NLP Project 2: Team lite lite

Nithish Raja Chidambaram (nr293), Akash Peri (ap659), Vinayaka Suryanarayana Bommathanahalli (vb247)

Our goal for this assignment was to implement a Named Entity Recognition model. For our baseline model, we decided to implement the basic NER model described in the project. To build this lexicon, we made the following decisions:

* If the token is not ‘O’ (a valid NER) and we have not seen it before, then record the mapping. We only record the first occurrence.
* If the token is ‘O’, we do not store it for space reasons

Then when evaluating on our training data set, we did the following:

* See a word we've seen before, use that NE value
* See a word we've never seen before, classify as O

This is a very simplistic baseline. However, we thought an effective baseline because the majority of the words we will encounter are not Named Entities.

However, our first run gave us 35% accuracy. This was quite a bit lower than we were expecting. We thought this was because we were not correctly classifying multi-word NEs. We were treating them as separate words, resulting in two different side-by-side NEs, rather than one combined one. We updated this, but were still only around 50%.

Proposed Enhancements:

Our baseline model is not that great, and we can definitely improve on it. To do this, in our extension, we intend to use a Hidden Markov Model, trained with the probabilities from our lexical model. We chose a Hidden Markov Model because we're more familiar with the machine learning approaches, and wanted to learn more about HMMs. Then, we can use the Viterbi algorithm to find the most likely NE tag sequence in the state space of possible tags, given the transition probability from hidden state to hidden state.

* 9 Hidden states: 4x B-*, 4x I-* and O
* Transition probabilities: Probability of transitioning from NE tag to NE tag; calculated from the training set
* Emission probabilities: P(w|tag)

To be able to capture the transition probabilities, we need to change our lexical model to one that works off of bigrams. That way, we can capture the transition probability across the entire training corpus, for transitioning from NE tag to NE tag.

To capture the emission probabilities, we must now enhance our lexical model again. We will record all occurrences of a token, as opposed to just whether we saw it or not. We will also record the different NE tag mappings it was assigned. For instance, 'Toronto' was both a B-LOC and B-PER in the data set. We will use these in the Viterbi algorithm.

For this task, we chose to use HMMs over SVMs. HMMs are good at modelling how known information relates to unknown information. If animal walks like a duck, quacks like a duck, and swims like a duck, then it is likely a duck. For us, HMMs are good at modelling sequences as well. This is because once we train an HMM, it contains the various hidden states, transition probabilities between those states, and an initial probability matrix. We can easily determine the probability of a sequence happening, and also track partial sequence probabilities as well (in case we want to classify "University of British Columbia" by the probability for "University of"). This is an advantage they have over SVM's, where the string we are looking up has to be converted to some static feature representation that can be influenced by string length. The disadvantage to HMMs however, and especially the one we are implementing (as opposed to MEMM) is that we are severely limited in the number of additional features we can consider (for instance, we can't directly use the POS tags in our HMM). In addition, we can't capture higher dimensional relationships, like we might be able to with kernelized SVM, to find trends based on augmenting the data with functions. This is related to the previous point, where an SVM can easily be extended to contain a new feature for Proper noun POS tags where the word starts with a capital for instance.

We've decided to implement extension 1 and 2 for the project. This is because we have experience with implementing n-gram models, and would like to see how large of ‘n’ will lead to significant improvements in the model; and because we are interested in learning more on CRF, as it is an alternative sequencing model.