Views From a Kill

James H. Martin

Berkeley Artificial Intelligence Research
Computer Science Division
University of California, Berkeley

ABSTRACT

Metaphor is a problem for natural language knowledge acquisition systems. Experts will make utterances based upon domain metaphors which the acquisition system may not possess. An approach is presented which uses knowledge about previously understood metaphors to process new uses. This approach is contrasted with several formal proposals for metaphor understanding which do not use explicit knowledge about metaphors. A system for representing metaphorical knowledge, as part of a general knowledge representation language, has been built. A knowledge acquisition system, UCTeacher, is described which can process newly encountered metaphors using knowledge of the domain and explicit knowledge about how similar metaphors have been used before. A detailed example from the system is presented.

Introduction

Metaphors are a widespread phenomena in natural language. The vast majority of metaphors are conventional parts of the language and are easily understood. Explicit knowledge of conventional metaphors is what makes these metaphors so easy to understand. Metaphor becomes a problem only when the hearer does not already have knowledge of the underlying metaphor that the utterance is based on.

This is exactly the situation faced by UCTeacher. UCTeacher is a natural language knowledge acquisition system for the the UNIX Consultant (Wilensky 1984). UC is a knowledge based consultant system that answers users questions about the UNIX operating system. Experts on UNIX can use UCTeacher to give more knowledge to UC simply by telling it the new information in English. One major problem for UCTeacher is learning new extensions to known metaphors during the knowledge acquisition task. For more information on other aspects of knowledge acquisition and UCTeacher see (Martin, 1985).

Consider the following examples from the UNIX domain:

1) You can kill a process by typing '^C'.

2) You can get into lisp by typing 'lisp' to the shell.

3) To leave the mail program type 'exit'.

* This research was sponsored in part by the Defense Advance Research Projects Agency (DOD), Arpa Order No. 4031, Monitored by Naval Electronic System Command under Contract No. N00039-C-0235, and by a GTE Laboratories Fellowship.

4) Run a file through the spell program to check for spelling mistakes.

Each of these examples contains a metaphor which is a specialization of a very general metaphor as applied to the concept COMPUTER PROCESS: example one involves viewing a process as an active agent that can be killed, examples 2 and 3 involve the metaphor that an interactive computer process is an environment that one can enter and leave, example 4 is an instance of a conduit/pipe metaphor. Experts in the domain of interest will frequently use such metaphors when giving new information to the system. The problem faced by UCTeacher is to find a way to understand these utterances given the fact that it does not yet possess the metaphors underlying them.

Our Approach

Knowledge about previously understood conventional metaphors is used directly in learning the new use of an old metaphor. Take examples 2 and 3 from above. At first it is not clear how the terms 'get into' and 'leave' can be applied to UNIX programs. The system has no knowledge of the fact that programs can be thought of as environments. It is the fact that there is a general container/environment metaphor in English which has been conventionally used in a number of other ways that allows it to understand these new uses. The basic strategy will be to identify the metaphor being used and then try to find conventional uses of that metaphor that are similar to the current situation. UCTeacher then analogically maps one of these known uses to the current situation. This strategy will be effective to the extent that new uses of conventional metaphors are closely related to other previously understood uses.

Previous Work on Metaphor

There have been two major approaches to the metaphor problem by the AI community. The first approach views metaphors as analogies. The problem of understanding a metaphor seen as a problem of analogically mapping information from a source domain to a target domain. Winston (1980), Carbonell (1981) and Gentner (1983) have all proposed various mechanisms for deciding how to selectively map information. The second approach has been inspired by the work of Lakoff and Johnson (1980). They assert that much of ordinary language is based on a relatively small set of systematic underlying conceptual metaphors. In this view conventional metaphors are not simple idioms nor are they the result of analogical reasoning. They reflect a set of underlying knowledge structures that are structured using conceptual metaphors. Carbonell (1980) has made a proposal that direct mappings be used to represent this type of metaphorical knowledge. He further proposed that these direct mappings could be used for analyzing metaphors. Jacobs (1985) implemented a system using similar knowledge for the purpose of generating utterances containing conventional metaphors.

Representing Knowledge about Metaphors

The conventional metaphors of English are represented as structured mappings in an abstraction hierarchy. Included in this abstraction hierarchy are metaphors which are directly related to word senses. Consider the verb kill. The literal definition can be paraphrased as 'cause the death of a living thing'. Clearly the 'process' in example (1) does not fit neatly into this definition. Now consider the following examples:

5) The Mets killed the Dodgers.

6) The senate killed the immigration bill.

7) The Islanders killed the penalty against them.

8) He killed the conversation when he came into the room.

9) A holding penalty killed the 49er's drive.

10) My editor told me to kill the last three paragraphs of my story.

The above examples contain related senses of the verb kill each with a very specific meaning. Among these senses are termination, defeat and deletion. These senses are represented by specific mappings between the KILL concept and the target meaning. Each of these mappings is in turn an instance of an abstract metaphor. In the example section it will be shown how these mappings can be used to process example (1).

The knowledge representation language that is being used to represent these mappings is KODIAK (Wilensky 1984). KODIAK is an extended semantic network language in the spirit of KL-ONE (Brachman, 1979). A unique feature of KODIAK is the semantic relation called VIEW. A VIEW is structured association between two concepts that asserts that one concept can be thought of in terms of another concept without asserting that the two concepts are related via a more abstract category. VIEWs are the tool that are being used to represent metaphorical mappings.

Metaphor and Knowledge Acquisition

The task faced by the knowledge acquisition system is to process utterances like those in examples 1 through 4, given the fact that the system's knowledge of both the facts and metaphors of the domain is incomplete. UCTeacher upon encountering an unknown metaphor uses the hierarchy of abstract metaphors and specific instances of known metaphors to come to a correct construal of the utterance. In particular the hierarchy is used to suggest plausible mappings and specific instances of metaphors are used in the creation of new mappings and concepts.

Metaphor Extension Algorithm

A four stage process is used in order to come to a correct construal of an utterance containing a new metaphor.

- Exploit the metaphor abstraction hierarchy to limit the search for a new mapping.
- Examine specific metaphoric word senses from the current example that have the mapping found in the first stage as an ancestor in the hierarchy.
- Analogically map one of these senses to the current situation.
- Create new VIEWs to connect the new meaning to the literal meaning.

The first step restricts the search to the most specific metaphorical mapping that can account for the current problem. The second step finds other ways that this metaphor has been used previously. The third step maps the meaning of one the previous uses onto the current situation. Creating the views in the final step allows this metaphor to be preocessed directly in the future. The following section describes an example of this processing in the current UCTeacher system.

An Example from UCTeacher

Consider the following working example from the UCTeacher system:

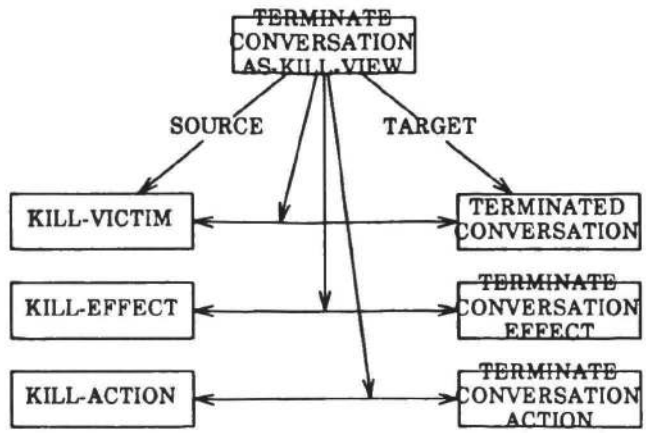
```
You can kill a process by typing '^C'.
```

The situation is such that the system has never heard the verb 'kill' applied to the concept UNIX-PROCESS. Moreover the UNIX-PROCESS concept is in direct violation of what the system knows can fill the role of the victim of a kill. In addition there are no known metaphorical VIEWs that could have been used to resolve this constraint violation. The system now attempts to create a new metaphorical mapping based on some old mapping in an attempt to resolve this constraint.

The first step performed by UCTeacher is to descend down the hierarchy to find as specific a mapping as possible that would cover this violation. Specifically it attempts to find a mapping with a source concept that covers the category living thing and a target category that allows a computer process. In this case the most specific VIEW that covers this is a VIEW from abstract entities to 'person'. Note that there are more specific VIEW's below this one but they violate the constraint on the viewed-thing being a computer-process, indicating that there are no known personifications of this concept as yet. The view that was found, PERSONIFICATION, will be used to guide the search for candidate VIEW's in the next step.

The second step is to consider pre-existing VIEWs from the kill-victim concept. These correspond to various specific metaphoric senses of the verb 'kill'. However only those VIEWs that are members of the category found in step one are considered. For each of these VIEWs an attempt is made to match the viewed-thing of this VIEW to the target concept 'UNIX-PROCESS'. The VIEW that most closely matches the target concept will be used as a plausible source VIEW.

UCTeacher uses a hierarchical matching process to try to find a closely matching candidate VIEW. This hierarchical matching process attempts to abstract up the KODIAK hierarchy from the role that the viewed thing plays in the candidate VIEW until it finds a common ancestor with the target concept. In this case until it finds a common ancestor with UNIX-PROCESS. The VIEW which is selected is the TERMINATE-CONVERSATION-AS-KILL VIEW. This VIEW is represented below.

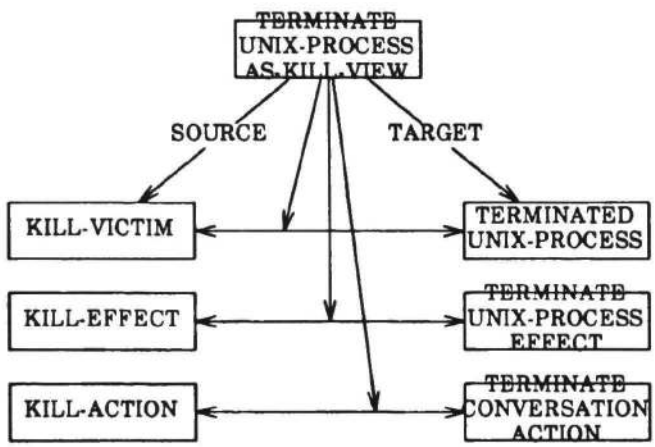


Killing a Conversation

This VIEW represents a mapping from a terminated conversation to the victim of a kill and corresponds to a word sense of kill indicating termination. The associated mappings are also shown relating the various actions and effects. These associated mappings plus the primary mapping on the victim constitute the COMPLEX-VIEW which represents this particular word sense.

The common parent concept that is found between TERMINATED-CONVERSATION and UNIX-PROCESS is TERMINATED-PROCESS. The process referred to here is the abstract notion of an ongoing sequence of actions with effects. It is important to note here that the system did not match the concept CONVERSATION against UNIX-PROCESS directly. It only abstracts on the role that CONVERSATION plays in in the context specified by the VIEW. This role is explicitly represented in the knowledge base by the concept TERMINATED-CONVERSATION. In this way the system is able to match only on those aspects of a concept that are relevant to the current context as defined by the VIEW. In this case what is relevant about the concept CONVERSATION is the fact that it can be terminated. It is this fact that needs to be mapped over to the target concept UNIX-PROCESS.

In the final phase of processing UCTeacher creates a new VIEW using the source view as a template. This new VIEW can be used directly in the future by both the analysis and generation components of UC. The following figure represents the views created for the new concepts.



Killing a Process

Conclusions

An expert using a natural language knowledge acquisition system will make utterances containing metaphors from the domain of interest. This poses a problem for the knowledge acquisition system since it may not possess the necessary metaphors for the domain. The answer to this problem is to give the system explicit knowledge about metaphors and a mechanism that can extend known metaphors to new domains. This approach is parsimonious with that suggested by Lakoff, among others. This knowledge about metaphors takes the form of a hierarchy of abstract metaphors and specific instantiations of metaphors with their meanings.

References

Brachman, R. J. et al., "Research in Natural Language Understanding". BBN report No. 4374, Cambridge, Ma. 1979

Carbonell, J.G., "Metaphor: A Key to Extensible Semantic Analysis" **Proceedings of the 18th Meeting of the Association for Computational Linguistics,** 1980

Carbonell, J. G., "Invariance Hierarchies in Metaphor Interpretation", **Proceedings of the Third Meeting of the Cognitive Science Society,** Cognitive Science Society, pp. 292-295, August 1981.

Gentner, D., "Structure Mapping: A Theoretical Framework for Analogy", **Cognitive Science** Vol. 7, No. 2, pp 155-170. 1983.

Jacobs, P. "A Knowledge-Based Approach to Language Production". PhD. Thesis. University of California, Berkeley, Report No. UCB/CSD 86/254, August 1985.

Lakoff, G. and Johnson, M., "Metaphors We Live By", University of Chicago, 1980.

Martin, J., "Knowledge Acquisition through Natural Language Dialogue" **Proceedings of the 2nd Conference on Artificial Intelligence Applications** Miami, Florida, December 1985

Wilensky, R., "KODIAK: A Knowledge Representation Language". **Proceedings of the 6th National Conference of the Cognitive Science Society,** Boulder, CO, June 1984

Wilensky, R., Arens, Y. and Chin, D., "Talking to Unix in English: An overview of UC". **Comm. ACM,** Vol. 27, No. 26 pp. 574-593, June 1984

Winston P., "Learning and Reasoning by Analogy", **Comm. ACM,** Vol. 23, No. 12, pp 689-703, December 1980