NLP Project: Word Sense Disambiguation
Lauri Tarpila, Elina Gundyreva

1. Project topic

Our project was about Word-sense disambiguation using Yarowsky's supervised and unsupervised word sense disambiguation algorithms. We started by reading the Yarowsky's papers from 1995 and 1994. We used Python as the programming tool.

The task was straightforward: find 3 ambiguous words and create a method that takes in sentences which contain the target word; returns the collocations for the target word. From there we can easily create the decision list and make predictions.

2. Project implementation

The words we chose were 'rock', 'bass' and 'crane'. These words have many senses but because we only looked at 100 instances and did not take the verbs into account, we found around 2-3 senses for each word.

Rock:

1. rock as music

2. rock as material

Bass:

1. bass as guitar/drum

2. bass as singer voice

3. bass as fish

Crane:

1. crane as bird

2. crane as machine

When picking up sentences which contained the target words, we noticed some problems. For example, we first picked sentences for 'rock', which had the word 'bedrock' in them. We got though this by searching for sentences, which contained the word rock with no other letter around it. This way rock'n roll would still be accepted but bedrock not. In addition, to make the programming easier we lowercased the sentences. This was mainly because the word 'crane' appeared with a capital C and without.

Manual classification of the 100 instances of the words was a bit troublesome but luckily there were only 300 sentences to go through. After that we could make all things automated.

We chose only to use plus and minus 3 words around the target word as a dataset. This was purely a choice which would make our calculations simpler. In addition, the method that produced the collocations only produced +-3 k word collocations and +1- and -1-word collocations. Again, we only chose these to make

things simpler, but these were in many ways sufficient. With the collocations, we could easily make a method which creates the decision list and can make predictions.

3. Results and conclusions

For supervised learning, we split the 100 instances of the target words into a training and a test set. The training set included the 80 first sentences and the test set was the last 20. To make the decision list for each ambiguous word, we used the training set. Then we made predictions to the test set and calculated the accuracy. The accuracies are below:

Rock: 50%

Bass: 55%

Crane: 85%

The small accuracy in rock is caused probably mainly because of a small training set. The test set happened to contain contexts which were not familiar to the training set. For bass 55% is not that bad since there are 3 different senses. But again, here a bigger training set would probably help. Crane was successful maybe because the contexts of different senses were different but within the same sense the contexts were similar.

We also implemented an unsupervised learning algorithm for learning new senses. Here we only used +-3-word collocations to make the calculations easier. We created the initial decision list based on the supervised learning results of classifying 80 instances, and used it to classify 100 new instances using the 'one sense per collocation' principle. From that, a new decision list was created and the test set was classified. Accuracy was calculated for the test set of 20 tagged instances.

The accuracies are below:

Rock: 55%

Bass: 75%

Crane: 75%

As we can see, accuracy increased for 'rock' and 'bass' words but decreased for 'crane'. In the algorithm, we didn't calculate the threshold for assigning values to decision list, but for each collocation, used sorting to assign the sense of the max possible log value, which may lead to misclassification in case of the biggest log values not matching. But overall, the accuracy seems satisfactory, the unsupervised learning algorithm was able to produce new collocations.