Poetics 19 (1990) 99–120
North-Holland

A SURVEY OF QUESTION ANSWERING IN NATURAL LANGUAGE PROCESSING *

Stefan WERMTER and Wendy G. LEHNERT **

This paper surveys the different basic approaches of Question Answering (QA) used over the last two decades. Our goal is not to give an exhaustive overview of QA-systems, but to capture the main trends and developments, the progress being made, and some of the problems still remaining. QA-systems are classified as procedural QA-systems, conceptual QA-systems, and logic-based QA-systems. Procedural QA-systems belong to the earliest attempts to model question answering by attaching specific procedures to words. Conceptual systems emphasize the aspect of meaning representation of questions, memory, and answers. Logic-based systems use logic predicates to represent knowledge for question answering. General issues of noun phrase disambiguation, variability, and cooperative behavior are described as examples of problems during question analysis, answer generation, and user modeling. Uniform knowledge representations, logic representations, dictionaries, and transportability are identified as current trends in question answering.

1. Introduction

Question answering is a field of natural language processing with two complementary goals: first, the examination of basic issues in natural language understanding and generation and second, the development of natural language interfaces. Question answering (QA) systems are interactive systems relying either on their own memory representation or communicating with memory structures of underlying systems like database systems, expert systems, or tutoring systems. The basic task of QA-systems is to provide the user with wanted information.

In the last two decades hundreds of more or less sophisticated QA-systems have been developed for different tasks and domains. The following classification should serve as a framework for comparing the major approaches in question answering. QA-systems can be classified as procedural QA-systems,

* This research was supported by the Advanced Research Projects Agency of the Department of Defense, monitored by the Office of Naval Research under contract #N00014-87-K-0238, the Office of Naval Research, under a University Research Initiative Grant, Contract #N00014-86-K-0764 and NSF Presidential Young Investigators Award NSFIST-8351863.
** Authors' address: S. Wermter and W.G. Lehnert, Dept. of Computer and Information Science, University of Massachusetts, Amherst, MA 01003, USA.

conceptual QA-systems, and logic-based QA-systems. This classification considers both the historical development and the basic representation mechanisms of QA-systems.

Many of the earliest QA-systems were peripheral modules designed to support natural language access to a database. Natural language questions were highly restricted in their grammatical constructions and their vocabulary so they could be translated into a series of database queries using specific translation procedures. These procedures were associated with specific natural language constructs and determined the semantics of a word, a phrase, or a whole question. The technique of translating natural language constructs into database queries with associated procedures is called procedural semantics and the systems using this technique are classified as *procedural QA-systems*.

The second class of QA-systems, *conceptual QA-systems*, is more specifically tailored for natural language purposes and is centered around semantic primitives for natural language. Different sets of primitives were developed for different purposes, but all of them tried to reduce the number of possible surface structures in natural language by translating the natural language construct into a canonical meaning level represented by a few primitives. These well-defined primitives could then be used to build more complex memory structures and concepts. The best known theories in this class are preference semantics (Wilks (1975)) and conceptual dependency (Schank and Colby (1973), Schank and Riesbeck (1981)).

While the early QA-systems centered around the ideas of procedural semantics, other declarative representations occurred in question answering at about the same time as the conceptual QA-systems. One important branch of declarative representations is logic. In procedural QA-systems the representation of the domain knowledge in the database is distinct from the representation of the translation process of the natural language question. However, *logic-based systems* can use logic-based mechanisms for both the representation of domain knowledge and the representation of translation processes.

2. Human and computational question understanding

A lot of problems at different levels are involved in the process of understanding questions. Before we look at several systems in more detail, we will now consider some general problems of question answering to get a feeling for the underlying processes of man and machine. One way to see these differences is to understand when questions are similar.

Similar questions can be defined in a variety of ways. In one approach, questions are considered to be similar if they have the same answer. A simple example shows that this definition alone is not adequate. 'Who gave the capital of the United States its name?' and 'Who was the first American

President?' can be answered in the same way by saying 'Washington', but the questions are obviously very different. One reason why we consider these questions to be different questions are the different access paths used to answer them. While the first question asks for a person whose name was used for a city, the second question asks for a person who held a public office.

These simple examples suggest that similar questions are questions which can have the same answer and use the same access paths to determine that answer. This understanding of similar questions is still rather abstract and leaves a lot of room for more specific interpretations. While humans do not have to think about these differences in answering questions, computational models have to make these differences explicit. Classes of questions depend on the task and the domain, so we see different models relying on different classes of questions: Ram (1987) identifies 5 classes and Lehnert (1978) uses 13 different classes of questions.

To understand the differences between these classifications, we have to look at some of the motives behind them. Questions can be differentiated with respect to the effort necessary for understanding them. A general rule of thumb is that as the effort gets bigger, more semantics must be involved in the understanding process.

To illustrate this point, we will give some examples of questions that are similar in the sense of having the same possible answer and the same access path. For example, consider the use of a syntactic transformation like wh-movement applied to the question word 'who': 'Who is the champion?' versus 'The champion is who?'. Moving more into the direction of semantic issues, substitutions could describe similar questions: 'Who is the champion?' versus 'What is the name of the champion?'. A substitution by synonym can be found in the example: 'Who is the coach?' versus 'Who is the trainer?'. A substitution by compound synonym is: 'Where do the old men and old women live?' versus 'Where do the senior citizens live?'. A substitution by metaphor is: 'Who is the iron fist?' versus 'Who is Joe Frazier?'. Even this small number of examples shows how difficult it is to understand questions and to paraphrase them in a form which is understandable for a machine.

In addition to problems with questions, there are lots of problems with answer generation as well. Often questions have many answers and humans (or systems) have to choose which possibility is most appropriate in a given situation. A simple example could be the question: 'Where is Tom?'. Depending on the context of this question the answer could be 'in New York' or 'in graduate school'.

The computational mechanisms making these decisions have to rely on complex knowledge sources, such as representations of context or history lists for keeping track of the known facts in a communication process. Furthermore, representations of the user's assumed knowledge, representations of plans, and rules of cooperative behavior play an important role in the man–machine communication. Another important issue is the linguistic power of the answer generation module: which grammatical constructions are possible, and how variable is the generated language? Models have been built to demonstrate how some of these knowledge sources could lead to systems which exhibit different forms of cooperative behavior such as justifications, clarifications, corrections of user misconceptions, and rejections of questions. These aspects of cooperative behavior and variability will be examined in more detail below.

3. Procedural QA-systems

One of the earliest attempts to model language comprehension was the procedural QA-system LUNAR (Woods (1978)). This system, designed in the early seventies, was intended to support geologists searching a database for information on moon rock samples collected during the Apollo missions.

Although other natural language interfaces had already been developed at that time, LUNAR can be regarded as a milestone, insofar as it combined the technique of procedural semantics with a 'real world' application. Three reasons led to this development: (1) the physical domain was restricted enough to experiment with its semantic relations, (2) a NASA database existed with a reasonable number of 13,000 entries of moon rock analyses, and (3) the theory of procedural semantics developed in Woods' thesis seemed to supply a suitable framework for the problem. Because LUNAR can be regarded as a prototype for procedural QA-systems, we will concentrate on LUNAR in this section.

The task of LUNAR was to answer requests like: 'What is the average concentration of olivine in breccias?' or 'Analyses of strontium in plagioclase' (Tennant (1981)). These natural language questions had to be mapped into database queries. The database had a very simple structure consisting of one table of seven fields for the following categories: (1) sample identifier, (2) phase of the sample, (3) chemical constituents, (4) concentrations of the constituents, (5) units of measure, (6) source identification number, and (7) a tag field for identification purposes.

The translation of a natural language question into the database query was accomplished by procedural semantics. Procedural semantics can be characterized as an interpretation method which associates a semantics for words by attaching specific procedures to the words. These procedures then transform the natural language question into a format which can be understood by the database.

To make this abstract idea clearer, let us have a closer look at a representation in LUNAR. The basis of the representation was a logic-like meaning representation language. This language specified what the system was able to understand. The main components were designators to name objects, propositions that could be tested in the database, and commands that carried out actions. A simple example for this language would be (Woods (1978)):

```
(and (contain x3 oliv) (not (contain x3 plag)))
```

This example can be interpreted declaratively as an assertion which states that x3 contains olivine but does not contain plagioclase. In this case, 'x3' 'oliv' and 'plag' are designators, 'contain', 'and', and 'not' are propositions. An example of a command can be found in the following example, which tests to see if object S10046 contains olivine.

```
(test (contain S10046 oliv))
```

LUNAR's model of procedural semantics was divided into 3 parts. First, the natural language question was translated into a syntactic parse tree. Second, pattern action rules worked on the parse trees and transformed them into a meaning representation language. The action part of the pattern action rule transformed the parse tree only if the pattern part of the rule was fulfilled. Pattern action rules solved some of the ambiguities in modifier placement and quantification. The third phase, the execution of the translated query, determined the answers for the translated query.

This model of procedural semantics was useful for LUNAR because of its restricted domain and the assumption that every element in the database belonged to exactly one of the seven categories mentioned above. This radical use of the categories made it possible to avoid ambiguities on the meaning representation level. But using an unambiguous representation language assumes that pattern action rules translate ambiguous questions into an unambiguous internal form. In general, it may be difficult to formulate these pattern action rules.

Pattern action rules are used in LUNAR to restrict the number of possible parses for prepositional phrase attachment and conjunction scoping. In general, there are many different constituents in a sentence to which a prepositional phrase can refer. For instance, in the example 'the picture of the girl with the pony-tail', the word 'pony-tail' can only refer to the girl and not to the picture. But in an example like 'the picture of the girl with the white margin', the white margin would be attached to the picture and not to the girl. Conjunction scoping is also related to referential ambiguity. In the example 'boys and girls with fever', we can not say whether boys who have no fever are included in this descriptor or not.

With both prepositional and conjunctional attachment problems, all possible parses are initially constructed. Then, pattern action rules act as heuristics which filter out some of the less plausible parse trees. Looking more closely at the prepositional phrase attachment, the parser uses the technique of 'selective modifier placement'. This technique tries to attach a found constituent to known constituents by looking up the known constituents from a stack. Based on the list of syntactically possible attachment points for known constituents, semantic knowledge is used to evaluate the constituents and choose a preferred attachment for the current modifier. In this process the known constituents are semantically distinguished into three different classes: constituents which forbid, require, or allow the current modifier.

This technique of 'selective modifier placement' can be classified as an incremental process, interleaving syntactic and semantic processing. Whenever a syntactic constituent is found, the semantic component takes over and tries to attach to the best possible known constituent.

While the framework of LUNAR was found to be successful as a prototype, it was never applied by NASA geologists in real operational use. Some informal tests done in 1971 showed that the system could process about 78% of the queries of geologists. 12% of the queries failed but could be handled after simple system modifications and 10% failed as a result of deeper syntactic or semantic flaws.

4. Conceptual QA-systems

Although LUNAR demonstrated its usefulness as a QA-system, there were some serious drawbacks. First, the system used a static representation relying heavily on the predefined table structure of the database. All questions had to be answered in terms of the 7 categories in this table. Second, the system relied on procedural semantics which amounted to many specific procedures for handling specific questions. These two features, static underlying memory and specific procedures, resulted in a design that was especially tailored for the task of question answering, but that could not be extended in an easy way. The lack of sophisticated knowledge structures and the lack of independent procedures led to shallow understanding and little flexibility.

To gain a *deeper level of understanding*, more knowledge had to be incorporated in QA-systems. QA-systems which relied on different knowledge structures were developed in the second half of the seventies and early eighties. In an effort to demonstrate the importance of knowledge structures, these *conceptual QA-systems* read short stories, built a representation for those stories dynamically, and answered questions about the stories.

Conceptual question answering is motivated by two goals: we would like to understand human question answering as an information processing phenomenon, and we would like to simulate these human abilities on a computer. This change in perspective shifts us away from database queries to text understanding, and from simple data structures to complex knowledge structures.

A system expected to reach this level of advanced understanding has to rely on sophisticated meaning representations. We need meaning representations that can be automatically derived from the source text and that are also suitable for supporting memory processes like inference generation. A number of systems were developed using this very general approach. Winograd's SHRDLU used meaning primitives to describe a blocks world for a robot simulation (Winograd (1972)). Wilks used meaning primitives along with preference semantics to build machine translation systems (Wilks (1975)) and Schank used meaning primitives to model language based on human actions (Schank et al. (1973)). Because our focus here is on question answering, we will not describe these systems any further, but concentrate instead on some QA-systems which relied on conceptual dependency for their meaning representations (Schank and Colby (1973), Schank and Riesbeck (1981)). Conceptual dependency originally consisted of 11 meaning primitives for basic events, such as an abstract transfer or a physical transfer. The primitives for these events specified appropriate case frames with different slots for the semantic constituents corresponding to an agent, recipient, object, source, or destination of an event. By filling these slots with specific instantiations it was possible to represent complex events in terms of a conceptual dependency graph.

The earliest system to use conceptual dependency for connected text (SAM) read simple stories about everyday events, like events in a restaurant or in a train (Cullingford (1978)). These stories were analyzed by using specific expectation-driven knowledge structures called scripts. Scripts enabled the system to understand events and make simple inferences about many events that were not explicitly mentioned. Scripts were constructed out of networks of conceptual dependency graphs and encoded information about stereotypical situations. Because only a part of the knowledge used to understand stories can be characterized as stereotypical, other more general memory structures were also created. For instance, plans were used in addition to scripts to overcome weaknesses of the stereotypical knowledge (Wilensky (1978)). Plans are more general memory structures which cover situations that are not as stereotypical as 'going to a restaurant' or 'going to the movies'. Plans describe more general methods for attaining goals and can be applied in a variety of novel circumstances.

Although scripts by themselves cannot account for all the inferences needed in general, scripts did support text comprehension and a wide range of text-oriented question answering. One QA-system which relied primarily on scripts was QUALM (Lehnert (1978)). QUALM knew about scripts, about specific types of questions, and about searching memory for answers. In this sense, QUALM was a much more sophisticated question answerer than LUNAR because it worked on a deeper meaning level. While LUNAR had to rely on knowledge hidden in its translation procedures, QUALM operated very differently. The process of transforming a sentence into an internal meaning representation was performed by an independent parser (Riesbeck and Schank (1976)). All the processes associated with question answering were therefore language-independent, and dependent only on the meaning representations. The same question answering procedures could be applied to stories read in English, German or any other language handled by the parser, without any modification to accommodate a new language. After QUALM generated an answer based on information in the story representation, the resulting conceptual dependency graph was then translated back into natural language by a natural language generator (Goldman (1975)).

Using knowledge about stereotypical events associated with restaurants, QUALM could answer questions like (Lehnert (1977)):

Why did John go to New York?
Because John wanted to go to Leone's.

This 'why question' asks for the cause of the event 'going to New York', so it is a question about the motives or goals underlying the event.

Did anything unusual happen on the subway?
A thief picked John's pocket.

This 'occurrence question' asks for an unusual event in the context of a subway. Scriptal knowledge about subways is used to decide what is usual in the context of a subway and what is not. Many other question types could be answered with QUALM, including 'component questions' ('Who went to New York'), 'Yes or no questions' ('Did John eat lasagna?'), and 'How questions' ('How did John get to New York').

QUALM was a model of human question answering which endeavored to simulate the underlying strategies humans use. LUNAR was a more performance-oriented retrieval system which made no claims about cognitive validity. The answers of LUNAR were conceptually much simpler than the answers found by QUALM, but that did not matter as long as the users (in LUNAR's case the geologists) were satisfied with the answers. While QUALM's underlying theory and memory was more sophisticated, LUNAR addressed some problems in language analysis that QUALM's parser did not. For instance, prepositional phrase attachment was important in the technical domain of LUNAR, but could be ignored by SAM and PAM. Since complex noun phrases can easily be avoided in simple stories, verb-oriented sentence analysis may suffice for simple texts. Most importantly, we should note that QUALM separated the processes of question answering from the processes of sentence analysis in a way that LUNAR never could have. The emphasis for QUALM was therefore more on memory and conceptual representation than language analysis per se.

After QUALM, other QA modules were built for increasingly sophisticated text comprehension systems. FRUMP (DeJong (1979)) read newspaper stories in about 70 scriptal domains and IPP (Lebowitz (1983)) made generalizations based on stories about terrorism. Over several years, new memory structures were found to be useful in modelling human question answering. The most complex system along these lines was the BORIS system (Lehnert et al. (1983)). BORIS demonstrated what was probably the most comprehensive question answering capability ever reached. On the other hand, BORIS relied on a difficult integration of 22 kinds of different memory structures and needed an enormous amount of knowledge to process and understand the story.

5. Logic-based QA-systems

At the same time that conceptual question answering was focussing on simulating human processes of question understanding, another branch of question answering began to receive attention within the AI community. Three motivations inspired the development of logic-based QA-systems.

First, procedural QA-systems like LUNAR had demonstrated a high performance for their specific application area, but had also demonstrated difficulties in transferring the database interface to a new domain or new task. This lack of flexibility led to a shift in attention from the procedural representations of semantic grammars to more declarative mechanisms. These declarative mechanisms were expected to make QA-systems more general and easier to transfer across different domains.

Second, in the mid-seventies, mathematical logic was being transformed into a programming language with a clear declarative semantics. While logic had had some influence in procedural systems, logic itself had not been used as an executable language. With the development of logic as a programming language, predicates could be formulated in first order predicate calculus (FOPC) or subsets of FOPC like Horn clauses.

Third, by now it was clear that the ability to make inferences was essential for any natural language processing system. While conceptual question answering systems covered complex inferences and expectations in the form of diverse memory structures, logic oriented question answering concentrated on inferences as simple deductions. By encoding relevant knowledge in logic, the task of making inferences was reduced to the problem of finding a proof in a deductive database. Although automatic theorem proving was no trivial matter, it was an established research area with a community of active investigators.

With this background let us now have a look at a representative logic system. CHAT80 (Warren and Pereira (1982)) is a QA-system for world geography with knowledge that is encoded in a subset of FOPC. More specifically, it relies on predicates in Horn clause form and is realized in the logic programming language PROLOG.

Three major processing modules can be identified in CHAT80: (1) a syntactic parsing module builds a syntactic parse tree out of the natural language question; (2) a semantic interpreter translates the parse tree into a logical form; (3) any problems of quantifier scoping ('every', 'each') are tackled by a heuristic algorithm.

Example: (Warren and Pereira (1982))
How many countries export oil?
answer(N) ⇐ numberof (X, country(X) & exports (X,oil,N))

Not only are the facts of the database expressed in PROLOG, but the processes of syntactic and semantic analysis ((1) and (2) above) are also realized by declarative PROLOG statements. This means that the task of question answering can be effectively reduced to a proof procedure for establishing the existence of data in the database. However, there is a tradeoff here between this uniform representation for different forms of knowledge and a lack of methods for influencing control of the proof procedure. In addition to this control issue, another representational problem is important to mention: CHAT80 relies on the closed world assumption. This assumption means that all facts which are not present in the database are assumed to be false. This leads to consistency problems in incomplete domains because missing facts which were assumed to be false might be later included into the database as true facts.

One of the advantages of a declarative representation like logic is a clean interface for the access of other systems. Although CHAT80 works with a database of PROLOG clauses, it is relatively easy to attach a relational database and obtain a deductive relational database. Furthermore, a change of the domain is not so difficult as in the procedural approach, because the knowledge does not have to be buried deep in the code of procedures. Transportability is one of the advantages of CHAT80 and logic-based QA-systems in general.

In conclusion, the restrictions on the control of the question answering process are the price to pay for the plain logic representation in logic-based QA-systems.

6. Noun phrase disambiguation for question understanding

Having described a gross outline of existing QA-systems we will now have a look at some problems which all QA-systems must confront. Although there are a lot of differences between QA-systems, especially if they belong to different classes, all of them have to handle noun phrases. Noun phrases may be one of the most important constituents in both questions and answers, because noun phrases often contain crucial information.

In general, noun phrases consist of nouns, adjectives, prepositions, determiners, and conjunctions. While adjectives modify nouns, prepositions and conjunctions build relationships between nouns. Within a noun phrase, ambiguities can occur in different ways. For instance, there are lexical ambiguities and syntactic ambiguities.

6.1. Lexical ambiguity in noun phrases

Lexical ambiguity is concerned with the meaning of words. Most words stand for more than one concept and so most words are ambiguous. In many cases the overall context or other words in the noun phrase can constrain the interpretation of an ambiguous word:

The bank with the highest interest rates
The bank with the white sand

In this example 'bank' is lexically ambiguous, because it could refer to a financial institution or the land next to a river. This lexical ambiguity can not be resolved in the absence of context, but references to interest rates or sand tell us which word sense must be intended.

6.2. Syntactic ambiguity in noun phrases

Syntactic ambiguities within noun phrases appear in noun phrases that contain prepositional phrases or conjunctions. A prepositional phrase consists of a preposition followed by another noun phrase. As we saw with lexical ambiguities, some syntactic ambiguities can be resolved using the meaning of other words in the noun phrase.

The picture of the girl with the hat
The referee of the match with the whistle

We can understand these examples because we use underlying knowledge about the relationships between the nouns. A picture wearing a hat or a match with a whistle make little sense. These syntactic ambiguities are therefore ruled out by likely relationships between nouns. Knowledge about likely relationships is typically encoded using selectional restrictions. For example, 'X with Y' is likely if X is human and Y is an article of clothing. Ambiguous prepositional attachments as illustrated above cannot always be resolved, as the following well-known example shows:

The man on the hill with the telescope

Ambiguities which cannot be resolved on the basis of selectional restrictions can only be resolved if additional context is available. If the context sets us up with feelings of being watched, then the telescope probably belongs to the man. If the context is concerned with meeting someone at a specific place, then the telescope probably belongs to the hill.

While the attachment of constituents plays an important role in the resolution of syntactic ambiguities in prepositional phrases, prepositional phrase attachment is not the only source of syntactic ambiguity in noun phrases. Ambiguities caused by the use of conjunctions are similar to problems in prepositional phrase attachment in the sense that the relationships between different constituents must be considered.

Smog in New York and Boston
Smog in New York and rain

These examples show why conjunctional ambiguities involve attachment problems: while Boston should be interpreted as a companion to New York, rain should be understood as addition to smog. From a purely syntactic point of view, the words 'Boston' and 'rain' can not be differentiated. At the same time, conjunctional ambiguities are different from prepositional ambiguities because conjunctional ambiguities can be caused by missing constituents such as adjectives or prepositions. The following example shows how an additional preposition can be used to prevent unintended interpretations.

Smog in New York and in Boston

Now Boston is no longer an independent constituent and it is clear that New York and Boston have to be attached to smog in the same way.

These conjunctional ambiguities arise because of missing prepositions. In such cases, underlying word meanings support the right attachment decisions. Sometimes common semantic features influence the interpretation (Boston and New York). Sometimes the semantic features of a constituent prevent the incorporation into a prepositional phrase (smog in New York and in rain). One reasonable rule of thumb is that the attachment of a preposition can be assumed to be local and is carried over conjunctions only if the candidate constituent has common semantic features and does not have constraints which prevent the attachment of the preposition.

Another form of conjunctional ambiguity involves the uncertain attachment of adjectives.

Cold beer and lemonade
Cold beer and coffee

Seen from a purely syntactic point of view we cannot decide if cold refers only to beer or if cold refers to lemonade as well. In the second example we understand that cold only refers to beer, because coffee is usually hot or warm.

6.3. Noun phrases and semantic similarity

At the beginning of this article we considered the problem of defining similarity across questions. Now we shall discuss the basis for semantic similarity over noun phrases. Semantic features are typically used to characterize the semantic contents of nouns. For example 'physical' is a semantic feature of the word 'house'. A simple measure of noun phrase similarity can therefore be based on a comparison of their semantic features. However, it is not enough to simply count the number of overlapping semantic features because not all features have the same importance.

To weigh the different semantic features of a noun phrase, we can divide the noun phrase into different parts, allowing us to emphasize some semantic features over others. To begin, we will decompose the noun phrase into its central part and its attributive part. We then define the *central part of a noun phrase* to be the head noun with a directly modifying constituent, while the *attributive part of a noun phrase* includes any prepositional constituents. By giving the central part of a noun phrase more weight than the attributive part, we can emphasize the semantic features of the main central concept over the semantic features of its attributes.

To better understand this idea of central and attributive noun phrase components we will consider some examples:

A house with a window
A house with a garden
A house and a garden

In the first two examples 'a house' is the central part while the prepositional phrases 'with a window' and 'with a garden' are the attributive parts. A conjunction separates two distinct central parts in the third example.

Unfortunately, it is not always so easy to distinguish the central and attributive parts of a noun phrase. Syntactic ambiguities often force us to consider the semantic features of the candidate constituents, as the following examples illustrate:

Strength of arms and legs
Strength of character and aggressiveness

In this example semantic knowledge is needed to decide that 'strength' and 'strength and aggressiveness' are central, while 'of arms and legs' and 'of character' are attributive.

This notion of central and attributive noun phrase components as the basis for a similarity measure for noun phrases has advantages in the case of compound nouns as well.

Lock of the room of the printer
Lock of the printer room
Printer room lock

The central part of the first two examples is the lock modified by the attributive part of the printer room. The central part of the third example is the whole compound noun 'printer room lock'. Taking advantage of the heuristics that the most important conceptual knowledge in a compound noun is at the end, we could compare the semantic features of the last noun in the compound noun with the semantic features of the central part of the noun phrase. This comparison can help us decide whether a noun phrase with prepositional components and a compound noun are similar.

To conclude this section on the similarity of noun phrases we will point out one remaining problem: the focus of the context. Finding a measure of similarity is a very hard problem, because it depends on the underlying features, the relationships between these features, and the general context. 'Sleep' and 'Unconsciousness' can be regarded as similar from the perspective of states, but they are not similar from the perspective of symptoms. Therefore, physicians can understand the noun phrase 'pain during sleep and unconsciousness' as the two different symptoms pain and unconsciousness, while persons without a medical background might think of one symptom during two different states. Therefore, similarity within noun phrases depends not only on semantic features and a distinction of central or attributive components, but the domain context as well.

6.4. Interaction of noun phrases within questions

In previous sections we demonstrated how the resolution of certain lexical and syntactic ambiguities demands semantic knowledge about words or contextual knowledge. Semantic and contextual knowledge is also necessary for the larger task of question answering since one cannot hope to answer a question correctly if the question itself has been misunderstood. Because noun phrases are so essential for questions, their disambiguation is an important part of the understanding process. Issues like the similarity of the meaning of questions and the generation of answers can only be attacked if noun phrases are understood. Consider the following two examples:

How long is the bridge?
How long is the match?

Since the concept of a bridge has a distance feature, but no time feature describing a duration, the noun phrase 'bridge' influences the interpretation of 'how long' so that we understand this to be a question about distances. On the other hand, one expects that the concept of a match (in the word sense of a game) has a time feature, but probably no feature for spatial distances. This forces us to interpret 'how long' in terms of a time measurement. This is another example of how potentially ambiguous words can disambiguate other words or phrases.

Another disambiguation problem arises when we must determine which constituents in a question belong to the noun phrase and which constituents belong to the verb. Compare the following questions:

Did Tom eat the steak with the knife?
Did Tom eat the steak with the teacher?
Did Tom eat the steak with barbeque sauce?

In the first two examples, 'with the knife' and 'with the teacher' refer to the verb 'eat', while 'with barbeque sauce' refers to the noun phrase 'steak'.

The described examples demonstrated the importance of interactions between noun phrases and other constituents in a question. Interrogative constituents ('how long') or verbs ('eat') interact differently with noun phrases depending on the meaning of the noun phrase. Questions cannot be understood without understanding the interactions of the noun phrases within questions.

7. Surface structure variability for answer generation

Our last chapter described some of the problems in understanding noun phrases. We saw how some noun phrases can only be understood if we know how different constituents relate to one another. The use of semantic, syntactic, and contextual knowledge makes it possible to disambiguate some of the ambiguous relationships that arise. While noun phrase disambiguation is frequently approached as a problem in language analysis, we will now turn our attention to language generation as it applies to finding answers for questions.

The issue of answer generation has been relatively neglected within AI circles when compared with the amount of activity devoted to understanding processes. But answer generation is slowly gaining more attention because interactive systems are of great interest, and because language generation is a good proving ground for planning and memory interactions.

7.1. General aspects of language generation

The complexity of the generation process depends on the task involved. Sometimes it is enough just to answer a question with a number (How many children do you have?), but for other questions the answer may require a whole sequence of sentences (What can you do to ensure quality education for your children?). Although the character of these questions is rather different, they have several things in common.

First, for each question we must decide *what* information is needed to answer the question. This decision is determined by the domain-dependent part of the QA-system which has access to the facts of the domain and how they are related. We must access this knowledge base, find the facts relevant for the current question, and retrieve these facts. Different kinds of indices can support the search for facts, while different heuristics may be needed to select the most appropriate facts out of those that were indexed. This selection process is especially important in cases where a lot of information has been found. For example, in answering 'Which animals have four legs?', we probably do not want to return a list of hundreds of animals. A better answer might identify the total number of animals the system could return along with a few examples or general classes. Preferences about what to generate depend on the domain and on the expectations of the user. Heuristics influencing generation have to take these considerations into account.

Once we have decided what information to use in answering a question, we must decide *when* different parts of the answer should be generated. Decisions have to be made about how to organize retrieved facts. If a question must be answered with long sentences or several sentences, temporal and causal dependencies between clauses have to be taken into account.

Finally, we must specify *how* to present our information. A flight booking system might contain knowledge about strategies for how to convince people to book a flight. A medical consultant system might have strategies for how to make the facts sound like suggestions. Many of these high level decisions about how to generate an answer are dependent on specific task orientations like sales or advice giving. The more that is known about the task and the user, the easier it is to build a sophisticated generation model.

Although the generation process was described as the three separate modules addressing 'what', 'when', and 'how', these three sets of factors must interact with one another. For instance, a flight booking system might not be smart enough to offer the cheapest seats first. In this case, the selling task influences not only the 'how', but the 'what' and 'when' as well.

7.2. Surface structure variability in the generation process

The previous section characterized the issue of how to present answers as a general problem for every QA-system. For every answer, a QA-system must decide which words should be used and how they should be put together in the answer. Usually there are several ways to express the same answer to a question. This raises the important issue of surface structure variability.

Surface structure variability comprises general strategies for how to generate different surface structures in an answer for a question. Variability in the surface structures for an answer requires more than fixed templates and canned answers. Therefore variability is an important issue for QA-systems which need a high degree of flexibility in their answers. For example, active and passive surface structures can usually be considered as one form of variability.

The stewardess served the meal.
The meal was served by the stewardess.

Both the active and the passive form are based on the same event description in which a meal is transferred from a stewardess to an unspecified airline passenger.

Serving-event:
  Actor: stewardess
  Object: meal

Another aspect of variability determines the position of constituents in a sentence. Although one could claim that different positions imply a different emphasis, the following sentences essentially contain the same information:

The plane will arrive at 8:00 pm.
At 8:00 pm the plane will arrive.

While the examples for variability we have seen so far are syntactic, semantic variability depends more on the domain. For instance, verbs which might be considered as synonyms in one domain might not be synonyms in a different domain. Consider the following examples:

The patient's skin was yellow.
The patient's skin looked yellow.

Surgeons do not differentiate between the verb forms 'to be' and 'to look' in the context of medical histories. On the other hand, for psychiatrists there is a big difference between a patient who was afraid or just looked afraid. These examples demonstrate that strategies of semantic variability have to consider the domain. Other generation strategies for variability could rely on knowledge about discourse. These strategies are more domain dependent and try to simulate different models for user and machine.

8. Cooperative behavior: A user-oriented problem

The last two sections described two problems for the analysis of questions and the generation of answers: noun phrase disambiguation and variability. Both of these problems need syntactic and semantic knowledge to understand ambiguous structures and generate variable surface structures. These two problems are language-oriented problems which will not vary across different users. In this chapter we look at cooperative behavior as an example of a user-oriented problem. Language is important for cooperative behavior as well, but the demands of the user are the primary concern in creating a cooperative computational model.

Cooperative answers are not just correct answers, as the following example will illustrate:

Q: Can you tell me what time it is?
A: Yes.

What we need here are representations that allow us to predict what kind of information the user is looking for. These issues of cooperative behavior of QA-systems have received more attention recently trying to provide more useful and cooperative answers (Gal and Minker (1985), Grosz et al. (1987), Kaplan (1982), McKeown (1983)).

8.1. Failed presuppositions

Cooperative behavior for answer generation is closely connected to the understanding process because cooperative behavior depends on the questions. Every question contains implicit suppositions which the question answerer has to know. These presuppositions give hints about what is probably known and what to focus on in the answer.

Q: When did Tom go to school this morning?

Implicit in this question are many presuppositions. For example, we are referring to a person named Tom, Tom goes to school, and this normally happens in the morning. If one of these presuppositions is not fulfilled, we could simply respond that we do not know. But a more cooperative system would try to identify the incorrect presupposition and come up with a response designed to overcome the misunderstanding. The following examples show how a system could try to identify a suitable answer depending on which part of the presuppositions was wrong:

A: Tom hasn't left yet.
A: Tom didn't go, but Jim did.

A: Tom didn't go to school, he went to church.
A: Tom didn't go this morning, but he will this afternoon.

In all of these cases we detect a user's misconception and try to correct it by augmenting the answer with information that contradicts a faulty presupposition.

8.2. Anticipating follow-up questions

While the last section focused on how to augment negative answers, this section looks at the augmentation of positive answers. In many question-answering situations the questioner is not only interested in the positive answer 'yes', but is implicitly asking for more detailed information.

Q: Do you have any single rooms?
A: Yes, two.

Once again we see how the answer 'yes' might not be enough. To prevent the system from understanding a user question too literally and thereby force the user to ask follow-up questions, the system should anticipate follow-up questions. But it is difficult to predict what a user wants without some knowledge of the user's goals.

In relatively closed domains it is possible to assume a small set of user goals to aid answer generation. For example, in domains like hotel or airline reservations, a lot of questions occur so frequently that it is possible to infer what the user wants to know with a strong certainty. In more complex domains the issues of user modeling are much more complicated.

8.3. Explanatory questions

As we just saw, an accurate user model can be crucial for effective QA-dialogues. But not all questions require knowledge about a user's goals. For example, explanatory questions focus more on the world knowledge than on the user model. Explanatory questions are typically 'why' or 'how' questions and ask for an explanation. For example:

Why is the pressure in the container so high?
How did you come to this conclusion?

The first question is a question for an explanation from the domain, while the second question asks for an explanation from the system, how it came up with a specific answer. These 'why' and 'how' questions serve to explain the underlying concepts of a system and are important because they determine the credibility of the whole system. Early efforts to model explanatory answers were often no more than a logical proof used in derivation of an assertion. Unfortunately, logical proofs are neither concise nor particularly intelligible for most users: a theory of explanations that is well suited for human interaction remains to be proposed.

9. Current trends in QA-systems

Recent developments in question answering reflect past experience with procedural, conceptual and logic-based question answering. We will now identify some trends in terms of uniform representations, logic representations and transportability.

9.1. Uniform representations

Experiences with the conceptual QA-system BORIS taught us that conceptual memory structures had to be more uniform with easier communication between these structures. One system that integrates more uniform representations is SCISOR (Rau (1987a,b)). SCISOR specializes in newspaper stories about corporate takeovers. In this system syntactic, semantic, and world knowledge are all represented using the same formalism, yielding advantages over multiple representations with respect to communication processes. Finding an answer to a question involves matching a graph for the question against graph structures in the memory.

9.2. Logic representations

There are several reasons why logic-based question answering is important at this time. By combining a relational database with a logic deductive component, we can generate inferences from facts in the database. It is also fairly easy to parse natural language questions into a logical formalism supported by a relational database. The availability of both natural language access and inference mechanisms leads to developments emphasizing the use of automated dictionaries. Specific logic-oriented grammars (e.g. lexical function grammars) are built to shift as much syntactic and semantic knowledge into the dictionary as possible. This leads to QA-systems which have not only domain knowledge in a database but most of the language knowledge as well.

9.3. Transportability

Different domains require different memory structures, but the linguistic component of a QA system should be independent from specific domain structures. Ideally, we should be able to move from one domain to another without changing the linguistic component used throughout. The philosophy behind transportable systems is to develop an independent linguistic component, so that interfaces can be designed more efficiently and effectively.

Several systems have recently been developed with this focus on transportability. Using an intermediate description language, these systems translate the data formats of different databases into the same representation and thereby gain independence from specific databases. The long term goal is to achieve general natural language processors which can be adapted to a specific domain without building a completely new interface for every new application. Two of the most ambitious projects in this direction are TEAM (Grosz et al. (1987)) and TELI (Ballard and Stumberger (1986)), which attempt to supply a system designer with a toolbox of functions for constructing a natural language interface.

10. Conclusions

This paper introduced the basic approaches of QA-systems and classified them into procedural, conceptual and logic-based systems in accordance with the form of knowledge representation used. Although these representations led to different classes of QA-systems, we observe a common trend for these systems: the use of uniform and modular knowledge. Conceptual QA-systems use more uniform and more modularized memory structures and relationships. Procedural and logic-based systems rely more on uniform logic representations and modular automated dictionaries. In spite of these promising approaches in terms of knowledge representation, issues like noun phrase disambiguation and user modeling remain a field for further research.

References

Ballard B.W. and D.E. Stumberger, 1986. Semantic acquisition in TELI: A transportable, user-customized natural language processor. Proceedings of the Association for Computational Linguistics, 20–29.

Cullingford, R.E., 1978. Script application: Computer understanding of newspaper stories. Ph.D. Dissertation, Research Report 116, Computer Science Department, Yale University.

DeJong, G.F., 1979. Skimming stories in real time: An experiment in integrated understanding. Research Report 158, Department of Computer Science, Yale University.

Gal, A. and J. Minker, 1985. A natural language database interface that provides cooperative answers. Proceedings of the 2nd Annual Conference on Artificial Intelligence Applications, 352–357.

Goldman, N., 1975. Conceptual generation. In: R.C. Schank (ed.), Conceptual information processing, 289–371. Amsterdam: North-Holland.

Grosz, B.J., D.E. Appelt, P.A. Martin and F.C.N. Pereira, 1987. TEAM: An experiment in the design of transportable natural-language interfaces. Artificial Intelligence 32, 173–243.

Jin, W. and R.F. Simmons, 1987. Question answering with rhetorical relations. Proceedings of the Third Conference on Artificial Intelligence Applications.

Kaplan, S.J., 1982. Cooperative responses from a portable natural language query system. Artificial Intelligence 19, 168–187.

Lebowitz, M., 1983. Generalization from natural language text. Cognitive Science 7, 1–40.

Lehnert, W.G., 1977. Human and computational question answering. Cognitive Science 1, 47–73.

Lehnert, W.G., 1978. The process of question answering. Hillsdale, NJ: Erlbaum.

Lehnert, W.G., M.G. Dyer, P.N. Johnson, C.J. Yang and S. Harley, 1983. BORIS – An Experiment in in-depth understanding of narratives. Artificial Intelligence 20, 15–62.

McKeown, K.R., 1983. Paraphrasing questions using given and new information. American Journal of Computational Linguistics 9, 1–10.

Pollack, M.E. and F.C.N. Pereira, 1988. An integrated framework for semantic and pragmatic interpretation. Proceedings of the Association for Computational Linguistics, 75–86.

Ramp A., 1987. AQUA; Asking questions and understanding answers. Proceedings of the AAAI, 312–316.

Rau, L.F., 1987a. Knowledge organization and access in a conceptual information system. Information Processing and Management 23, 269–283.

Rau, L.F., 1987b. Information retrieval from never-ending stories. Proceedings of the AAAI, 317–321.

Riesbeck, C. and R.C. Schank, 1976. Comprehension by computer: Expectation-based analysis of sentences in context. Research report 78, Computer Science Department, Yale University.

Schank, R.C. and K.M. Colby, 1973. Computer models of thought and language. San Francisco, CA: Freeman.

Schank, R.C. and C.K. Riesbeck (eds.), 1981. Inside computer understanding. Hillsdale, NJ: Erlbaum.

Schank, R.C., N. Goldman, C. Rieger and C. Riesbeck, 1973. Margie: Memory, analysis, response generation, and inference on English. Proceedings of the IJCAI, 255–261.

Tennant, H., 1981. Natural language processing. New York: PBI.

Warren, D.H.D. and F.C.N. Pereira, 1982. An efficient easily adaptable system for interpreting natural language queries. American Journal of computational Linguistics 8, 110–119.

Wilensky, R., 1978. Understanding goal-based stories. Ph.D. Dissertation, Research Report 140, computer Science Department, Yale University.

Wilks, Y., 1975. An intelligent analyzer and understander of English. Communications of the ACM 18, 264–274.

Winograd, T., 1972. Understanding natural language. New York: Academic Press.

Woods, W.A., 1978. Semantics and quantification in natural language question answering. In: M. Yovits (ed.), Advances in computers, Vol 17, 2–87. New York: Academic Press.