Position statement: Inference in Question Answering

Bonnie Webber*, **Claire Gardent**†, **Johan Bos***

*Division of Informatics
University of Edinburgh
Edinburgh EH8 9LW, UK
{bonnie,jbos}@cogsci.ed.ac.uk

†CNRS – LORIA
BP 239 – Campus Scientifique
54506 Vandoeuvre-les-Nancy, FRANCE
claire.gardent@loria.fr

Abstract

One can only exploit inference in Question-Answering (QA) and assess its contribution systematically, if one knows what inference is contributing to. Thus we identify a set of tasks specific to QA and discuss what inference could contribute to their achievement. We conclude with a proposal for *graduated test suites* as a tool for assessing the performance and impact of inference.

1. Introduction

Our point in this position statement is that, to use inference in Question-Answering (QA) in a way that will support what Barr and Klavans (2001) call *component performance evaluation* – assessing the performance of system components and determining their impact on overall system performance – one must identify specific *question-answering tasks* that can potentially gain by exploiting inference. In the first generation of QA systems (i.e., those designed to answer questions in terms of information in structured databases), only a few QA tasks were seen to need inference. In all cases, inference complemented the extensional process of relational (SQL) database querying, through reasoning on the concepts involved:

- Stallard (1986) used terminological reasoning (in a description logic) for the task of mapping from the logical form (LF) representation of a user's query and the concepts it was couched in, into the concepts and relations that formed the *data model* for the database.

- In the context of QA from multiple databases, inference was used in (Hendrix et al., 1978) in the task of developing plans for what databases to access for concept extensions, which would then be combined to produce an answer.

- Kaplan (1982) used inference on the query and its presuppositions for the task of generating a response to a question whose direct answer was not deemed useful.

- Pollack (1986) used inference on the query and an enhanced *data model* for the task of identifying and correcting user misconceptions that underlay otherwise unanswerable (or not usefully answerable) questions.

- In (Mays, 1984; Mays et al., 1982), when a question couldn't be usefully answered at the time it was asked, inference in the form of a temporal tableaux reasoner was used to generate a response to a question whose direct answer was not deemed useful. Specifically, it was used to identify whether the situation described in the question could occur in the future. If so, the QA system could offer to monitor for its occurrance, at which time the question could be answered.

Not all of these QA tasks are relevant to today's (or even tomorrow's) Open-Domain QA systems, which are designed to answer questions on the basis of *unstructured data* (i.e., free text). Nevertheless, it is still the case that there are places where inference can enhance the capabilities of Open-Doman QA systems (Burger et al., 2000; Hirschmann and Gaizauskas, 2001) and/or improve the quality and/or accuracy of their answers. As already noted, our point in this position statement is that, to use inference to these ends, one must identify specific *question-answering tasks* that will drive inference. This will then allow development of the kinds of *graduated test suites* with respect to which *evaluation* can be carried out on both the QA system and the inference engines themselves.

Note that the position we are taking here is very similar to that in (Hobbs et al., 1993), where the authors identify a set of *discourse tasks* that need to be solved in order to explain why the sentences of a text, in combination, would be true. These *discourse tasks* include (but are not limited to): interpreting compound nominals; resolving definite referring expressions; further specifying vague predicates; identifying how predicates apply to their arguments; disambiguating the arguments to predicates; determining coherence relations between adjacent segments of text; and detecting relation of an utterance to the speaker's overall plan. These, in turn, may depend on solving lower-level tasks such as resolving attachment and/or word sense ambiguities, resolving anaphora, and filling in missing (semantic) arguments. But by first specifying the discourse tasks, the authors can show exactly how inference (in their case, *weighted abduction*) can potentially – with efficient search and sufficient background knowledge – be used to solve them. (Note that weighted abduction is not a technique for *forward reasoning*. So any discourse task that requires determining the additional conclusions that can be drawn from a text may require another form of reasoning.)

In the first part of this statement, we identify a set of *question-answering tasks* in which inference could allow enhanced or extended QA services. Our goal is not to comment on what has or has not already been done in using inference in Open-Domain QA systems, but rather to lay out general areas where inference can contribute. We conclude by saying a bit more about *graduated test suites*.

2. QA Tasks

For this short position paper, we restrict the label *QA tasks* to ones that follow from a *functional role* of question or answer, rather than as text *per se*. That is, it is well known that inference can support discourse processing: texts can be parsed using *deduction* – it is what DCGs are all about – and (theoretically) they can be assigned a consistent explanatory interpretation using a combination of *weighted abduction* (Hobbs et al., 1993) and *consistency checking* (Blackburn and Bos, forthcoming). While this kind of interpretation can knit together elements of a text and supply missing (implicit) elements of its fabric, and thereby be critical for deriving answers to particular questions or even particular classes of questions, discussing the role that inference can play in discourse understanding requires its own paper, which we or other people should write.

Similarly, QA interactions are *dialogues*, and work done by Perrault, Cohen, Allen, Litman, Pollack, Walker and others has clearly shown that inference is needed to support dialogue processing – e.g., to decide what a question is really asking for. But this too is a large enough area to require its own paper.

Our focus in this paper then is on the significant set of tasks that remain after both discourse and dialogue understanding are, for the moment, put aside. Among these, we can identify several where inference could provide enhanced or extended QA services.

2.1. Expanding the search criteria for *potential* answers

It is standard procedure in QA to establish search criteria based on the question that has been posed. These search criteria make up the formal *query*, which is used to find *potential* answers in the form of candidate documents that may provide evidence for or contain a *proper* answer.

To increase the yield of potential answers, alternative terms can be added to the query. While this does not intrinsically require inference, what inference can do is expand queries with truth-functionally or defeasibly equivalent *global* reformulations of the original question. These can be used to augment the query with terms that could not have been identified using essentially *local* translation of individual words that ignores their context and functor-arguments dependencies, including implicit (semantic) arguments. For example, abductive reasoning on the question

(1) What do penguins eat?

(solving the implicit argument of *when* the eating event takes place – the same generic "in general" as the generic subject penguins) might produce a defeasibly equivalent version in terms of their *staple diet*. This term would not be added for a question like

(2) What did the characters eat in the seduction scence from the film "Tom Jones"?

which has its (optional) event argument instantiated.

Inference can also expand a query with one-way *entailments* of the original question. For example, being *awarded a degree in Computer Science* (CS) entails being *enrolled for a CS degree*. Given the question

(3) How many students were enrolled in Computer Science at Cambridge last year?

computing its one-way entailments would allow the query to be expanded with *award*$\wedge$*degree*.

Finally, inference can expand queries through subconcepts that form a *partition* (i.e., disjoint cover) of a concept in the original query; a distinct sub-query can be formed for each one. In this way for instance, the query

(4) How many people work for IBM?

could be decomposed into a set of sub-queries such as e.g., *How many men work for IBM? How many women work for IBM* or *How many white collar workers does IBM have? How many blue collar workers does IBM have?*.

Although we have discussed these expansion techniques in terms of constructing a query (either initial or follow-up, in case the initial query does not produce sufficient results), the same techniques could benefit the *ranking* of potential answers with respect to the question, if *recall* on the original query is felt to be sufficient.

2.2. Determining *proper* answers from *potential* answers

A proper answer to a wh-question may be found within a single clause, or it may be distributed through the potential answer (*answer locality*). Moreover, a proper answer may be explicit in the text (i.e., derivable simply by pattern matching), or it may require inference or other method of information fusion (*answer derivability*).

Even where an answer appears to be *explicit* in a text, inference can help determine whether it is a *proper* answer (Bos and Gabsdil, 2000), as with the following potential answers to:

(5) Q: Who invented the electric guitar?
A1: Mr. Fender did not invent the electric guitar.
A2: The electric banjo, cousin of the electric guitar, was invented by Bela Fleck.

A proper answer to this question must entail either (1) that there is someone who invented the electric guitar, or (2) that there is no such person, or (3) that it is true of everyone. All of these are logical relations between a potential answer and a representation of the question in terms of its question domain D (here, persons) and its body B (here, inventing the electric guitar). As such, inference can be used to determine whether any of these relations hold.

Inference can also help when *proper answers* are only implicit in *potential answers*. In (Hobbs et al., 1993), Hobbs et al. show that *weighted abduction* can be used to solve a variety of *discourse tasks*, thereby making explicit information that is implicit in a text. This can be applied to potential answers. For example, a potential answer to the question

(6) Where do condors live?

might contain the compound nominal *the California condor*. As in resolving "the Boston office" (Hobbs et al., 1993), this can be (abductively) resolved to condors whose location is California. That this is a matter of abductive inference rather than simple pattern matching, can be seen by not wanting to draw similar conclusions in determining proper answers to the similar question

(7) Where do terriers live?

Here, compound nominals such as "Yorkshire terrier", "Boston terrier", "West Highland terrier", etc. in potential answers would yield such incorrect proper answers as Yorkshire, Boston, etc.

There is much more to be explored here. Nevertheless, it is clear that inference can be used to support more than one aspect of this task.

2.3. Comparing *proper* answers to wh-questions

The way in which answers are sought in open-domain QA means that one cannot avoid the problem of determining whether proper answers derived from different potential answers (candidate documents) are the same (i.e., mutually entail one another) or different. In the latter case, one may also not be able to avoid the problem of determining whether (i) one answer is more specific than another (i.e., the more specific answer entailing the more general one, but not vice versa); (ii) two answers are mutually consistent but not entailing in either direction; or (iii) two answers are inconsistent. Determining such relations among proper answers becomes a QA task for Open Domain QA, where it was not one for database QA because the underlying relational DB query system was able to recognize and remove all duplicates.

The outcome of such determination depends on whether the original question is taken to have a single answer (a unique individual or property or set) or alternative answers, the set of which is of unknown cardinality. Whatever the reason, these are problems that inference can help solve.

- Answers determined to be equivalent (mutually entailing) can be replaced by a single member of the equivalence class;
- Answers that differ in specificity (one-way entailing) can be replaced by either the most specific one (as with the answer to *When was the Bastille taken?*, where *14 July 1789* is preferred over the less specific *14 July* and *1789*) or by a conjunction of the most specific answers (as with answers to *Who is Noam Chomsky?*, where *MIT linguist*$\wedge$*left-wing activist* is the preferred way to combine the answers in the set *MIT linguist*, *linguist*, *MIT academic*, *political activist* and *left-wing activist*);
- Answers that are mutually consistent but not entailing can be replaced by their conjunction (as with *MIT linguist* and *left-wing activist* above);

- Answers that are inconsistent are the only true alternatives. In the case of questions with unique answers, only one of them can be correct. In the case of questions with alternative answers such as *Where do penguins live?*, all the alternatives may be distinct proper answers.

2.4. Comparing questions

Where efficiency is a goal of QA, it can be supported by determining whether a new question is one that has previously been answered (Harabagiu et al., 2001) or is related in a systematic way to one that has previously been answered. (This is the reason that FAQ-lists exist.) Inference is a valid way of computing both *equivalence* relations between questions and *subsumption* – i.e., whether one question is more specific than another one. The latter allows two different forms of answer re-use. Consider the questions

(8) Where can I go skiing in the Northern Hemisphere in June?

(9) Where can I go for winter sports in the Northern Hemisphere in June?

If one has cached the answer to (8), then one has a partial answer to question (9), which subsumes it. Conversely, if one has already cached the answer to the subsuming question (9), that answer may contain or provide a basis for an answer to question (8). That is, if (9) has been answered by answering the set of questions that follow from each possible way of instantiating the general term "winter sports", then one already has an answer to (8). On the other hand, if question (9) has been answered in general, then (much as with the "linked" questions in TREC-10) sources for that answer might prove a good place to start looking for an answer (8), rather than posing it against a completely open domain.

2.5. Determining *proper* answers to yes/no questions

One may take the set of proper answers to a yes/no question to comprise simply *yes* and *no*, or one may take it more broadly to include temporal and/or modal qualifiers as well – eg. *possibly*, *sometimes*, *it depends*, etc. In the first case, determining a proper answer requires identifying what support exists for a positive answer (*yes*); what support exists for a negative answer (*no*); and on which side the support is stronger. Practically, this could involve separate queries – one seeking evidence for the positive assertion, the other, for the negative assertion. These queries could differ because lexical items can have distinct negative-polarity counterparts. For example, given the question

(10) Does Anacin contain any stimulants?

a query seeking evidence for the positive statement might contain the terms ANACIN, CONTAIN and STIMULANT, while the query seeking evidence for the negative statement might contain the terms ANACIN, LACK and STIMULANT. But because *potential answers* retrieved in response to such questions may themselves contain explicit negation (i.e., *no* or *not*), deciding what they support requires determining the scope of negation. Here, inference can determine which of the readings are consistent. Inference can also be used as discussed in Section 2.2. to determine whether two pieces of *evidence* are the same or different, so that instances of the same evidence or instances of stronger and weaker evidence aren't multiply counted.

In general, it is easier to find positive evidence than negative evidence, as what does not hold is most often conveyed implicitly, by the lack of evidence for it (i.e., the *closed-world assumption*). But for certain yes/no questions, evidence for a negative answer may be easier to come by than for a positive one. For example, in a question with a universal quantifier such as

(11) Did Larsson score in every game he played for Celtic?

a single piece of negative evidence (e.g., "Larsson failed to score in Tuesday's game") is needed to justify a negative answer, while a positive answer requires either a potential answer that itself contains a universal quantifier or a set of potential answers that cover the entire set of games. The latter is essentially (extensional) database question-answering, with the *closed-world assumption* that the database covers all positive instances.

2.6. Generating responses in lieu of or support of a direct answer

Unlike in TREC-9, TREC-10 systems were asked to identify when they couldn't answer a question. In database QA, finding no answer to a question was not an uncommon occurence. One reason for this occurring was failure of a presupposition in the question. For example, the question

(12) Have any women been awarded a Pulizer prize for sports journalism?

may have the direct answer *None* because the existential presupposition that there is a Pulizer prize for sports journalism is false. Hence, techniques were developed (Kaplan, 1982) for recognising presupposition failure and for generating responses such as *There is no Pulizer prize for sports journalism*. But as shown in (Blackburn and Bos, forthcoming), verifying presuppositions involves inference in order to check their consistency and informativity in context.

Another reason for not being able to answer a question is that *positive* information is lacking. Here, a partial response can be formulated if *negative* information can be found that *excludes* something from the set of proper answers. For example, given the question

(13) Which French cities did Reagan like?

information to the effect *Reagan disliked Paris* provides a useful partial response. Inference can be used to recognize that an individual is excluded from the set of proper answers.

A third situation motivating a response is the case of negative answers to extensional yes/no questions, which are rarely very informative – e.g.

(14) Q: Did Hearts played a home game against Celtic in January?
A: No.

In such cases, the answer to a "weaker" question – one that can be computed from the original one by subsumption reasoning, may provide the basis for a useful response – e.g. *Did Hearts play a game against Celtic in January?* or *Did Hearts play a home game against Celtic?* or *Did Hearts play a home game in January?*. More complex questions, such as ones containing quantification and/or negation, may require more complex subsumption reasoning to establish weaker questions that are worth posing.

Note that weakening the question only makes sense for questions answered extensionally, not ones answered through inference or pattern matching such as

(15) Do penguins migrate?[1]

Other situations in which responses are useful in lieu or support of a direct answer, many of which require forms of inference, are described in (Webber, 1986).

3. Graduated Test Suites

While TREC evaluation of QA systems has focussed on the full end-to-end task, some systems have also carried out what Barr and Klavans (Barr and Klavans, 2001) call *component performance evaluation* – assessing the performance of system components and determining their impact on overall system performance. The components of interest here are those that use inference. We see *graduated test suites* as a tool for assessing their performance and impact, allowing: (1) comparison against similar components that do not use inference; (2) comparison of components that differ in what inference tools they use; and (3) assessment of the impact of improvements in inferential ability. We also see graduated test suites as a way of evaluating automated reasoning tools on the inference problems raised by QA.[2]

We now discuss two of the above QA tasks, making explicit what one would expect to see in a distinct test suite for each. As in TREC, developing the test suites would involve carefully crafting a set of examples to the correct level of difficulty, fixing evaluation criteria and delimiting in a more precise way the linguistic task involved.

Expanding the query. Section 2.1. identifies four ways of expanding the query: through equivalence, through entailment, through multiple sub-queries and through abduction. For each of these tasks, inference can be involved as follows.

When expanding the query with semantically equivalent reformulations, inference can be used in at least one of two ways: First, given a subsumption based hierarchy KB encoding relations between word meanings, inference can be used to *find* the set of (structured) concepts which are logically equivalent to the structured concept representing the initial query. Alternatively, for reformulations produced by some other mechanism (e.g, parsing the query and then generating paraphrases from the resulting semantic representation(s)), inference can be used to *check* that they are indeed semantically equivalent.

Similarly, when expanding the query with more specific variants, inference can be used either to *find* within a hierarchy, the set of most specific concepts subsumed by the concept representing the query, or, for potential variants found by other means, simply to *check* that each indeed stands in some kind of entailment relation to the initial query.

Thirdly, when expanding concepts (and/or sets of concepts) in the query into *partitions* (i.e., disjoint covers) of more specific sub-concepts, the task for automated reasoners would be to check that the conjunction of queries $Q_1, \ldots, Q_n$ obtained by replacing a concept in the original query Q by a partition of its immediate sub-concepts is equivalent to the original query.

Finally, queries can be expanded by making implicit information explicit. This requires some kind

[1]Many types of penguin migrate, swimming north each autumn in the Southern Hemisphere and south each spring.

[2]Automated reasoners have been optimised for their performance on problems from mathematics and logic. As this is not necessarily optimal for NL problems, we need to drive their optimisation in this direction. That is the reason for having test suites for both QA components and automated reasoners.

of abduction – e.g, weighted abduction (Hobbs et al., 1993) or model building (Gardent and Konrad, 2000a; Gardent and Konrad, 2000b). With the first, the reasoner is given a semantic representation of the query, along with relevant world, domain and/or lexical knowledge and returns the cheapest explanation (proof) of the query, making explicit the hypotheses (either abduced or assumed) that support it. Similarly, model building will produce a (minimal) model satisfying the formula which encodes the explicit and implicit information expressed by the query.

In all cases, the information (facts in model or logical formulae) resulting from query expansion can be converted to a form appropriate to the query. If queries are Boolean combinations of key words and/or phrases, NL Generation techniques can be applied to each semantic component to produce a parse tree whose leaves constitute a string of lexical *lemmas*, from which key words and phrases can be identified and added to the query.

Determining proper answers. For **wh-questions with a single answer**, the problem of determining a proper answer from a potential answer depends on (i) the *expected answer type* (positive, negative, unknown); (ii) the *answer locality* (whether the answer is contained in a single clause or distributed over the text), and (iii) the *derivability* of the answer (whether it is explicit in the text and derivable simply by pattern matching, or it requires inference or other method of information fusion).

Test-suite examples could therefore be divided into 12 classes, of different complexity, depending on the values of these factors. For example, consider *expected answer type*. Formulated in first-order logic, with ϕ_A representing the meaning of the potential answer A, D the domain of the question and B its body, (1) if the expected answer type is positive, there is at least one object having the properties set by the question. So the inference task is simply: **Prove** $\models \phi_A \rightarrow \exists x(D(x) \wedge B(x))$**.** (2) Alternatively, if the expected answer type is negative, there is no object having the properties set by the question. So the inference task is: **Prove** $\models \phi_A \rightarrow \neg\exists x(D(x) \wedge B(x))$**.** (3) Finally, if the expected answer type is unknown, then *both* the above inference tasks are required.

For **questions with multiple answers**, we can only comment now on the use of inference for questions that can be expanded into a set of more specific sub-queries with known cardinality, such as

(16) What is the longest river on each continent?

which can be expanded into *What is the longest river in Europe? What is the longest river in Asia?* .... Once expanded in this way, each sub-query is a simple wh-question with a single answer. This is then the case discussed earlier.

4. Summary

There is no question that QA would not also be enhanced through the use of inference in *discourse tasks* involved in finer-grained examination of the texts retrieved in response to user-queries. It would likewise be enhanced by the use of inference in *dialogue tasks* involved in understanding the user's current utterance with respect to the current QA dialogue. Here we have focussed solely on the use of inference in *QA tasks* – tasks that follow from the *functional role* of a question or an answer – and how it could contribute to achieving these tasks, over and beyond methods that don't use inference.

When considering the development of *graduated test suites* to assess system performance on QA tasks and its impact on overall system performace (and also the performance of automated reasoning tools), it makes sense to consider the use of previous TREC questions and the set of passages (potential answers) that the retrieval components of TREC QA systems have returned in response. The usefulness of doing so is most obvious in the case of two of the tasks discussed here: determining proper answers from potential answers and comparing proper answers to wh-questions. What now requires discussion is what to do next.

5. References

Valerie Barr and Judith Klavans. 2001. Verification and validation of language processing systems: Is it evaluation? In *Proceedings of ACL Workshop on Evaluation Methodologies for Language and Dialogue Systems*, Toulouse, France.

Patrick Blackburn and Johan Bos. forthcoming. *Computational Semantics*. Current draft available from http://www.comsem.org.

Johan Bos and Malte Gabsdil. 2000. First-order inference and the interpretation of questions and answers. In *Proceedings of Gotelog 2000*, pages 43–50, Goteborg, Sweden.

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, and *et al.* 2000. Issues, tasks and program structures to roadmap research in question & answering. Technical report, National Institute of Standards and Technology. Available on-line at http://www-nlpir.nist.gov/projects/duc/papers/QA.roadmap-paper_v2.pdf.

Claire Gardent and Karsten Konrad. 2000a. Interpreting definites using model generation. *Journal of Logic, Language and Information*, 1(2):193–209.

Claire Gardent and Karsten Konrad. 2000b. Understanding each other. In *Proceedings, 1st Annual Meeting of the North American Chapter of the ACL*, Seattle WA.

Sanda Harabagiu, Dan Moldovan, and et al. 2001. Falcon: Boosting knowledge for answer engines. In *Proceedings of the 9th Text Retrieval Conference (TREC 9)*, pages 479–488, National Institute of Standards and Technology. Available on-line at http://trec.nist.gov/pubs/trec9/papers/smu.pdf.

Gary Hendrix, Earl Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, 3(2):105–147.

Lynette Hirschmann and Rob Gaizauskas. 2001. Natural language question answering: The view from here. *Natural Language Engineering*, 4.

Jerry Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. 1993. Interpretation as abduction. *Artificial Intelligence*, 63(1-2):69–142.

Jerrold Kaplan. 1982. Cooperative responses from a portable natural language database query system. In Michael Brady and Robert Berwick, editors, *Computational Models of Discourse*, pages 167–208. MIT Press, Cambridge MA.

Eric Mays, Aravind Joshi, and Bonnie Webber. 1982. Taking the initiative in natural language data base interactions: Monitoring as response. In *Proceedings of the European Conference on Artificial Intelligence*, pages 255–256, Orsay, France.

Eric Mays. 1984. *A Modal Temporal Logic for for Reasoning about Changing Data Bases with Applications to Natural Language Question Answering*. Ph.D. thesis, Dept of Computer and Information Science, University of Pennsylvania, Philadelphia PA.

Martha Pollack. 1986. *Inferring Domain Plans in Question-Answering*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania.

David Stallard. 1986. A terminological simplification transformation for natural language question answering systems. In *Proceedings of the 24th Annual Meeting, Association for Computational Linguistics*, pages 241–246, Columbia University.

Bonnie Webber. 1986. Questions, answers and responses. In Michael Brodie and John Mylopoulos, editors, *On Knowledge Base Systems*, pages 365–401. Springer-Verlag, New York.