Context within Common Sense

Avni Shah
MIT EECS Dept.
ashah24@mit.edu

Rebecca Bloom
MIT EECS Dept.
rebeccas@mit.edu

ABSTRACT

There exists many software applications that attempt to use common sense reasoning to assist users do everyday tasks, such as schedule their day or write emails. One such program is ARIA, which relies on the information in OpenMind's common sense database. However, these applications are only as useful as the quality of the common sense information with which they are reasoning. One vast improvement in the area of common sense is to introduce the idea of context, since much common sense information is only valid within a certain realm. Thus, the objective of this project was to improve the quality of the information in OpenMind by adding context to potentially contextually-ambiguous statements currently in the database. Our application allows users to add common sense to the sentences in OpenMind as well as search for contextual sentences on a particular concept.

Keywords

Common sense, context, OpenMind

INTRODUCTION

There are currently many systems that attempt to incorporate common sense knowledge in their interactive applications. The belief is that by giving the system common sense knowledge about the world, the system will become "smarter", and thus be able to assist the user by presenting him with relevant, useful information based on what he is doing.

One example of an interactive application that uses common sense is Erik Mueller's SensiCal. [1] By accessing a large database of common sense information (Thought Treasure), this "smart calendar" is able to fill in missing information and detect potential problems for the user. For example, if the user inputs "Take Lisa to lunch at the steakhouse" and the system is aware that Lisa is a vegetarian, it may alert the user that he is taking a vegetarian, who does not eat meat, to a steakhouse. In this example, SensiCal learned from the information in Thought Treasure that vegetarians do not eat meat and was able to apply this knowledge to help the user avoid making a mistake.

Another such application is ARIA (Annotation and Retrieval Integration Agent) which is used to assist a user writing an email or web page. ARIA uses common sense to extract the "who, what, where, when and why" out of the user's text and present the user with pictures to annotate his email or web-page. [2]

An important realization is that an interactive application that uses common sense is only as good as the common sense that it is using. If the knowledge base is faulty or deficient, then the application will not be useful to the user. The database that ARIA uses to retrieve common sense information is OpenMind. The uniqueness of OpenMind lies in that the information in the database has not been inputted by a small group of computer programmers hoping to teach the system about common sense. Instead, OpenMind obtains its information from thousands of users who enter information into the system. The hope is that since there are such a random selection of people inputting data, the information will not only be more diverse, but will also more accurately reflect true common sense in society.

One of the limitations of OpenMind is that users often input information without paying attention to the context in which this information is true. For example, the database contains statements such as "a bride wears a white wedding dress." While this might be true in some religious contexts, such as Christianity, this statement is untrue for others. In Hinduism and in Chinese culture, bridges wear red and not white.

Our goal for this project was to improve the quality of the existing data in OpenMind by adding necessary context, thus making OpenMind a more useful source of common sense information. Our application extracts contextually ambiguous statements from OpenMind's database and allows users to attach context to these statements. In the previous example, a user might input "In Christianity, brides wear white wedding dresses" or "In Hinduism, brides wear red wedding dresses." Thus making this statement more accurate from a contextual point of view.

FUNCTIONALITY

There are three main functions that the user can perform with our application. Figure 1 depicts the interface, i.e., the options, that the user is presented with.

Figure 1: Interface to User

The first functionality is to add context to sentences currently in OpenMind. A user is presented with a randomly generated sentence from OpenMind's database and is presented with options to add contextual reference to the sentence. Due to time constraints, a few specific contexts (time and conditionality) were chosen to be in the application. After analyzing the sentence, the user can choose to add a time context ("when"), a conditional context ("if"), or another type of context ("other"). If the sentence does not need to be modified, then the user can choose "none" and will then be presented with another sentence to analyze. If the user chooses a specific context to add to the sentence, he will be presented with another window, which looks like Figure 2.

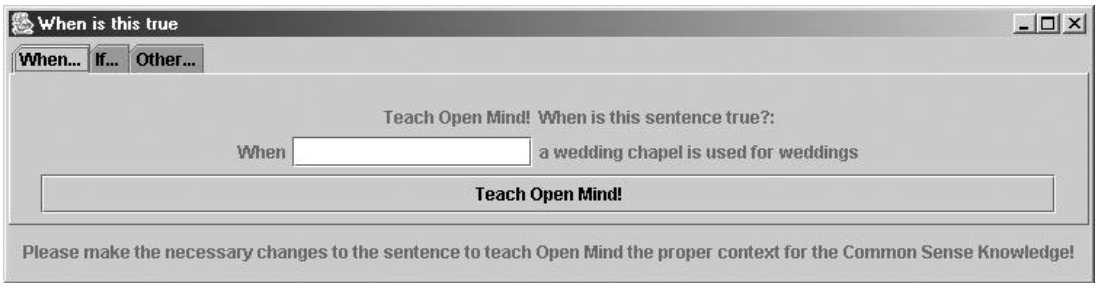

Figure 2: Inputting Context

Here, the sentence is not modifiable, but a basic template is laid out for the user. If the user chooses to add any other type of context, then the user is presented with a similar window, but this time the sentence is fully modifiable.

The second function that a user can perform is to view what others have written about a particular sentence. After being presented with a randomly generated sentence from OpenMind's database, a user can choose the "What others said about this sentence" option to see other users' modifications of the that sentence. This function was included because at first a user may not immediately realize a particular sentence is contextually ambiguous. However, after seeing how others added context to a sentence, the user may come up with another way to amend the sentence.

While the first two functionalities mentioned are directed towards improving the information in OpenMind, the final one is directed towards helping the user find contextual information. The user is able to enter a concept, such as "wedding" or "wedding dress", and then "Search for context about a certain concept". This will return to the user all the contextual sentences that other users have inputted about this particular concept. Thus, using the example from the introduction, if the user search for "wedding dress", he would be returned the sentences "In Christianity, brides wear white wedding dresses" and "In Hinduism, brides wear red wedding dresses". Thus, the user is given more contextually-accurate information than he could get from the current OpenMind database.

IMPLEMENTATION

To obtain information from OpenMind, Hugo Liu's interface with OpenMind (OMCSNet) was used. This interface provided a link to the information in the form of a semantic net, i.e., nodes and predicates/relationships to other nodes. Due to the large size of the semantic net, it was necessary to extract only the most relevant information, that is, the information that might have the

most contextual relevance, culturally and otherwise. For example, sentences having to do with weddings, entertainment, etc. were extracted from the semantic net and used for this project. Although there are quite a few sentences used, due to time constrains, all the possibly contextually ambiguous sentences were not extracted from OpenMind. Fortunately, more sentences can easily be added to the list of sentences actually used.

Once a sentence is randomly chosen and presented to the user, any modifications made to the sentence are linked to the original sentence and stored in an internal database. This internal database, which is maintained over all sessions, accounts for the system being able to present the user with different functionality. First, the system is able to keep track of all users' modifications to a particular sentence, which allows the user to see "What others said about this sentence". Second, the system is able to search for modified sentences having to do with a particular context, thus enabling the user to "Search for context about a certain concept". This internal database is kept offline, meaning that it does not link dynamically with OpenMind's database. However, a file of modified sentences is maintained so that the information can eventually be stored in OpenMind's database and thus accessible to all OpenMind's users.

LIMITATIONS

The original intent of this project was to actually search OpenMind's database to find seemingly contradictory statements that would not be contradictory if context was added to them. With the use of a semantic net representation of the information in OpenMind, it would have been easy to find contradictions such as "A person wants to eat" and "A person wants to not eat", since one sentence is just a negative of the other. However, there are not enough sentences like this in OpenMind to make this sort of search worthwhile. Instead, in order to be really useful, the system needs to find more subtle contradictions. For instance, it would need to find sentences such as "A bride wears white" and "A bride wears red," which seem like opposing sentences until one adds cultural or religious context to each sentence. Unfortunately, finding such contradictions is a very difficult problem, and requires the system to be smart enough to realize, for instance, that this is a contradiction because a person cannot wear two colors at the same time. Unfortunately, given the time constraints for the project, it was impossible to create such a complex reasoning system. Thus, the project was limited to presenting the user with a single sentence, and having the user – not the system – determine whether the sentence had contextual implications.

EXTENSTIONS

There was one main functionality that, due to time constraints, was not implemented in this project. As mentioned earlier, the system has an internal database that links modified sentences to their original sentence. Unfortunately, this linkage is not retained when the modified sentences are fed back into OpenMind. Therefore, although OpenMind is being given more contextually accurate information, OpenMind has no idea which sentences were originally contextually inaccurate. Because of the limited time span of this project, a method of tagging the original sentences in OpenMind's database could not be found. However, the ability to do might greatly improve the usefulness of OpenMind's information as users could search within OpenMind for contextual information linked to a particular concept.

ACKNOWLEDGMENTS

We would like to thank Henry Lieberman, Push Singh, Hugo Liu, and our fellow MAS.964 students for all their help and suggestions.

REFERENCES

1. A Calendar with Common Sense. Available at http://lieber.www.media.mit.edu/people/lieber/Teaching/Common-Sense-Course/Common-Sense-Calendar.pdf.
2. Adaptive Linking between Text and Photos Using Common Sense Reasoning. Available at http://lieber.www.media.mit.edu/people/lieber/Teaching/Common-Sense-Course/AH2002-aria.doc.
3. CHI Conference Publications Format. Available at http://www.acm.org/sigchi/chipubform/.