Dividing Up The Question Answering Process*

Marc Luria

Division of Computer Science
Department of EECS
University of California, Berkeley
Berkeley, Ca. 94720

Abstract

This paper describes a question answering program which divides question answering into two separate processes: answer formation and answer expression. Rather than gathering possible answers and choosing the best among them, the program accesses the database and finds all components of possible answers, e.g. a causal chain, and then passes this information to an expression program which formulates a proper answer.

1. Introduction

I have developed a question answering program that will answer questions about simple stories. In my program, question-answering is divided up into two separate processes: 1) answer formation and 2) answer expression. The program first looks down a causal chain which is formed by the story-understanding program and figures out in what part of the chain the answer lies. The answer can also be a subset of the chain, sometimes a quite long one. The second part of the program takes this long chain and decides what things are important to express to the questioner. This answer expresser uses general rules of expression to figure out what it needs to include to make the answer understandable, informative and interesting.

This solution is different from other question-answering algorithms (e.g. Winograd 1972, Lehnert 1977) which view question answering as one process. These programs gather possible answers, and then choose the 'best' answer from among them. My system first gets the chain which I consider to be the answer to the question, and then figures out which parts of the chain should be generated into English as the answer. The advantage of my approach is that it allows one to treat the answer as one entity and use the answer expression mechanism to express what people are interested in. The resulting answers are generally more informative and conversationally appropriate than those generated by other algorithms.

The program works in conjunction with FAUSTUS, a story understanding program that specialises in goal-based stories. [2,3] After a story is initially 'read' by PHRAN, a parser, [1] it is then passed to this PAMELA and 'understood'. The question answering program is passed a database which consists of events, inferences, and most importantly, for my purpose, causal chains which instantiate events in the story as steps of particular plans and plans for particular goals. Contained in this causal chain is the actual 'understanding' of the sequence of events in the story, what caused what, and what goal actors had in mind when they performed a particular act or plan. After a question is asked, this question is parsed by the same parser that parsed the story, and then the answer is formulated by looking at the database. Finally, the answer is passed to the answer expresser which sends the answer to a natural language generator.

*This research was sponsored in part by the Office of Naval Research under contract N00014-80-C-0732 and the National Science Foundation under grant MCS79-06543.

2. Program Examples**

The following examples were processed by my program.

Story: Susan saved her money from her allowance. One day she rode her bike to the bookstore and bought the book that her teacher had recommended. Susan did very well on her math test the following week.

Q1: Why did Susan buy the book?
A1: So that she could study from it and do well on her exam.

Q2: How did Susan do so well on her math exam?
A2: She bought a book that her teacher had recommended and studied from it.

Q3: How did she get the book?
A3: By riding her bike to the bookstore.

3. Finding the Best Answer

A difficult and important part of answering a question is not in finding an answer to the question, but finding the best answer. In a database of causal chains, if one can find an event in the database then there may be many possible answers to a given question. Consider the previous story.

If we ask:

Q4: Why did Susan buy the book?

The following answers are obtained by stepping down the causal chain.

A4a:Because she wanted to have it.
A4b:Because she wanted to read it.
A4c:Because she wanted to know math.
A4d:Because she wanted to do well on her exam.

Note that the items nearer the top of the goal structure constitute better answers although the best answer would be something like:

A4e:So that she could study it and do well on her math test.

However, in a more complicated story, merely looking to the end of chain might not work quite as well. For example, if in the previous story we added:

> She put the book on her head and learned the material through osmosis. Susan did very well on her math test the following week.

Clearly, Answer4d is no longer a good answer.

One possible solution is including only 'important' answers. Important inferences might include abnormal plans, natural disasters, etc. The problem with this was that even though these 'important' inferences definitely should be included in the answer, one should not necessarily stop at that point in the chain and say that this is the answer. For example, just stopping at 'important' events in response to question4 one would get:

A4f:So that she could put it put it on her head.
A4g:So that she could learn by osmosis.

which is less desirable than:

A4h:So that she could learn from the math book by osmosis and do well on her exam.

4. Dividing up the Question Answering Process

My program is able to find these better answers because of the separation of finding the answer (the subset of the chain) from expressing the answer to the user. Instead I use the two programs:

Answer-Formulator: looks down a causal chain, figures out where what parts of the chain are relevant to an answer and returns a chain.

Intelligent-Expresser: takes this causal chain as input, figures out from its general rules of expression what is important to say so that the questioner will a) understand the answer and b) get the kind of information that people are generally interested in, and outputs to a natural language generator, some intermediate form from which it could generate an answer.

**At this point the program is not connected to the natural language parser at Berkeley called PHRAN or the generator PHRED (Wilensky and Arens, 1980). The questions and answers are therefore translated from the conceptual form I now use.

For example, my program would produce answer4e above by the following process. First it would find Susan buying the book in the database and then follow the chain, in this case, to where it finds that she did well on her exam. This whole part of the chain is passed to the expression mechanism which would notices that studying the book and doing well on her exam were important parts of the answer. In this case, the Intelligent-Expresser uses the general conversational rule of not informing someone of something they already know. Having the book and reading the book are thereby eliminated because they are stored in the data base as normative purposes for buying and for having a book, respectively.

This approach also allows one to generate answers that were otherwise problematic to represent in a conceptual form. For example, the simple question:

Q5: Did Susan go to the bookstore?
A5: Yes, she rode her bike there.

The answer is obviously yes, because this event appears in the database. However, 'yes' is something that is difficult to represent in conceptual form. 'Yes' is not really a concept but rather a word that is almost exclusively used in a conversation. The answer formation part of my system looks in the database for concepts similar to going to the bookstore. Realizing that riding to the bookstore was similar to going there it would answer:

```
(ride
  (actor (person (object susan1)))
  (object (bicycle (object bicycle1)))
  (destination (bookstore(object bookstore1]
```

This part of the chain and the context in which the question was asked is passed to the answer expression part of the program, that would a) see that this is a simple verify question, b) realize that the concept to be verified was in fact found in the database in a slightly different form and c) figure out that it should answer 'yes' plus some intermediate form that represents that it should include the ride concept.

This same method can be extended to other types of verify questions. For example,

Q6: Did Susan ride her bike to the bookstore so that she could do well on her math test?
A6: Yes, she bought a book at the bookstore which she used to study for her exam.

Q7: Did Susan buy the math book so that she could do well on her math test?
A7: Yes, she used it to study for her exam.

The answer formation part looks to see if a chain with the starting place of 'riding to the bookstore' and ends with 'doing well on her math test', exists in the database. This whole chain does exist and includes, she rode to the bookstore was a plan for being at the bookstore, which was a precondition for buying a book, which was a plan for having the book which was a step of reading the book, which was a plan for knowing the math material, which was a goal from doing well on her exam.

The answer expression part of the program gets this chain, realizes it should answer 'yes' and decides how much in addition to the 'yes' it would need to include in the answer. Notice how in Answer4 it had to include more information from this chain than it had to include in Answer5.

5. Conclusion

This intelligent expression part of the program is not something that is designed to be used exclusively in question-answering but is a system that would be valuable in any context where an interactive natural language system is important. It differs from a generator in that it does not merely generate something from a conceptual form into English, but rather decides what kinds of things are important to be said, which is then passed to a generator. Hopefully, this kind of system could be expanded to work on other conversational tasks as well.

References

[1] Lehnert, W., 1978. *The Process of Question Answering: A Computer Simulation of Cognition.* Hillsdale, N.J. Lawrence Erlbaum Associates, Inc.

[2] Wilensky, R. , 1978. *Understanding Goal-Based Stories.* Technical Report 140, Computer Science Department, Yale University, New Haven, CT.

[3] Wilensky, R. and Arens, Y. 1980 *PHRAN - a Knowledge Based Approach to Natural Language Analysis*. University of California at Berkeley. Electronic Research Laboratory Memorandum No. UCB/ERL M80/34.

[4] Wilensky, R. 1981. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science*, Vol. 5, No. 3. 1981.

[5] Winograd, T. 1972 *Understanding Natural Language*. New York. Academic Press.