Is question answering a rational task?

Karen Spärck Jones
Computer Laboratory, University of Cambridge
William Gates Building, JJ Thomson Avenue, Cambridge CB3 0FD, UK
sparckjones@cl.cam.ac.uk

Questions and answers: theoretical and applied perspectives
Proceedings, CoLogNET/ElsNET Symposium, Amsterdam, December 2003

Abstract

This note discusses issues in the treatment of question answering (QA) as an NLP task, considering historical influences on its interpretation and evaluation, and the inherent problems that these exacerbate. The discussion suggests that a rather different approach may be needed for systems with real, and wider, utility.

Introduction

Text retrieval has always been the poor relation in NLP (for bad reasons as well as good). Question answering (QA) looks much more sexy. But is QA a real, i.e. autonomous, task? Is it a generic task, something that we all need whole *systems* to be able to do for us, or at least something for particular classes of users (aka analysts?); or should it be more properly approached as a necessary subfunction in a wider-ranging information management system, handled as part of the routine interpretation of user input and not requiring a special 'QA' module?

There is currently a surge of interest in QA, but it figured in early NLP research. There are important questions to be asked about the assumptions and goals of current QA work. This has arisen out of research on retrieval, but earlier work on QA has in my view had an indirect but significant influence on it. At the same time, there are valuable lessons to be learnt from this past work. I will therefore approach what I see as the key questions about QA from a historical point of view.

Historical antecedents

Question answering

As just noted, QA was an early interest in NLP research: Simmons review (1965) lists a surprisingly wide range of studies. These included work on natural language query from databases, database creation and query from natural language input, and natural language query from an existing text base (Simmons et al. 1964). Most of the studies were database oriented and are well illustrated by Raphael's SIR (1968): this was designed to extract data about object relations (e.g. part-of) from input sentences, integrate this into a database, and reply to relational questions, which might require inference (e.g. over transitive relations). Simmons' rather different version of the task involved retrieving relevant text segments, parsing these and comparing the resulting structure with that for

the parsed question. All of these early trials were very limited linguistically and, in the database cases, in their domains and data.

Being able to answer questions was one of the justifications for such general approaches to NLP as that represented by the use of scripts, but this was not scripts' raison d'etre. QA was approached much more directly, as an NLP task in its own right, through building NL front ends to databases. It had the merits of appearing to be a tractable, but not trivial, application for NLP, and a potentially useful one to boot (it certainly seemed a better bet then than MT), with a further advantage as a vehicle for investigating computational theories of language processing and meaning representation. All this is evident in Woods et al.'s LUNAR system (Woods 1978), which was a significant advance on earlier work and a tour de force in its own right. The system was a front end to an independent database, with a potentially real user community. It was thus well engineered as well as linguistically sophisticated, with a clear architecture and logical meaning representation language to connect input and data. But it was able to rely on very direct and close access to the database, itself a single table, on a narrow domain, and on informed users.

Even so, and indeed because of their language and data limits, database front ends appeared a very attractive NLP application area. But unexpected problems appeared, especially when attempts were made to go commercial. One was question 'presumptions': if I ask how many students took CS101 in 1979, it is not enough when the database entry is blank to reply none; it is desirable to address the presumption that the course was given and reply accordingly. Giving appropriate responses, rather than just answers, requires more careful question analysis. It also naturally leads to the use of a domain model, more encompassing than, and abstracted from, the databases's data model. However a domain model turned out to be needed for other reasons as well.

One was front ending to distributed databases and, more generally, making front ends portable (e.g. as with Grosz et al.'s TEAM, 1987). Another was coping with the consequences of distancing users, with their natural ways of using their natural language, from the actual database: systems had to be robust to deal appropriately with very variably expressed questions and with misconceived ones, both implying a rather comprehensive system view of the larger domain. These real world challenges showed that the apparently easy task of building effective natural language front ends was far from easy, and since the late 1980s database access needs have been better served by exploiting modern interface technology, query by example and so forth.

In the meantime, the focus shifted to dialogue systems, whether essentially ones designed to support inquiry or ones requiring an ability to handle questions among other input forms. These were seen as appropriate for more complex, 'negotiatory' tasks, as early illustrated by Bobrow et al's GUS system for airline trips (1977). Dialogue systems had to provide cooperative responses, and thus led to attempts to understand the rationales for questions and their 'real content' by modelling the user, whether in terms of objective user properties, like technical expertise, or subjective ones, i.e. beliefs, desires and intentions. Questions could not always be taken for what they seemed. Other models were also needed for these extended dialogue systems, e.g. discourse models, and manipulating the various models became very complicated (CL-88 1988). There were particular problems about making, and acting on, inferences about the user's intensional state using the very weak information supplied by a few brief inputs. As with the simple database query case, there has been something of a pullback from natural language dialogue systems, except where patently desirable, as with speech; and here systems have been most successful with very limited domains, a large element of system direction, and simple pattern-based conversation models (De Mori 1998).

Information retrieval

Document, or text, retrieval - conventionally though misleadingly labelled 'information retrieval' (IR) - has been the other source for current work on QA. Retrieval is clearly, in the widest sense, an inquiry task sharing the idea of 'an anomalous state of knowledge' (aka ignorance) in the user with QA. But it has not been treated as QA. The good reason for this is that text retrieval has been interpreted as designed to meet a broader, and vaguer, need than QA, namely a need for information *about* a topic. TR requests are interpreted as for 'documents about X' e.g. about boundary layer flow, not for answers to questions like `'What is boundary layer flow?'`. Further, when users have submitted questions, they have been treated as for texts about something (as is clear from the early Cranfield test data, Cleverdon et al. 1966).

This interpretation of user need is not only a reflection of the challenge of building suitably powerful QA systems (more knowledgeable than human librarians). It reflects the more positive view that a user who wants to know X needs a good substantive response in the shape of material that supplies detail, and can be considered in the light of what the user already knows in a constructive mental dialectic.

On this assumption, IR research has clearly demonstrated that the user's needs can be met be met by applying essentially simple but soundly based statistical techniques, without any conventional NLP (Voorhees and Harman in press). These techniques can also support precision-oriented matching, in a sort of 'quasi-QA' (to which older conventional strategies like boolean searching, or newer ones like link exploitation, can also be applied). Statistical methods can of course also be used to retrieve shorter text segments than whole documents (though this has not been very thoroughly evaluated).

The TREC QA evaluations

It is evident, however, that there is a real need to answer questions, whether of the simpler kind like `'When was the Great Fire of London?'` or the less simple like `'Which is the best restaurant in Los Angeles?'`. Library experience, Web engines like AskJeeves, and IR engine query logs all show this. The TREC IR evaluation data has also contained user questions. The TREC QA track, running since 1999 (Voorhees 2000, 2001, 2002, in press), has been a novel and significant initiative on QA proper, applying the evaluation experience that TREC has established for large scale text retrieval tests to this related task. The TREC cycles, beginning with the most modest version of the QA task, have slowly progressed through task variations representing more demanding test conditions, with large data sets and extremely carefully designed performance evaluation.

The major difference between TREC and previous QA research has been that TREC addresses QA from free, general-purpose, open-domain text, the very opposite of the earlier database case. The general model has been single-shot 'factoid' questions, e.g. `'Who assassinated Abraham Lincoln?'`, with extracted text snippets as responses (primarily to avoid introducing the extra requirement for, and complication from, 'well-formed' answer generation). Initially snippets were either 50byte or 250byte. But it was soon clear that the statistically-based techniques used for conventional text retrieval were adequate for the latter, and in 2002 exact answer extracts replaced 50byte snippets. Initial constructed questions were replaced by real log-derived ones, with no guarantee of answers in the file, separate question sets of different types (factoid, list) were succeeded by mixed sets. Careful checking of answer assessment, and large data sets (eg 500 questions, $>$ 1M news items), suggest the general pattern of results is sound and, with its many participating teams, usefully instructive. It is important to note that while the evaluation relies on the notion of correct answer, there can be more than one 'expression' of the correct answer: the assessors, judging submitted results, might accept a range of variant answer forms, e.g. `'John Wilkes Booth', 'J.W. Booth'` for the question about Lincoln.

For the shorter answers, NLP-based systems have generally been better than statistically-based ones, though the latter are surprisingly competitive, and shallow pattern-matching techniques exploiting the Web as a data source have been unexpectedly effective. In earlier cycles the NLP systems shared a common model of question typing, candidate passage retrieval, and shallow sentence and question parsing and structure matching, i.e. essentially Simmons et al.'s early 60s strategy). However the systems gained from using new resources like Wordnet as a source of semantic information, and from experience (ultimately from the MUC evaluations) of named entity identification. Morever LCC's system (Moldovan et al. in press), which has performed extremely well, uses a deeper linguistic analysis and iterative strategy. There has been much more system variation recently, e.g. more elaborate typing, but also significant 'hybridisation', with extensive use of statistical information.

Top ranked performance in 2001, with snippets, measured by mean reciprocal rank over 5 ranked returns, was .68, with 11th ranked .32. Best performance in 2002, with exact answers and an average precision-type measure over single returns, was .85 (83% correct), 11th ranked .45 (29% correct). Given the challenges the evaluations present - the character of the task itself, the large, uncontrolled text file, and the very weak world knowledge available in the form of WordNet, gazetteers and so forth, - these are impressive results.

The TREC QA evaluations nevertheless raise many questions about the nature of the QA task. This is not a criticism of these evaluations per se. Past experience with retrieval evaluation, and with other task evaluations e.g. for information extraction and sumarising, shows how challenging well-founded evaluation is, and how many simplifications and compromises are necessary, especially initially, to size up what is involved, both in understanding the task and in establishing a suitable evaluation methodology. Moreover, the TREC tests are a background, or warm-up, for research on more ambituous forms of question answering, aimed at the higher quality system output that professional analysts require, which is being supported through the AQUAINT programme (AQUAINT 2003). In my analysis of QA in the next section I will use TREC QA as a discussion base because it raises key issues about QA as a task. But this is not intended as a critique of TREC, which is valuable precisely because it makes us focus on these questions in a way that only concrete evaluations, that require specific definitions of what questions and answers are, can.

TREC QA properties

The most salient differences between QA as interpreted in TREC and earlier research have already been noted: the unstructured data file, general-purpose material and hence open domain. In addition, because QA is one-shot (an initial study of questions in context was not followed up), there can be no clarification, no elaborative dialogue, no user modelling worthy of the name. TREC is also, by implication, concerned with general-purpose functionality: the questions are those that anyone (aka Joe Public) might ask, and the answers those that are natural and reasonable from that point of view.

All of these conditions are constraints on what the QA system is expected to deliver, and on the means it can adopt to do this.

But the real problem is in the key assumptions on which the whole edifice has rested: that the question is clear, and that its answer (assuming it's in the file) is clear and correct. There is no doubt that, with an 'extractive' model, exact extracts are better than fixed length fragments even if these contain the answer. However, as Voorhees and Tice's analyses (2000) show, the ground assumption is a flakey plank over a deep wide ditch. Even so innocent-looking a question as `'Where is the Taj Mahal?'` lays a trap for correctness (`'Agra, India'` vs `'Atantic City'`). Again, `'Who is Johnny Mathis's trainer?'` (`'Vasquez'` vs `'Lou Vasquez'`).

More importantly, underlying the key assumption there are three issues:

1. What is the nature of the user's need?
2. What is the appropriate response?
3. What is the QA interchange environment?

The problem, in a TREC QA-type situation, is that none of these are known. So the operational compromise has been to accept the assessor's decision as final but rely on these being based on a sort of common, general-knowledge, reference-librarian's approach than some random 'ordinary' user's (so, say, the Taj Mahal is in Agra).

However, as Voorhees and Tice also observed when moving from constructed to real questions, things look very different when the whole process is data driven, even if the file is very large. The set of perfectly proper answers you may get back may not match any common general knowledge or even 'rational' individual user expectation. This is not just a matter of technical correctness, for example:

```
Q   Who wrote 'The Antiquary'?
A1  The author of 'Waverley.'
A2  Walter Scott.
A3  Sir Walter Scott.
```

where A1 is what it says on the title page, and A3's 'Sir' was after publication. The significant difficulties arise in a case like this:

```
Q   Who is John Sulston?
A1  Former director of the Sanger Institute.
A2  Nobel laureate for medicine 2002.
A3  Nematode genome man.
```

There are no context-independent grounds for choosing any one of these, and I may not have any personal context-driven expectations about the sort of answer I want.

Moreover, if we recognise that users can get valuable information by inference and construction from new material combined with what they know already, the scope for QA is far wider. This can be far more than simply deriving the kind of exact answer TREC QA has been seeking but which we cannot expect QA systems to reach because of the rich knowledge or complex inference it requires, though this is of course an important gain. The point is rather that this information can be helpful in filling in the bigger picture even if it does not provide precisely what is asked for or, perhaps more realistically, supplies pertinent information when the user does not have a very clear idea of what would constitute a direct and correct answer. All of the three answers to `'Who is John Sulston?'` would suit this latter situation. Further, if we suppose that the question is `'Who is the director of the Sanger Institute?'` and also suppose that the database does not contain any information about who is the director now, is the answer `'John Sulston was the director of the Sanger Institute till 2000.'` of no value to the user? [1]

The guidelines for TREC-03 QA (currently in progress) are significant in that they respond in part to these issues. Thus while factoid questions need exact answers as before, a definition-type question can be answered by a set of 'nuggets', which deals with the problem illustrated by A1-A3 for Sulston above (though performance will still be by reference to an assessor-established reference set). Further, there is a subsidiary 'passage' task allowing up to 250 bytes of extract embedding a correct answer, reflecting the notion that the user may not find locating an answer in a short extract hard, while a system may (but not exploring the idea that the user may find context information helpful).

[1] Or, to take LCC's returned answer to the question `'Have WMD been found in Iraq?'`: `'Is it not ironic that that the only WMD found in Iraq were never lost in the first place...'`. Demonstration system visited August 2003.

Assessment is therefore under the same requirement as with exact answers, and the last point made about the Sulston example in the previous paragraph still holds. Indeed the problems posed by definition-type questions also apply elsewhere, e.g.

```
Q   When was the Great Fire of London?
A1  After the Great Plague of 1665, the Great Fire of London
    destroyed the city.
A2  The Great Fire of London, during the reign of Charles II,
    was a major catastrophe.
```

Doing QA right

These considerations of principle, as well as practical ones allowing for the *de facto* limitations of NLP systems, suggests that the QA goal needs revisiting. QA should not be designed on the assumption that it is rational to seek single, correct answers. We should rather accept that, with large, unstructured text files, there are likely to be a number of candidate answers which may be of value to the user. There will be multiple candidates for many different reasons: the question is ambiguous, the NLP systems capabilities are limited, the answer-related material in the database does not fully meet the question specification, and so forth.

This may not seem very satisfactory: the user is in danger of getting swamped by material that they do not find of use. But there is a potential remedy, the one used in text retrieval (and indeed implicit in the original TREC QA evaluation scenario), namely to rank the candidate answers. Of course, given the very weak contextual information that is available for the user, and the 'no-dialogue' protocol, ranking can only be hopeful rather than authoritative. However, as the similar retrieval case shows, even imperfect ranking can help the user find their way around the system output.

Thus, for example, suppose we have the question:

```
Q Are postcards normally made of cardboard?
```

and the 'correct' answer

```
A Postcards are normally made of cardboard.
```

(or any of its straightforward syntactic or lexical variants) is not in the file. Suppose, however, - and assuming full sentences as the form of responses, - that the file contains many candidate responses, which can be selected on some combined IR and NLP basis, and which include the following:

```
Aa Postcards are sold of cardboard.
Ab Postcards were made of cardboard.
Ac We make postcards of cardboard.
Ad Postcards are made of recycled boxes.
Ae My grandfather makes postcards of cardboard.
Af Cardboard is the preferred material for companies
   making postcards.
Ah Cardboard is in short supply so we cannot produce any
   postcards.
Ai White cardboard is better than grey cardboard for
   postcards.
```

Some ordering of the candidate responses - given that we are assuming that those given here are only some of the set, seems clearly helpful for the user. As the example suggest, finding sensible ordering criteria that only have syntactic, lexical, and shallow semantic, data to work on is no simple matter. But let us suppose we have some criteria and hence decide on the order:

```
Af > Ah > Aa > Ac > Ab > Ai > Ad > Ae
```

There is information in the candidates which is pertinent to whether boxes are normally made of cardboard and which the user could exploit to arrive at the conclusion that postcards are probably usually made of cardboard. Further, the ranking could make it easier for him to work through this information, even though the details of the ranking are open to argument: eg, what precisely, without any further text context, does `'We make postcards of cardboard.'` tell us?

It tells us something. But clearly, the user has to choose whether to dig further into its source text. If that text is a story about some Swiss Family Robinson effort in do-it-yourself postcard making, then it does not lend support to 'normally'. On the other hand, if the source is a company report from MegaStationery Inc, the user could take it as adding force to the 'probably usually' conclusion. The TREC tests called for source text support for submitted answers, to avoid 'bogus' correct answers and to help assessment, and subsequently, more narrowly for answer justification, again primarily for assessment and formal evaluation purposes. However as the example just given suggests, there are primary reasons, in the task itself, for looking at source support for answers, not just secondary reasons in the need for proper formal evaluations.

Conclusion

As noted, the TREC QA initiative came from research on text retrieval, and not from that on database access or dialogue systems. It is at least possible, however, that the early front end work has left a legacy in the NLP community, namely believing that one can expect to get exact and correct answers. But reviewing the earlier work shows how problematic this is, even in rather favourable conditions, like having a limited domain and permitting clarificatory dialogue. The difficulties encountered with the earlier front end and dialogue inquiry research remain, and indeed become more challenging, with the current version of QA, and are compounded by new ones, like not being able to have a domain model at all. There are some compensations, for example having large corpora to sustain system 'learning', and being able to pull out file text to support and amplify system responses (and there are also all the advantages of modern MMIs). But these cannot enable 'free-for-all' QA, when there is so little leverage to be got from the single-shot, single-sentence question without a context except, perhaps, to some extent under the 'obvious answer' model (where the Taj Mahal is the one in India). Of course having substantial user context information available would make it a different ball game, and one that is a background to the AQUAINT enterprise.

My argument is thus that for systems that can have extremely little information about the user, more *openness* is needed in response than is represented by the return of a single 'correct' answer, i.e. an exact extract or (ideally) a well-formed generated answer. For a variety of reasons, it may be impossible for the system to return a correct answer, but the system can nevertheless return useful information. This is not to imply that the notion of trying to answer questions, as opposed to doing (short) text retrieval using classical IR methods, is a mistake. Equally, arguing for more open QA systems when dealing with very large, free-text, open-domain data sources means more than simply calling for some snappy man-machine interface gizmos. Claiming that the long, earlier period of work on database front ends may have had a hidden, but malign, effect on current QA research, through encouraging a belief in the possibility of exact answers, does not mean that there is no value in question processing with the pinpointing that requires some degree of NLP. As the earlier postcard examples suggest, question and potential answer analysis can be expected to impose some useful focusing on the varied materials that are likely to lurk in the data file.

The TREC QA evaluations are very important as ways of exploring the QA strategy space, especially given the challenges involved in designing performance evaluations that are legitimate and informative. But I believe they are better seen as technology studies, than as task ones: QA in the real world has to be more hospitable, and flexible, than any QA research so far has assumed.

References

AQUAINT: see www.ic-arda.org/infoExploit/aquaint/index.html
(visited November 2003)

Bobrow, D.G. et al. 'GUS, a frame-driven dialogue system', *Artificial Intelligence*, 8, 1977, 155-173.

CL-88: *Computational Linguistics*, Special Issue on User Modelling, Discussion Section, 14, 1988, 79-103.

Cleverdon, C.W., Mills, J. and Keen, E.M. *Factors determining the performance of indexing systems*, 2 vols, College of Aeronautics, Cranfield, 1966.

De Mori, R. (Ed.) *Spoken dialogues with computers*, London: Academic Press, 1998.

Grosz, B.J. et al. 'TEAM: an experiment in the design of transportable natural-language interfaces', *Artificial Intelligence*, 32, 1987, 173-243.

Moldovan, D. et al. 'LCC tools for question answering', *The Eleventh Text REtrieval Conference (TREC-02)*, Spec Pub 500-251, Washington DC: NIST, 2003, in press.

Raphael, B. 'SIR: Semantic information retrieval', in *Semantic information processing*, Ed. M. Minsky, Cambridge MA: MIT Press, 1968.

Simmons, R.F. 'Answering English questions by computer', *Communications of the ACM*, 8, 1965, 53-70.

Simmons, R.F., Klein, S. and McConlogue, K.L. 'Indexing and dependency logic for answering English questions', *American Documentation*, 15, 1964, 196-204.

Voorhees, E.M. 'The TREC-8 question answering track report', *The Eighth Text REtrieval Conference (TREC-8)*, Spec Pub 500-246, Washington DC: NIST, 2000, 77-82.

Voorhees, E.M. 'Overview of the TREC-9 question answering track', *The Ninth Text REtrieval Conference (TREC-9)*, Spec Pub 500-249, Washington DC: NIST, 2001, 77-82.

Voorhees, E.M. 'Overview of the TREC 2001 question answering track', *The Tenth Text REtrieval Conference (TREC-01)*, Spec Pub 500-250, Washington DC: NIST, 2002, 42-51.

Voorhees, E.M. 'Overview of the TREC 2002 question answering track', *The Eleventh Text REtrieval Conference (TREC-02)*, Spec Pub 500-251, Washington DC: NIST, 2003, in press.

Voorhees, E.M. and Harman, D.K. (Eds.), *TREC: Experiment and evaluation in information retrieval*, Cambridge MA: MIT Press, in press.

Voorhees, E.M. and Tice, D.M. 'The TREC-8 question answering evaluation', *The Eighth Text REtrieval Conference (TREC-8)*, Spec Pub 500-246, Washington DC: NIST, 2000, 83-105.

Woods, W.A. 'Semantics and quantification in natural language question answering', in *Advances in Computers*, Ed. M. Yovits, Vol. 17, New York: Academic Press, 1978.