Areeb Malik, amm385
Benjamin Jaeger, bgj9
Zach Silversmith, zbs7
CS4740
12 March 2012

WSD Proposal

Algorithm

We noticed that a number of successful teams at SENSEVAL-3 used a SVM classifier for supervised word sense disambiguation. Hence we have chosen to implement one for this project. Each word will have its own mutliclass SVM model to determine which word senses are appropriate. Since SVM in its simplest form is a binary classifier, each word sense will have it's own binary SVM classifier. The SENSEVAL-3 task definition specifies that more than one sense may be appropriate for a word. Having a per-sense classifier will allow us to give independent predictions for each sense. We will try using both a hard margin and soft margin classifier. Given that the training set may have errors, we predict that soft margin will perform better. We will also try different kernels, linear and non-linear, to see which is most appropriate for disambiguation.

Features (more to come)

- *Collocation*: We believe this will allow us the biggest gain in accuracy in determining word senses. For each word sense, we will compile a list of *k* words commonly found within a range of *n* words from the target. The lists will represent the word pairs with the highest correlation based off our training data. Then, when parsing a test sentence, we can check for the existence of any of these words and use this knowledge to aid our decision. This will likely follow the Lesk Algorithm.
- *Neighboring Parts-Of-Speech*: We can hopefully gain knowledge about a word sense by eliminating senses with parts of speech that don't fit into the current context. For instance, we would hardly ever see two verbs adjacent to each other, so this can be used to realize that our word sense is unlikely to be any that corresponds to a verb.
- *Bag-Of-Words Analysis*: As a fundamental level of analysis, we will consider each word out of context to notice any strong biases in terms of word sense. Knowing a particular word sense has a very limited use can help us break ties in terms of our most likely sense.

Evaluation

In order to develop our model and choose the best features, we will split the train 80% / 20% into train and cross validation sets. To determine performance, we will consider both percent accuracy and F-measure. To evaluate the efficacy of a feature, we will try using the classifier without that feature and see how much performance degrades.

Implementation

We will use Python and PyML which has a SVM library.

Tasks

Zach - Analyze training data to determine fundamental probabilities, collocation frequencies, etc. Implement cross validation.
Ben - Design algorithm that uses these statistics to estimate the part of speech.
Areeb - Research gains and losses from using various combinations of features. Compare changes in certain variables such as whether to use hard or soft margin for the SVN.