these constraints, we were quite happy with our TREC-8 performance. The next section introduces the Xerox TREC-8 question answering system. This is followed by an analysis of the TREC-8 results and a commentary on the future prospects of the system.

[1]See: http://www.infotoday.com/searcher/jan00/feldman.htm for a nice survey of future prospect in information retrieval.

2 Xerox Question Answering System

The Xerox TREC-8 question answering system consists of the following basic components:

(1) A question parser

(2) A sentence boundary identifier

(3) A sentence scoring function

(4) A proper noun tagger

(5) An answer presentation function

The Xerox Q&A system operates on a set of documents retrieved by an independent information retrieval system. For convenience, we worked with the top-ranked document set generated by AT&T's TREC-7 adhoc system and provided as a service by Amit Singhal to Question Answering Track participants. Initially, the question is parsed and typed according to the semantic category of the anticipated answer. The documents are divided into sentences, and each sentence is scored by computing its similarity to the query. The top scoring sentences are then passed to a proper noun tagger, and the tagged elements are compared to the question type. All sentences which do not contain an element that matches the question type are removed. The answer presentation function processes the remaining sentences and generates the final result to be presented to the user.

who	<Person>	where	<Place>	what	<What>
whose	<Person>	when	<Time>	which	<What>
whom	<Person>	how	<How>	why	<Unknown>

Table 1: Mapping from keyword to answer category

2.1 The Question Parser

The question parser attempts to identify the question *type* and any secondary arguments which may be associated with that *type*. We define the question *type* as the general semantic class of the expected answer and attempt to identify the following types:

```
<Person>, <Place>, <Time>, <Money>, <Number>,
<Quantity>, <Name>, <How>, <What>, <Unknown>
```

The latter four types correspond to an incomplete resolution of the question category. There are two steps to this process. First, we search for the basic keyword which defines the question type, using the mapping shown in Table 1. The <How>, <What>, and <Number> question types may have an associated secondary argument. The secondary arguments are used to further specify the question type. The query is tagged for part of speech and the secondary arguments are extracted using regular expressions defined over sequences of part of speech tags. For example, we identify the sequence: <How> *Adj* (*Adj* = Adjective). If *Adj* is *long* or *short*, the question type is <Quantity>. If *Adj* is *rich* or *poor*, the question type is <Money>. If the secondary argument is not found or not matched, the question type remains <How>. We follow the same pattern for <What> questions. For example, "<What> cost" and "<What> is the cost" both get mapped to <Money>. In hindsight, we realized that who/whom/whose questions should cover both people and organizations (e.g. Q169: *Whom did the Chicago Bulls beat in the 1993 championship?* is clearly looking for a basketball team.).

If the question does not have one of the basic keywords, it is usually mapped to <Unknown>. However, certain secondary arguments will define the question type without such a keyword. For example, a request such as: *Find the price of a Jaguar XK8*, will be correctly typed as <MONEY> due to the presence of the word *price*. This allows the system to handle some non-standard question formats. The secondary argument list was created by extracting a few thousand questions from a large corpus and by using our general background knowledge of the English language, and consists of 38 elements. The list was constructed in an afternoon and could certainly be enriched. The current system has no knowledge base and no tools for semantic analysis, so much of the extracted secondary information cannot be used. For example, for Q118: *What two researchers . . .*, the system can recognize that it should be looking for researchers, but doesn't know that a researcher is a <Person>. There is clearly a need for some semantic resources in the system.

2.2 Identifying Sentence Boundaries

Sentence boundaries are recognized by a tokenizer written in *awk*. A word token consists of a string of characters delimited by spaces. A word token which ends in a separator character ("?", ")", etc.) defines a sentence boundary. The "." character also defines a sentence boundary unless the word token appears on a list of 206 common abbreviations or satisfies the following *awk* regular expression:

```
/^([A-Za-z]\.([A-Za-z]\.)+|[A-Z]\.|[A-Z][bcdfghj-np-tvxz]+\.)$/
```

The tokenizing routine is applied to each of the top ranked documents to divide it into "sentences". For more information on our approach to word and sentence recognition, see Grefenstette [1].

2.3 Sentence Scoring

Each sentence is scored according to the number of words it has in common with the query, using the following weighting function: proper noun = 1.0, number = 0.8, common noun or unknown word = 0.4, other content word = 0.1. The raw sentence score is the weighted sum of the number of unique query words it contains. Each word is counted only once, so a sentence that has all the content words contained in the query receives the maximum possible score. The question keyword (e.g. who, what, etc.) and function words, as identified by the part of speech tagger, are ignored. The sentence score is normalized by dividing by the maximum observed score. Similarly, the score of the document which contained the sentence is normalized by dividing by the score of the top ranked document. The top ranked document data provided by Amit Singhal for use by the track contains the score of each document.

The final sentence score is:

$$S(s,d) = 0.8 * S_n(s) + 0.2 * S_n(d)$$

where $S_n(s)$ is the normalized sentence score and $S_n(d)$ is the normalized document score. Essentially, the document score is used to break ties between sentences containing an equivalent number of query terms (of which there are often very many). Position is used as a second tie-breaking criterion for sentences with the same score from the same document. Sentences are ranked in the same order they appear in the document. The TREC corpus consists almost entirely of newspaper and newswire texts. News articles tend to be written with the most important information first. In addition, there are likely to be more proper nouns and fewer references in the early part of the text. We realized in hindsight that the sentence is perhaps too small a unit for matching question words and extracting answers. Since we don't currently have any automatic anaphor or ellipsis resolution technology, it would probably be better to analyze longer passages.

2.4 Proper Noun Tagging

ThingFinder is a proper name tagger developed at Xerox by François Trouilleux. It identifies nine different types of proper names, five of which correspond to semantic classes used by our question answering system: person names = <Person>, location names = <Place>, date expressions = <Time>, monetary expressions = <Money>, and other proper names = <Name>. The other four, not currently used by our question answering system, are: percentages, organization names, events, and legal citations (contracts, treaties, etc.). Trouilleux [2] provides a detailed report on the architecture and specifications of ThingFinder.

The remaining question types: <Number> and <Quantity> are extracted based on regular expressions applied to sequences of part of speech tags. The <Number> tag is simply a number, and can have a secondary argument which identifies the item being enumerated. A <Quantity> is a number followed by a one or more words giving the unit of measure. For example, the question *How many people . . . ?*, will be parsed to the pair (<Number>, people). A <Quantity> describes a distance, length, volume, etc, where the exact unit of measurement is not specified. The <Name> type matches any proper name or unknown word (i.e. a word which does not appear in our lexicon).

The question types <How>, <What>, and <Unknown> are used for incomplete question parses. Depending on the secondary arguments, acceptable answers will include noun phrases, verb phrases, and/or unknown words. Sentences are analyzed by ThingFinder in decreasing order of their score. Only sentences with elements which match the question type are retrained. All other sentences are filtered out. This process continues until five matching sentences have been found or the ranked list of sentences has been exhausted.

2.5 Answer Extraction

The Q&A track has two participation categories: under 50 bytes (= characters) and under 250 bytes. Xerox participated in both categories and defined a unique answer extraction routine for each category. The default presentation for the 50 byte category is the proper noun or character string which was tagged to match the question type. If more than one string was found in a given sentence, the strings are concatenated to create a multi-part answer. If the concatenated string is longer than 50 bytes, the answers are split up again, and some may be passed on to the next lower rank position. Sentences are processed from the top until a set of 5 answer strings has been generated. Sometimes this requires less than five sentences if one sentence contains a lot of possible answers.

For example, for (Q3): *What does the Peugot company manufacture?*, the second ranked sentence was:

> Mr Longuet said Peugeot's decision to stop manufacturing bodies for its 504 utility vehicle from the end of November left the fate of the company in Renault's hands. (FT934-4706)

The question was parsed to (Peugot company, manufacture, <What>). In this case <What> matches any proper noun or noun phrase, so the system produces the following set of possible answers:

> Mr Longuet, decision, bodies, 504 utility vehicle, end, November, fate, Renault, hands

The terms *Peugot* and *company* are removed from the list because they are part of the question. Clearly the concatenation of these strings is longer than 50 bytes, so the answer needs to be pruned. In such cases, priority is given to proper nouns and multi-word noun phrases, leading to a final answer of:

Mr Longuet/504 utility vehicle/November/Renault

Given more time, we would have filtering out proper nouns with tags which are less likely to match the question type. In this way, we might have eliminated *Mr Longuet* as a <Person> and *November* as a unit of <Time>.

We eventually plan to link answer selection to a robust parse of the target language sentence, in order to reduce the set of possible answers. We could then search for sentences with the relation SUBJ(Peugot company, manufacture) and extract <X> from a relation DOBJ(manufacture, <X>). Parsing might not always find the exact response (likely answer here: *bodies*), but it would narrow search region so that the correct response would fall in the 50-byte window. Alternatively, we could rank answers according to their proximity to the question terms. A 50-byte answer window is sufficiently generous that parsing may not perform better than more approximate methods. However, parsing alone is unlikely to provide sufficient recall to substantially improve the system. The system will also need to be capable of recognizing noun phrase variants and semantically related verbs. For this example, the response *504 utility vehicle* is correct, though perhaps too specific for the question being asked.

We recognize that squeezing multiple answers into the same text string violates the spirit of the Q&A track, since the goal is to find the single right answer to the question. However, we would like to point out that while the system often lacks the world knowledge to make the correct decision, the same task can be easy for the person who asked the question. A human reader would immediately recognize that *504 utility vehicle* is the only plausible correct answer in the above example. To give another example, consider question (Q25): *Who was the lead actress in the movie "Sleepless in Seattle"?*. The first answer returned by our system is:

Tom Hanks/Meg Ryan/John Grisham (FT933-16459)

An English native speaker knows that Meg Ryan is the only female name in the list, and thus the only possible correct response. The assessors confirmed this supposition by marking this answer correct. The alternative solution, coding substantial semantic knowledge into the system, is a very expensive proposition. However, we do hope to gather some semantic information for future versions of the system, using resources such as WordNet or thesauri automatically generated from the corpus.

We made the decision to return multiple responses in a single text string without much knowledge of the assessment process. The assessors were given the following instructions (excerpt[2]):

- If the answer string contains the answer plus misc. other stuff, judge YES.
- If the answer string contains the answer plus other text that interferes with recognizing the answer, probably judge NO.

This means that the assessors had a lot of flexiblility in judging multiple responses. We believe these instructions are an appropriate and reasonable simulation of user reactions. A partial analysis of the results suggests that we did not benefit from our decision to return multiple answers. In fact, it probably hurt our performance. We only returned multiple answers when they were in the

[2]TREC-8 Question Answering Track Evaluation - presentation by Ellen Voorhess at the TREC-8 Conference

same sentence. As mentioned previously, the 50-byte limit is sufficiently generous that we would probably have been better off simply picking the most appropriate sub-string from the sentence.

For the 250 byte category, the default presentation is the sentence containing an answer which matches the question type. If the sentence is too long, it must be truncated. The simplest way to do this is to present a 250 byte text string centered on the answer. However, it is quite possible that important context could be lost in this fashion. In addition, since many sentences contained more than one possible correct answer, it was often impossible to find a single consecutive 250-character string which contained all the answers. We chose instead to summarize the sentence by successively deleting the least important words. Words were ranked from least to most important as follows:

(1) All function words

(2) Adverbs, adjectives, verbs, common nouns ($<$ 5 characters)

(3) Adverbs, adjectives, verbs, common nouns ($\geq$ 5 characters)

(4) Multi-word noun phrases and proper nouns

(5) Distance from "answers"

Fortunately, the system rarely had to procede further than item (2) in this priority list.

For example, for question (Q116): *Which team won the Super Bowl in 1968?*, the system returned:

> With Namath as their leader, the AFL's 1968 New York Jets went into Super Bowl III as an 18-point underdog and won, 16-7, against the NFL champion Baltimore Colts, who, 13-1 that season, had romped past such NFL powers as the Chicago Bears, Vikings, 49ers, Giants and Rams. (LA071790-0057)

which becomes:

> Namath _ _ leader _ _ AFL's _ New York Jets _ _ Super Bowl III _ _ 18-point underdog _ _ _ 16-7, against _ NFL champion Baltimore Colts _ _ _ _ _ season _ _ romped _ _ NFL powers _ _ Chicago Bears _ Vikings _ 49ers _ Giants _ Rams

This approach allowed us to keep all the possible correct responses in the answer string in a relatively readable format. However, we did delete two important words (*1968* and *won*) because they had fewer than five characters. Oops! In hindsight, we realized that the sentence summary algorithm should also retain all words which are part of the question. Our assumption throughout is that users of the system will always have the option to view the corresponding full text with the selected answer highlighted. Therefore, the Q&A system should try to maximize the probability that the answer appears in the summary, rather than trying to maximize the chance that the user can verify that the answer is correct simply by looking at the summary.

3 TREC-8 Results and Failure Analysis

Xerox submitted one run in the 50-byte category and one run in the 250-byte category. The results are presented in Table 2. The columns of the table are: average reciprocal rank (1/rank if answer in top 5, 0 otherwise), number/percent of questions with a correct response in the top 5, and the average rank of the Xerox system compared to all other participants (based on average reciprocal rank). In other words, Xerox's 50-byte run averaged between 8th and 9th place out of a total of 20

Category	Avg. 1/Rank	#/% Correct	Avg. Rank
50-byte	0.317	89 / 45%	8.43 / 20
250-byte	0.453	117 / 59%	9.96 / 25

Table 2: Xerox TREC-8 Q&A Track results summary

runs. Our system is managing to answer about half the questions correctly, a result which puts us slightly above average relative to the other TREC-8 Q&A track participants.

Table 3 breaks down the results by question type. *Who* and *what - location* questions are easiest overall, although Xerox also does well on *where* and *what - name* questions. Not surprisingly, *what - location* questions are much easier than *where* questions because the type of location is specified. However, Xerox does about the same on both categories, because the version of ThingFinder used in these experiments has only one tag for location. In other words, it doesn't distinguish between cities and countries. We have surprising success with *what - name* questions relative to the other participants. We attribute this to the fact that no such questions appear in the training set. Due to lack of time, we ignored the training set of questions entirely. While this strategy is not recommended in general, it may have inadvertently helped us write a more general question parser. This may explain our relative success with the *what - money* and *other* questions as well.

Question	Num	XRCE			All Systems		
Type	Q's	#C	%C	ARR	#C	%C	ARR
who	48	23	0.48	0.35	18.0	0.38	0.29
where	21	**12**	**0.57**	**0.40**	7.1	0.34	0.23
when	18	7	0.39	0.31	5.6	0.31	0.21
how - number	19	7	0.37	0.37	6.3	0.33	0.26
how - measure	8	2	0.25	0.07	2.0	0.24	**0.16**
how - money	4	1	0.25	0.25	0.8	0.21	0.18
what - person	8	3	0.38	0.31	2.2	0.28	0.23
what - time	5	2	0.40	0.20	1.4	0.28	0.21
what - location	16	9	0.56	0.47	7.5	0.47	0.34
what - number	2	0	0.00	0.00	0.3	0.15	0.13
what - money	2	2	**1.00**	**0.50**	0.6	0.28	0.19
what - name	13	**8**	**0.62**	**0.41**	4.0	0.31	0.23
what (is) X	23	7	0.30	0.18	5.7	0.25	0.18
other	11	4	**0.36**	0.16	2.2	0.20	0.14
Total	198	87	0.44	0.32	63.8	0.32	0.24

Table 3: Xerox TREC-8 Q&A Track: question breakdown

We performed a failure analysis on the first 100 questions to learn how the system could be improved, focusing on the 50-byte task. This process consisted of reading the question, the top-ranked sentences, the answers returned by our system, and the correct answers. Based on this information, we tried to determine how our system could be improved or what new technologies would be needed to answer the question correctly. The results of the failure analysis will help us prioritize future improvements to the system. Table 4 summarizes the results for the 64 questions for which the system could give a better answer.

The most frequent problem was with our sentence ranking algorithm, with the usual result being that none of the top-ranked sentences contained a correct answer. Our sentence scoring

Problem/solution	freq.	Problem/solution	freq.
sentence ranking	11	tokenization/normalization	5
coreference	10	question word overlap	4
sentence parsing	9	semantic information	3
multiple answers	9	correct answer	2
no clear solution	8	question parsing	1
proper noun tagger	6		

Table 4: Failure analysis of first 100 questions

algorithm assigns proper nouns a high weight and verbs a low weight. While verbs are often less important than nouns in information retrieval, this is much less true in question answering, so the latter decision turned out to be a mistake. A verb match is often a strong indicator of relevance. Other than that, it is often difficult to tell what went wrong. In some cases, it may be that the IR system did not return the necessary documents in its top 200. We did not test this possibility.

Another frequent problem is references. For example, for the question (Q34): *Where is the actress, Marion Davies, buried?*, the following passage contains the answer:

> Actress Marion Davies, mistress of William Randolph Hearst, has been dead for almost 30 years, but someone remembers. Earlier this week, a fan left a red rose on her mausoleum in Hollywood Memorial Park.

The system would have had a much better chance of finding the answer if it knew that *her* refers to Marion Davies. Of course, extending the passage length of the answer to 2-3 sentences represents a cheaper solution to this problem. This is an interesting issue to explore in future experiments.

For nine questions, the system could probably find the correct answer with an accurate parse of the answer sentence. For example:

> The price collapse has been particularly painful to Grenada for which nutmeg is the main commodity export.

The main commodity export of Grenada is *nutmeg* not *price collapse*, as we unfortunately suggested[3]. This does not necessarily mean that our robust parser would always find the correct solution. Many TREC-8 participants approximate parsing with a proximity model, and this might well be equally effective. In addition, there were nine questions where the system returned multiple answers, including the correct one, but received no credit from the judges. A good parser would have found the single correct answer in most of these cases. The issue of sentence analysis has an impact on 18 questions in the first 100, making it the single most important issue to address in future work on the system.

There are eight questions which are sufficiently difficult that it is unclear how we could answer them correctly given the current limitations of the technology. There are six questions where our proper name tagger either made an error or does not have a sufficiently detailed tagset to find the correct anaswer. There are five questions where we had problems with word tokenization or normalization (stemming). For example, we find no overlap between *Winter Olympics* and *Olympic Games*, because *Olympics* is not lemmatized in the former case.

The system failed on four questions because of an implementation error. We made the obvious decision to ignore proper names and common nouns in the answer which also appear in the question. However, this was implemented in such a way that we also ignored answers which partially

[3]If the system had a thesaurus which identified nutmeg as a commodity, it could find the correct answer without parsing.

overlapped question words. This means that for question (Q51): *Who is the president of Stanford University?*, we filtered out *Stanford University President Donald Kennedy*. It is a not necessarily trivial to implement this correctly, because many proper noun variants which overlap the question should be filtered out. For example, if the question had been, *Who is the president of Stanford?*, the phrase, *the Stanford University President*, would not be a correct response. Most likely, one would want to accept responses where the question words serve as modifiers and not as the head noun.

There are three questions where the system needed semantic information to find the correct response. For example, for question (Q85): *Which former Ku Klux Klan member won an elected office in the U.S.?*, it helps a lot to know that a *Ku Klux Klan member* is a person. For two questions, we disagreed with the assessment, and there was one case where we need to improve the question parser. Looking at the results as a whole, it is clear that there is still a lot of work to do, which comes as no surprise.

4 Commentary

From the previous section, we can conclude that our system would benefit from having a lot more NLP technology. This will be our focus in the immediate future. It remains to be seen whether NLP techniques such as parsing and coreference analysis can be approximated by simpler, more efficient heuristics for the Q&A task. In general, we need to establish an optimized answer scoring function which weighs the plausibility of each answer by drawing on many different sources of information. We did not have time to do this for TREC-8, and it is unclear whether the training set of questions we were given was large enough to do this effectively anyway. With the 198 new questions provided in TREC-8, it should definitely be possible to train a good answer ranking function for TREC-9. Many TREC-8 participants have already developed large, complex scoring functions. Our current unweighted scoring model is clearly insufficient, since it does not try to measure the relative correctness of each of many possible answers within the same sentence.

At first glance, it is somewhat astonishing that all TREC-8 Q&A track participants use pretty much the same techniques. This has almost never been the case in the past for a first-year track. This means that there is a strong degree of consensus as to how the TREC Q&A task should be addressed given the current technology. However, we must recognize that the TREC task is a corpus-based information extraction problem, which is only a special subtask of the more general question answering problem. In this context, traditional AI solutions which rely on large knowledge bases are much less appropriate. It is interesting to ask how such systems would perform on the TREC-8 task. The answer is most likely, quite badly. This is not necessarily a problem with their technology. Rather, the knowledge bases of these systems are not constructed to answer questions appropriate to the time period covered by the TREC corpus. As TREC consists primarily of news articles, much of the information has a short lifespan. In addition, most questions were constructed directly from the documents, giving corpus-based extraction techniques a huge advantage. In general, our perspective is that corpus-based and knowledge-based techniques are highly complementary. The former techniques are recall-oriented and the latter techniques are precision-oriented. Future open-domain Q&A systems will need to incorporate both kinds of technology in order to be successful. We can already see this pattern emerging in the evolving market for Web search engine and directory service technology.

References

[1] G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *Proc. of the 3rd Conference of Computational Lexicography and Text Research (COMPLEX '94)*, 1994.

[2] Francois Trouilleux. Thingfinder prototype english version 2.0. Technical report, Xerox Research Centre Europe - Grenoble, April 1998.