Project Proposal

Our group members are most interested in language generation, and we brainstormed a number of ideas. We chose two ideas that we think are both interesting and will also be feasible to make good progress. We are presenting proposals for both ideas so that we can get feedback from you on which idea is best suited for the course project.

Group members:

- Alan Davis
- Jason Switzer
- Ryan Garabedian

Haiku Generator

For this project, we intend to build a natural language model that can generate haikus, short poems with well-defined structure and a strict syllable count. We will need to collect a fairly large corpus of haikus and add markup to each poem to indicate the location of line breaks and the syllable count of each word. At first, we can use a simple but approximate approach to count syllables and then later improve the accuracy with a more sophisticated technique. We will train a statistical language model from the corpus and stochastically generate haikus with the intent to create poetry that sounds natural, as if a human could have written it. Since this will obviously be a subjective evaluation, we will create a protocol to measure how natural the poems sound to compare the effectiveness of our haiku model as the project progresses.

An interesting extension to this would be to allow the user to input a word or phrase that they want to appear in a generated haiku. If the given word does not appear in the corpus, we can use a synonym for it or another word related to the given word, which could be learned from the corpus. One idea for how to do this is to use an n-gram Markov model with both forward and backward probabilities to generate words before and after the given word at a particular location in the haiku. We could generate one haiku for each of the possible locations that the given word could appear in a haiku and use a quantitative measure of how 'good' each haiku is and then choose the one that has the highest score.

Phases of development:

1. Collect corpus of haikus from websites, parse into plain text, markup with syllable counts
2. Train n-gram Markov model and generate random haikus (using forward probabilities only)
3. Use our protocol to evaluate how natural these haikus sound to create a baseline measure
4. Train n-gram Markov model and generate haikus (using forward and backward probabilities)
5. Create a way to find related words to a given word (maybe using a soft clustering approach)
6. Generate haikus that include a word or phrase input by a user (or a related word, if needed)
7. Improve syllable counting to generate haikus that follow typical haiku structure more closely
8. Improve semantic cohesiveness of haikus (to discuss one or two topics and avoid rambling)

Source Code Generator

This proposal is to use Statistical NLP techniques for natural languages and apply them to "unnatural languages," such as a simple programming language. We aim to generate source code from a probabilistic context-free grammar learned from a corpus of source code we will download from Google Code Search. We aim to automatically generate code that will compile and execute, and possibly do something interesting. There are many phases of development, features to build upon, and complications for this idea, as listed below.

Phases of development:

1. Choose a simple UL (unnatural language) grammar, such as a reverse polish calculator
2. Build a small corpus of code for the simple grammar
3. Train a probabilistic context-free grammar that generates syntactically correct code
4. Download and filter a corpus of code with incrementally more complex syntactic structures
5. Incrementally build the grammar until it generates valid code in a real programming language
6. Improve the grammar to generate code that not only compiles but actually does something

Potential complications:

- Will need to filter source code in the corpus that does not pass the grammar
- Will need to manually create source code if none is readily available for the simple grammar (particularly in the early stages, but possible at any point in our progress)
- Will have to be able to keep track of semantic information (e.g. variable types) across lines, blocks, and functions similar to semantic information in natural languages across paragraphs

It is important to mention that we think the output of a source code generator is not likely to be as interesting as the output of a natural language generator.