The Context Model: Language Understanding in Context*

Yigal Arens

Division of Computer Science
Department of EECS
University of California at Berkeley
Berkeley, CA 94720

1. Introduction

This paper describes the language understanding component of the Unix Consultant (UC) system being developed at the Berkeley Artificial Intelligence Research project. The purpose of UC is to hold a conversation with a naive user of the Unix operating system while he or she is working on the computer, answering questions and solving problems for the user. The system has several other components, including the common sense planner PANDORA (Faletti, 1982), and the plan understander PAMELA (Norvig, 1982).

Our natural language understanding system contains as a subpart the PHRAN phrasal analysis program (Wilensky and Arens, 1980a) (Wilensky and Arens, 1980b) (Arens, 1981). PHRAN's knowledge base consists of **Pattern-Concept Pairs** -- pairings of language structures with a conceptual representation of their meaning. It operates by matching the pattern parts of the pairs against the input and using the corresponding concept to describe its meaning.

The current system attempts to deal with the fact that PHRAN by itself unable to deal with reference, and cannot disambiguate unless the linguistic patterns used require a particular semantic interpretation of the words. In addition, we wish to account for the fact that the same utterance may be interpreted differently in different contexts. These inabilities on the part of PHRAN originate in the fact that PHRAN's knowledge is almost entirely of the *language*, as opposed to knowledge about the entire conversation, more general world knowledge, etc. Of course, in order to specify the patterns, PHRAN needs at least some information about the semantics of the words appearing in the sentences it analyzes, but this is limited to the semantic categories the objects described by the language belong to (e.g. Person, Vehicle) and a Conceptual Dependency representation (Schank, 1975) of the actions. In order to hold a meaningful and useful conversation, however, it is clear that such a system must go beyond the (almost) purely linguistic analysis of the sentence to include the effect and the interaction this analysis has on our model of the conversation and on our knowledge as a whole.

The system we are currently constructing has a single mechanism which addresses many of these problems, which we call the **Context Model**. The Context Model contains a record of knowledge relevant to the interpretation of the discourse, with associated levels of activation. There are rules governing how elements introduced into the Context Model are to influence it and the system's behavior.

PHRAN and the Context Model interact continually. PHRAN passes its limited interpretation of the input to the Context Model, and it in turn determines the focus of the conversation and uses it to resolve the meaning of ambiguous terms, of references, etc., and passes these back to PHRAN.

Although it too involves the use of spreading activation and associations among semantic structures for the purpose of understanding text, the Context Model differs substantially in scope from Quillian's work in TLC as described in (Quillian, 1969). TLC was concerned mainly with the determination of the conceptual representation of the input sentence, a task which is handled here mostly by the phrasal analyzer. The Context Model groups related entries in it and arrives at a notion of the situation being discussed. Alternative situations in which a concept may appear can be ignored, thus enabling the system to have a more directed spreading of activation.

(Grosz, 1980) develops in great detail a scheme for determining focus of a task oriented dialog and using it to resolve references. Grosz's system relies heavily on the inherent temporal structuring of the task -- whereas we are trying to develop a more general approach, independent of the type of subject matter discussed. Our system must have the ability to shift focus freely according to the user's input, including the ability to store and recall previous contexts into focus.

The resulting system is able to converse and answer questions, while allowing the user to move in a relatively free manner from one topic to another, as the next example illustrates.

1.1. Example

The exchange described below takes place with the UNIX Consultant (UC) system being constructed at Berkeley. The purpose of the system is to answer the questions of naive users of the UNIX operating system while they are using the computer. See (Wilensky, 1982).

[1] User: How do I print the file fetch.l on the line printer?
[2] UC: To print the file fetch.l on the line printer type 'lpr fetch.l'.

.
. (intervening commands and questions)
.

[3] User: Has the file fetch.l been printed yet?
[4] UC: The file fetch.l is in the line printer queue.
[5] User: How can I cancel it?
[6] UC: To remove the file fetch.l from the line printer queue you must type 'lprm arens'.

In this example the user first asks a question [1] and receives a reply from the system. Then come several other questions and answers, and then the second part of the example. The user asks another relatively straightforward question and then a more problematic one. In order to reply to the last question the system must find the referent of 'it'. The language used implies that this must be a command, but the command in question was issued long ago. The system is able to determine the meaning of [5] only because the context of [1] and [2] had been stored and so could be recalled upon the seeing of [3]. This example will be discussed in more detail in section 3.

2. The Context Model and Its Manipulation

The Context Model is in a constant state of flux. Entries representing the state of the conversation and the system's related knowledge and 'intentions' are continually being added, deleted, or are having their activation levels modified. As a result the same utterance may be interpreted in a different manner at different times. Following are short descriptions of the different elements of the system.

2.1. Entries

The Context Model consists of a collection of entries with associated levels of activation. These entries represent the system's interpretation of the ongoing conversation and its knowledge of related information. The activation level is an indication of the prominence of the information in the current conversational context, so that when interested in an entry of a certain type the

*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.

system will prefer a more highly activated one among all those that are appropriate.

There are various types of entries, and these are grouped into three general categories:

1) **Assertions** – statements of facts known to the system.
2) **Objects** – objects or events which the system has encountered and that may be referred to in the future.
3) **Intentions** –
   a) Entries representing information the system intends to transmit to the user (i.e. output) or other components of an understanding system (e.g. goal tracker, planner).
   b) Entries representing information the system intends to determine from its knowledge base.

2.2. Clusters

The entries in the Context Model are grouped into **clusters** representing situations, or associated pieces of knowledge. If any one member of a cluster is reenforced it will cause the rest of the members of the cluster to be reenforced too. In this manner inputs concerning a certain situation will continue reenforcing the same cluster of entries -- those corresponding to that particular situation. Thus the system arrives at a notion of the topic of the conversation which it uses to help it choose the appropriate interpretation of further inputs.

2.3. Reenforcement

When the parse of a new input is received from PHRAN the system inserts an appropriate entry into the Context Model. If there already exists an entry matching the one the system is adding then the activation levels of all entries in its cluster(s) are increased. The level of activation decays over time without reenforcement, and when it falls below a given threshold the item is removed.

2.4. Stored Clusters

Upon inserting a new item in the Context Model the system retrieves from a database of clusters all those that are indexed by the new item. Unification is done during retrieval and the entries in the additional clusters are also inserted into the Model, following the same procedure described here except that they are given a lesser activation. We thus both avoid loops and accommodate the intuition that the more intermediate steps are needed to associate one piece of knowledge with another the less the mention of one will remind the system of the other.

The system begins operation with a given indexed database of clusters, but clusters representing various stages of the conversation are continually added to it. In principle, this should be performed automatically when the system is cued by the conversation as to the shifting of topic, but currently the system user must instruct it do so. Upon receiving such an instruction, then, all but the least activated entries in the Context Model are stored as a cluster indexed by the most highly activated among them. This enables the system to 'recall' a situation later when presented with a related input.

2.5. Operations on Entries in the Context Model

After a new entry is made in the Context Model the process described above takes place and eventually the activation levels stabilize, with some of the items being deleted, perhaps. Then the system looks over each of the remaining entries and, if it is activated highly enough, performs the operation appropriate for its type. The allowed operations consist of the following:

1) Deleting an entry.
2) Adding another entry.
3) Transmitting a message to another component of the system (i.e. output to the user or data to another program, e.g. PANDORA (Faletti, 1982), for more processing)
4) As part of the UC system, getting information from the UNIX system directly (and inserting an entry corresponding to the result).

3. Details of the Example

In [1] the user asks a simple question. PHRAN analyzes the question and sends the Context Model a stream of entries to be inserted. Among them are the fact that 'fetch.l' is the name of a file, and that the user asked what is the plan for printing it on the line printer. The system records these facts in the Context Model. Indexed under the entry representing the user's desire to obtain a goal there is a cluster containing entries representing the system's intent to find a plan for the goal the user has and instructing the system to tell the user of this plan. This cluster is instantiated here with the goal being the particular goal expressed in the question. The entry expressing the system's need for a plan for the user's goal leads to the plan in question being introduced also. This happens because the system happens to already have this association stored. When the system looks over the entries in the Context Model and comes to the one concerning the need to find the plan in question it will check to see if an entry for such a plan already exists, and in our case it does. But if no plan were found, the system would insert a new entry into the Context representing its intent to pass the information about the user's request to the planner PANDORA (Faletti, 1982). PANDORA will in turn return the plan to be inserted in the Context Model.

So the system finds the plan (issuing the command above) and inserts a new entry instructing the system to output it to the user. And eventually that is done – hence [2].

The topic shifts and the previous context is stored (with the operator's aid, as mentioned above), indexed by the most highly activated entries, including the file name, the mention of the line printer, the event of printing the file, and the command issued.

In [3] and [4] we have an exchange similar to the previous one except that the system actually has to consult the operating system in order to find the answer to the question. There is one major addition however -- as a result of the existence of the new cluster described above, the system has all this extra information triggered and loaded into the Context Model. And this is what makes it possible for the system to determine the referent of 'it' in [5]. Several other commands were mentioned and executed more recently, but in the new cluster just loaded many entries match already existing ones causing all – including the command intended for cancellation – to be more highly activated.

4. Shortcomings

The system is not currently able to determine on its own that the topic has changed and that it must store the current context. In addition to linguistic cues, we should be able to use the Context Model too in order to help in such a determination, but this work has not been done yet.

When it is instructed to, the current system stores essentially a copy of the more highly activated elements of the Context Model when creating a new cluster. They are not assumed to have any particular structure or relations among them other than all being highly activated at the same time. This causes two problems:

1) As a result it is very difficult to generalize over such clusters (cf. Lebowitz, 1980). The system may at some point determine a plan for changing the ownership of a particular file, and store a cluster containing it. If it is faced with the need to change the ownership of another file, however, the system will not be able to use this information. In the example above this problem was not encountered because the clusters used were preprogrammed to include variables in place of particular files.
2) There is no way to compare two clusters and determine that in fact they are similar. Thus we may have many clusters indexed by a certain entry all of which actually describe essentially the same situation.

Another element missing from the system is a model of the user. Certain assumptions are made as to the knowledge the user has of the Unix operating system, but these are built in and cannot be modified according to past interactions. Constructing such a

model will probably require work beyond the scope of this project.

5. References

Arens, Y. (1981). Using Language and Context in the Analysis of Text. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C.

Faletti, J. (1982). PANDORA – A Program for Doing Commonsense Planning in Complex Situations. Submitted to *The Second Annual National Conference on Artificial Intelligence*, Pittsburgh.

Grosz, B., J. (1980). Focusing and Description in Natural Language Dialogues. In *Elements of Discourse Understanding: Proc. of a Workshop on Computational Aspects of Linguistic Structure and Discourse Setting*. A. K. Joshi, I. A. Sag, and B. L. Webber, eds. Cambridge University Press.

Lebowitz, M. (1980). Generalization and Memory in an Integrated Understanding System. Tech. Report 186, Yale University Department of Computer Science. Ph.D. Thesis.

Norvig, P. (1982). Integrating Frame-Based and Goal-Based Processing in a Story Understanding Program. Submitted to *The Second Annual National Conference on Artificial Intelligence*, Pittsburgh.

Quillian, M., R. (1969). The Teachable Language Comprehender: A Simulation Program and a Theory of Language. In *Communications of the ACM*, v.12, no.8.

Schank, R. C. (1975). *Conceptual Information Processing*. American Elsevier Publishing Company, Inc., New York.

Wilensky, R. (1982). Talking to UNIX in English: An Overview of UC. Submitted to *The Second Annual National Conference on Artificial Intelligence*, Pittsburgh.

Wilensky, R., and Arens, Y. (1980). PHRAN – a Knowledge-Based Natural Language Understander. In *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, Philadelphia.

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.

Wilensky, R., and Morgan, M. (1981). One Analyzer for Three Languages. University of California at Berkeley, Electronic Research Laboratory Memorandum No. UCB/ERL M81/67.