Final Project Ideas for CSCI181o

Students are encouraged to work in pairs on the final project. The project consists of building a computational model of some linguistics phenomenon. You will make a 20 minute presentation of your project during the last few weeks of the semester as well as turning in a substantial paper (10 pages or more) describing your contributions. The computer program will be included as an appendix to the final paper.

While I have listed some suggestions below, I urge you to think of projects on your own. Another source of ideas are the last chapters of the text. Chapters 12 and 13, in particular, introduce some interesting ideas. While we will be at least surveying the ideas in those chapters, there are many ways in which you can go deeper and find interesting projects that build on these ideas.

The following projects are just suggestions. Feel free to approach me with your own ideas. All projects should cite appropriate literature in (computational) linguistics and describe the current state of the art, as well as your own contributions.

By Monday, April 4, each group will need to turn in a one page description of their project, listing two or more references for current work relevant to your topic, and indicating who will be involved in the group.

Topics

1. **Computational model for intensional logic as a basis for games**: It is not too difficult to construct a computational system representing terms of intensional predicate logic and its possible world semantics. Construct a mystery game which helps players deduce the perpetrator from a variety of clues that allow players to reduce the possible worlds to one in which the perpetrator is determined. Make the game as user friendly as possible. In particular, make sure the system responds to a variety of user queries that a player might want to make.

2. `Question answering systems:` Our text talks a bit about question answering systems, but it is fairly limited. Develop a way of representing information and responding to natural-language questions. Perhaps add a deduction system so that you can figure out if a statement really answers a question. In general this can be quite tricky. E.g., "Will John be here soon?" with a reply of "John is stuck working late tonight" or "John got a flat tire". However, you should be able to do something interesting.

3. **Following directions:** We would like to create robots that can follow directions. Build a system that converts a set of English-language directions into commands for a robot. A good test bed might be following the directions in a cookbook. You can make up your own set of primitive commands that a robot can follow, and then construct a software system that translates natural language commands into robot commands. There are lots of interesting challenges to this project.

4. **Common knowledge** During the course of a conversation, each participant typically asserts statements that the other is unaware of. When the listener nods or says "OK" or gives some other positive response, that statement is added to the common knowledge.

5. **Presuppositions:** When we speak, our expressions depend on the context around us. When we say "the dog", for instance, we are assuming that we are in a context in which there is only one relevant dog. As another example, saying "Whomever invented calculus has caused generations of students great pain", involves the presupposition that there is someone who invented calculus. If you are listening to that sentence then your normal reaction is to assume there is such a person (though it's also possible to raise a question about the presupposition). Model this in a system with common knowledge. In particular, pick out the presuppositions implied by a sentence and determine whether the presupposition is already true in the context or whether the listener must add it to their common knowledge in order to accept the sentence.

6. `Extending semantics`: Extend (and implement) the parser and semantics provided to a much richer fragment of English or of another language.

7. `Discourse Representation Structures`: Chapter 12 of our text discusses extending the computational model to discourse representation structures, a popular method of representing semantics and dealing with references via pronouns. Learn about DRS and extend the ideas in the text.

8. `Semantic filtering`: We all know about spam filters that attempt to recognize and reject annoying e-mail. If our system is better at understanding words, we can be smarter about what we can recognize. Abusive e-mail is hard to recognize if you don't understand the text (just looking for abusive words is not sufficient). Write a program that detects abusive language in a collection of sentences. You may assume that the text arrives already parsed or you can enhance a parser to provide more help.

9. Model checkers are tools used to verify hardware and software systems in computer science. Information about the states of the system are represented in specialized tense logics (a form of intensional logic), for example in varieties of temporal logics. Investigate model checkers and either implement one or take one or more existing model checkers (e.g., Spin) and use it to verify a complex computing system. [Some model checkers may not involve tense logics. Please focus on those that are related to the material in this course.]