Project 1: Language Modeling

Luis Gustavo De Medeiros (lsd53) James Russo (jdr289) Noah Sterling(nas95)

October 3, 2016

1 Tagger Choice

For our implementation we will be using the ntlk library and therefore implementing both HMM and CRF for sure since these are both built in taggers for nltk.

HMMs basically assume the system is a Markov System with hidden states. An HMM has a set of states, which in this application, is B-CUE,O,I-CUE and a set of observations,which is the word plus its part of speech. In reality HMMS are useful if trained on a large corpus, and the probabilities used in HMM can be smoothed using the techniques from the last project. The Viterbi algorithm can be used to find the most likely set of states. We choose to use HMMs as they are easy to implement and understand, and generally work well in NLP applications.

We thought it would be interesting to see how the HMM and CRF perform against each other and we may play around with the original parameters such as number of states for HMM, etc. to see how this changes performance. HMM and CRF are related techniques so we thought it would be a good choice to pick them both. HMM(Hidden Markov Models)

2 Extension

Since we are using the nltk library we thought it would be interesting to try multiple extensions. The first extension we are looking to implement is using another type of tagger from the library since there are other options. The main tagger that looks most promising is the perceptron implementation. We would like to see how this tagger performs in compared to our other taggers.

Another extension we are thinking about implementing along with the perceptron tagger is to mess around with the training data to balance it more. We feel that the training data is fairly sparse with uncertainty and that a big part of increasing performance on the test data may come from better balancing the training data.

3 Kaggle Results

Team Name: Tagged Up

Span uncertainty

0.09912

Sentence uncertainty

0.30067

Our very modest kaggle results are probably a result of our very simple baseline implementation. For our baseline implementation we just took a list of known hedge words and build a dictionary out of it then looked for those hedge words in the test sets. We believe our machine learning taggers will be able to perform much better.