Question Answering from Large Document Collections

Eric Breck, John Burger, David House, Marc Light, Inderjeet Mani

The MITRE Corporation
{ebreck, john, dhouse, light, imani}@mitre.org

Abstract

We present a question answering system with a hybrid design, combining techniques from knowledge representation, information retrieval, and natural language processing. This combination enables domain independence and robustness in the face of text variability, both in the question and in the raw text documents used as knowledge sources. We describe the specific design of our current prototype, which has been entered in the 1999 Text Retrieval Conference's Question Answering track. We also comment on the evaluation of question answering systems.

Introduction

This paper describes an end-to-end question answering system called Qanda.[1] It is a robust, domain independent system that takes questions expressed in English and attempts to provide short, concise answers. The system's primary source of knowledge is collections of text, e.g., newswire texts, IRS forms and manuals, web page collections. By *short, concise answers*, we mean a single noun phrase or sentence. By *domain independent*, we mean the system is intended to answer questions about any domain discussed in the text collection being used. Finally, *robust* means the system attempts to return a reasonable answer regardless of the phrasing of the question or the form of relevant passages in the text collection.

IR techniques use the content words in the question to find the answers, namely those answers that co-occur with the content words. This string-level focus works well when questions are verbose and answers need only consist of relevant documents. However, when questions and answers are short, there are fewer content words, increasing the chance of skipping over the correct answer passage because of a slightly different choice of words.

KR techniques focus on concepts and inferential links between them. Both concepts and inferential links must be stored in a precompiled knowledge base and must be expressed in the knowledge representation language on which the inferential machinery operates. The need for a precompiled knowledge base and the lack of automated knowledge base construction make KR techniques domain dependent. In addition, current inferential techniques are not able to handle arbitrary phrasing of questions.

[1] Qanda is pronounced *kwanda*.

A central idea of our system is to combine KR's finer-grained representation and inference abilities with IR's robustness and domain independence to produce a system with the desired characteristics. NLP technology is used liberally throughout. The idea of combining such technologies is not new; both (Kupiec 1993) and (Hirschman et al. 1999) describe hybrid systems. However, the system described here offers a novel combination of technologies.

In the following sections, we will describe in more detail our system's use of IR, KR, and NLP. Then we will describe the actual implementation. We follow this with some general comments on evaluation of such systems and particular comments on the 1999 TREC question and answering track, in which our system participated. Finally we make some concluding remarks and discuss future work.

Our Approach: Mixing IR and KR via NLP

Let us start with an example of how such a hybrid system might work. Consider the question *What Arab country invaded Kuwait during the Bush administration?* NLP can be used to determine that the answer must be a country and that it should be a participant in an invasion event. A KR representation of this question might include:

$$\text{country}(x) \wedge \text{agent}(x, e) \wedge \text{invasion}(e)$$

IR techniques can then be used to find relevant documents using the terms *Arab*, *country*, *invade*, *Kuwait*, *during*, *Bush*, and *administration*. Next, NLP might be used to process these documents and populate a knowledge base with countries, locations, invasion events and participants. Then, simple KR inference techniques can be used to search for a binding of x that satisfies the KR representation of the question. Finally, satisfactory bindings (i.e., answers) are ranked using IR techniques, applied to the contexts in which the answers occurred.

What we use from IR

From one perspective, the question answering task described above consists of retrieving text passages (answers) relevant to a textual description of need (questions). Stated in this way, information retrieval would seem to be admirably suited to the task—IR is all about retrieving relevant texts. However, the typical IR passage is much longer than most answers, which might be viewed as "micro-passages". In addition, the standard IR definition of "relevance" is arguably much too weak for question answering. Nonetheless, we believe that there are important IR techniques that can be applied to the question answering task.

In most IR approaches, the objects of interest are *terms*, typically words or stems. Artifacts such as our questions, documents, and answers are treated as simple "bags" of terms, with little or no structure. These are compared with a variety of relevance or similarity metrics that have been developed over the past forty years of IR research. We can leverage this view of question answering to do two things:

- Focus on a small subset of the document collection
- Select among candidate answers

The first can be accomplished by computing the relevance of every document in the collection with respect to the question. This is exactly what IR systems are optimized to do. So, after indexing our document collection with an IR system, we can posit queries like the example above:

{*Arab, country, invade, Kuwait, during, Bush, administration*}

The IR system will rank the documents in the collection, using some measure of relevance. Then the question answering system can process these documents in order, possibly ignoring all but the most highly relevant documents.

The second use to be made of IR is to select among candidate answers. (The set of candidates can be derived as described in the next section.) This can be done by effectively treating each of the candidate answers or, more effectively, each of the answers' *contexts* (e.g., matrix sentences or paragraphs) as small text passages, and scoring them by their relevance to the original question. This is analogous to the previous process, although the relevance or similarity metrics used for sentences may be different from those used for whole documents.

The representation of questions, documents and answers, engendered by an IR approach is quite simplistic. However, it is complementary to the KR component described in the next section, due to its speed and robustness. Even for oddly formed questions and documents, the bag-of-words approach will usually find some relevant passages. Sometimes the question and an answer-containing document use different vocabulary to describe the same concepts. But well-established IR techniques such as relevance feedback (Rocchio 1971) can be used to ameliorate such situations.

What we use from KR

One may characterize a KR view of question-answering as finding variable bindings that satisfy a logical representation of a natural language question. The objects of interest for KR are *concepts*. These differ from IR terms in that they are more abstract, i.e., are not tied to any particular position in text. We use KR techniques to provide a representation for questions and the concepts discussed in the documents. The following example illustrates the various processing steps in which KR is used.

Text question:
What Arab country invaded Kuwait during the Bush administration?

Corresponding question representation:
$\exists x,e$ answer(x) $\wedge$ country(x) $\wedge$ agent(x, e) $\wedge$ invasion(e)

Inference via type hierarchy that x is also a location:
$\exists x,e$ answer(x) $\wedge$ country(x) $\wedge$ agent(x, e) $\wedge$ invasion(e) $\wedge$ location(x)

Sample document passages:
(a) *During the Carter administration, Indonesi invaded East Timor.*
(b) *… Bush and Iraq. When it invaded Kuwait …*
(c) *But Grant's forces invaded Virginia …*

Corresponding document representations:
(a) country(Indonesia) $\wedge$ country(EastTimor) $\wedge$ agent(Indonesia, e) $\wedge$ invasion(e)
(b) country(Iraq) $\wedge$ country(Kuwait) $\wedge$ agent(Iraq, e) $\wedge$ invasion(e)
(c) location(Virginia) $\wedge$ invasion(e)

Matching question and document representations, producing bindings for answer variable x:
(a) *Indonesia*
(b) *Iraq*
(c) *Virginia*

IR used to rank alternatives
(1) *Iraq*
(2) *Indonesia*
(3) *Virginia*

Some things to note: Reference resolution is used to unify different expression of the same entity, such as *Iraq* and *it* in (b). Also, the matching process allows some predicates to go unsatisfied, such as agent(x, e) in (c). There is also a preference for matching predicates explicitly from the question, as opposed to those introduced through type inference. Thus satisfying country(x) is better than merely satisfying location(x), making both *Iraq* and *Indonesia* preferable to *Virginia*. Finally, the IR techniques discussed in the previous section rank *Iraq* above *Indonesia*, due to the higher overlap with terms in the question.

What we use from NLP

Natural language processing is used because the input and output of the system take the form of linguistic artifacts—questions, documents, and answers. It seems clear that to robustly handle these, NLP techniques will be required.

Questions in natural language can be far freer than in any query language, and so we use NLP techniques to create several parallel representations of the question in more manageable forms. We create an IR query (a set of terms), a KR query (a set of propositions) and also a question-typing query to our system. The latter will eventually allow us to perform distinct processing steps for yes-no questions, procedural questions (*How can I do x*) and the phrasal-answer questions that we attack with the present system.

The documents in our collection are analyzed with NLP to find entities of interest and propositions that are true of them. For example, proper names of persons, organizations, and locations, are identified. These entities and propositions are entered into a knowledge base that can be used by the rest of the system. NLP is also used to find referential links between the entities, unifying them into coreference classes in the knowledge base.

The answer that results from our system also needs to be comprehensible. So we use NLP techniques to synthesize an appropriate answer from the output generated by the system.

Another important contribution of NLP will eventually be understanding of the context in which the document, query, and answer occur. Without understanding of the discourse or dialogue structure, the query, the document passages, or the answer might be misinterpreted.

Our Implementation

We have implemented the various phases of processing described above in a system called Qanda. The basic outline of processing in Qanda is as follows:

- Using NLP, the textual question is mapped to an IR query and a KR query.
- The IR query elicits a set of ranked documents from the collection.
- Using NLP, the returned documents are processed to dynamically create a knowledge base relevant to the question.
- The knowledge base is then queried to produce a set of candidate answers, which are then ranked using IR techniques

The question processing begins by performing punctuation and sentence tokenization on the question, after which the question is tagged for part-of-speech. Then, various phrasal entities are identified (our entity hierarchy is discussed below). Based on all of this information, a question-typing module formulates two kinds of queries—one for the IR system, and one for our dynamic knowledge base. The IR query is produced by removing certain parts of speech from the question, such as *wh*-words and punctuation). The knowledge base query is produced by looking for a wide variety of stereotypical question patterns based on lexical and part-of-speech information as well as the findings of the phrasal taggers. Based on these patterns, the query is generated containing a number of predicates that the desired answer must satisfy. To reiterate the example:

Question:
What Arab country invaded Kuwait during the Bush administration?

KR query:
country(x) $\wedge$ agent(x, e) $\wedge$ invasion(e)

IR query:
{*Arab, country, invade, Kuwait, during, Bush, administration*}

The IR system returns a (short) list of candidate documents, each of which is processed in the following ways: Like the question, they are tokenized and tagged for part-of-speech. The documents are then stemmed and function words are removed. The documents are then tagged for whatever predicates are in the knowledge base query. The predicates that are currently recognized are elements of the type hierarchy in Figure 1.[2] MITRE's Alembic NE tagger (Vilain and Day 1996) identifies Person, Location and Organization entities, as well as Dates and other time expressions. The system also uses retaggers that recognize Locations that are Cities or Countries, as well as Persons that are Male or Female, and retags, essentially classifying them further down the type hierarchy. We also use a measure-phrase tagger that classifies phrases such as *50 feet* and *three hours*. (These are often answers to *How tall*, *How much*, etc.). Finally, we have a simple coreference module that performs name coreference and resolves pronouns and definite references.

All of these tagged entities are then entered into our dynamic knowledge base, and grouped into coreference classes. If the knowledge base query contains predicates, the knowledge base is searched for entities that satisfy those predicates. If an exact match cannot be found (e.g., there are no countries in the knowledge base), we try to minimize the number of steps up the hierarchy (e.g., the next best would be a generic location). If no type even of a higher level can be found (in a given document) we fall back to returning sentences ranked solely by the IR techniques described next.

We impose an ordering on the resulting set of entities using one of a variety of standard IR measures, based on the contexts in which the entities occur. We examined simple overlap between the query and response, overlap normalized by response length (penalizing for long answers), as well as Dice score and Cartesian distance. The results reported in the next section were produced by

[2] The shaded predicates in Figure 1 constitute those fully processed by all the system's components. Thus, for example, although some Persons are recognized to be Male or Female, they are, in the end, treated as if they were merely Persons.

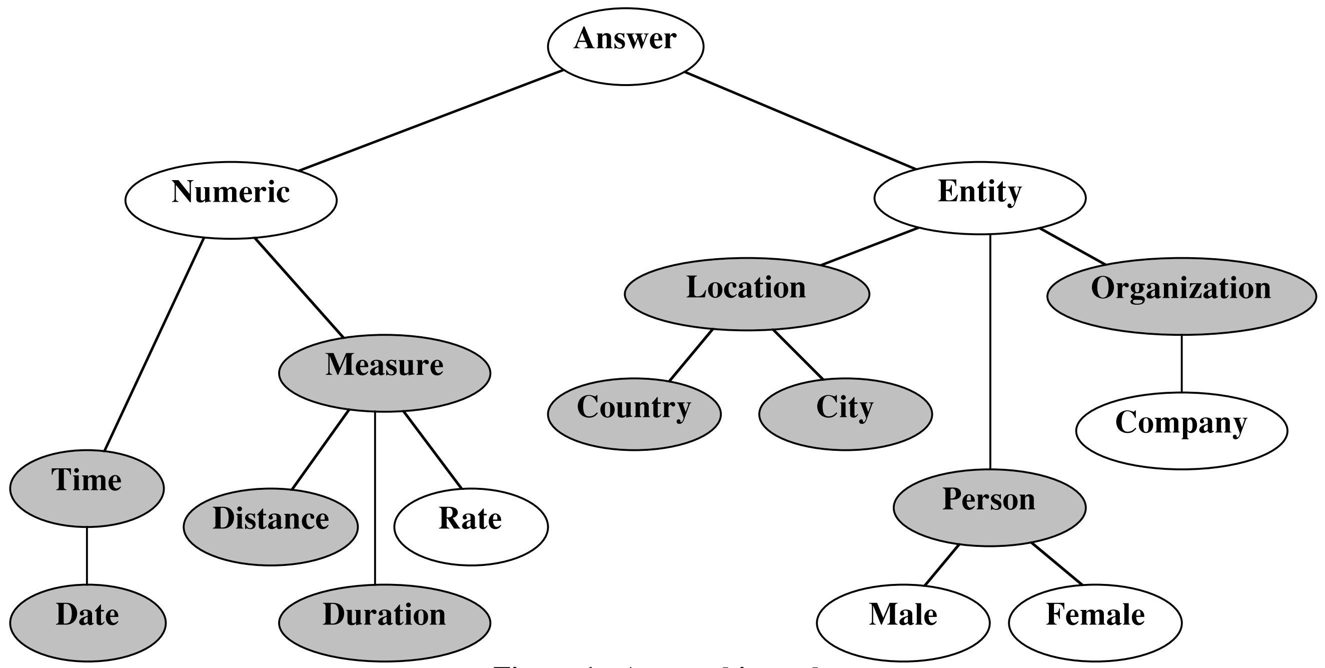

Figure 1: Answer hierarchy

sorting on query/response overlap, but the other measures performed similarly on our development set, and we hope to test on a larger set and determine if it would be worthwhile to use other metrics.

Finally, the answers are synthesized by concatenating the best (e.g., longest) member of a coreference class with the best-scoring sentence in which some member of the class appears. This can help to contextualize an entity that might otherwise be difficult to interpret. An answer such as *Iraq: It invaded Kuwait during the Bush administration* might be generated for the example above.

Evaluation

The evaluation of question-answering systems presents a number of interesting challenges. Both on-line evaluations (where humans score system answers) and off-line evaluations (where a program scores system answers against a "gold standard" reference) provide useful insights into the efficacy of question-answering systems. The nature of the evaluation can depend on a number of different factors:

- The types of questions, e.g., factual versus explanatory, context-dependent or not
- The document genres, e.g., news stories, expository texts from encyclopedias, technical reports
- The requirements for a successful answer, e.g., whether the answer is a document extract or synthesized, whether it is provided with document context

Since design of evaluations can be fairly resource-intensive, we decided to leverage the community-wide question-answering evaluation effort carried out in this year's Text REtrieval Conference's (TREC) Question Answering track (Voorhees and Harman 1999).

The evaluation tests the accuracy of a system in its ability to answer ad hoc questions whose answers are to be found in a large TREC subcollection containing a half-million documents. (Specifically, TREC disks 4 and 5, minus the Congressional Record.) The TREC-8 task is therefore restricted to answer *retrieval*. To help construct a set of test questions, each participating team submitted ten questions, along with their suggested answers (more than one alternative answer may be identified for a question) and a pointer to the document in which the answer was found. The track organizers at the National Institute of Standards and Technology (NIST) selected 200 of these questions to form a test set, using the suggested answers to help check the appropriateness of the questions; the suggested answers were then discarded.

Each participating system, given the 200-question blind test set and the collection of a half-million documents, returned a ranked list of five possible answers for each question, along with a pointer to the document used to derive the answer. Two forms of system answers were explored in the evaluation: a short "snippet" with length less than 50 characters, and a longer sentence-length extract, up to 250 characters.

Given a question and a system's output, the judges at NIST decide if the output contains a valid answer; they read the associated document if they need additional context to help in their decision. Thus, the judges establish ground truth from the systems' output, without necessarily

Correct ranked #1	60
Correct ranked #2	16
Correct ranked #3	14
Correct ranked #4	7
Correct ranked #5	9
No correct answer	94
Mean RAR	0.381

Figure 2: Results on 200 TREC questions

looking at all associated documents. For both short and long answers, each system is scored in terms of the Mean Reciprocal Answer Rank (Mean RAR), i.e., the mean over all questions of the reciprocal of the best-ranked, correct answer. If a correct answer is not found within the five, the score is zero for that question. (Thus, for each question, scores of 1, 1/2, 1/3, 1/4, 1/5, or zero are possible.) At press time, the participating systems' results were being evaluated, and final results are expected in September of 1999.

For purposes of this paper, we conducted a preliminary evaluation on the 200 test questions. An experienced evaluator at MITRE (not one of the system developers or authors) scored the performance of our system on the 200 long-form answers (250 characters), using the official TREC evaluation guidelines. For 50.3 percent of the questions, the correct answer was among the five returned by our system. 30 percent of the time, the correct answer was ranked first. For more detailed results, see Figure 2.

Conclusion

We have described a hybrid approach to question answering that utilizes techniques from IR, KR, and NLP to process a collection of text documents as its primary information source. An actual system has been implemented and initial evaluation seems to be promising. The system provides short concise answers to questions expressed in English and is robust and domain independent.

Future Work

We consider the system just described as an initial prototype. We have a number of ideas about how to improve its performance. The most straightforward is to improve the performance of the component technologies, such as the coreference module. Another direction we want to explore is to enrich the semantic representations we extract from the question and those that we extract from the documents. More specifically, events could be extracted along with their participants and their roles. For example, given the question *Which terrorist group bombed the World Trade Center?* the system would look for a terrorist group that was the agent of a bombing event whose target or patient role is filled by *World Trade Center*. Thus, more sophisticated information extraction techniques would have to be developed and more complex knowledge representation and inference would be needed.

Another avenue is to try to make use of processing done for previous questions. A very simple example of this would be to ask users if they are satisfied with an answer and if so to save away the question and the answer as a pair. When another user asks a question, this new question is matched against the old ones. A more complex way of making use of processing of previous questions is to construct a persistent knowledge base and perhaps to note fruitful inferential paths.

It would also be interesting to make use of existing knowledge bases such as WordNet (Miller 1990) or CYC (Lenat 1995).

Another important dimension of question answering lies in the nature of the answers. We would like to move well beyond simple answer retrieval into full-blown answer synthesis, where the answer may be constructed from multiple passages in one or more documents (Mani and Bloedorn 1999, Mani et al. 1999). Here we expect to identify methods to arrive at succinct yet complete and self-contained answers.

Finally, it seems likely that in some domains, users will have a sequence of questions they would like to have answered and that these answers will depend on the preceding questions and answers. Thus, a specific form of discourse processing might be necessary.

Acknowledgements

We would like to thank Lisa Ferro for producing the evaluation results above, as well as John Aberdeen and Warren Greiff for their comments. We are also grateful to MITRE's internal research program for funding this work.

References

Kupiec, Julian 1993. MURAX: A Robust Linguistic Approach for Question Answering Using an On-Line Encyclopedia. In *Proceedings of the 16th Intl. ACM SIGIR Conf on Research and Development in Information Retrieval (SIGIR-93)*. 181–190. Pittsburgh, Penna.

Hirschman, Lynette, Marc Light, Eric Breck, John D. Burger 1999. Deep Read: A Reading Comprehension System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 325–332. College Park, Maryland.

Lenat, Douglas B. 1995. Artificial Intelligence. *Scientific American*, September:80–82.

Mani, Inderjeet and Eric Bloedorn 1999. Summarizing Similarities and Differences Among Related Documents. *Information Retrieval,* 1:35–67.

Mani, Inderjeet, Barbara Gates and Eric Bloedorn 1999. Improving Summaries by Revising Them. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 558–565. College Park, Maryland.

Miller, George 1990. WordNet: An On-line lexical database. *International Journal of Lexicography*, 3:4.

Rocchio, J.J. 1971. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System*, 313–323. Englewood Cliffs, N.J.: Prentice Hall, Inc.

Vilain, Marc and David Day 1996. Finite-State Parsing by Rule Sequences. *International Conference on Computational Linguistics (COLING-96).* Copenhagen, Denmark. The International Committee on Computational Linguistics.

Voorhees, Ellen M. and Donna K. Harman 1999. *The Eighth Text Retrieval Conference*. NIST Special Publications. Forthcoming.