NLP Project 2 Report

Kevin Li (kl553) and Rebecca Jiang (rwj52)
Team Name: The Dynamic Duo

I. Abstract

In order to properly test our model, we had to split the training set given into a training and validation set. In order to ensure semi-randomness in our validation set, we decided to remove every tenth data point and put that into a separate file for validation.

II. Baseline

Our baseline implantation looks to see whether or not the entity is identified as a named entity in the training set. If it is not in the training set, our baseline implementation will not recognize it as a named entity.

In order to accomplish this, we initialize a dictionary where all the keys will be the entity that appears in training and their value is the tag associated with it. We do not keep track of O tags. Once this mapping is compete, we use this as a look-up table for the validation set. We loop through each token and check to see if it's in the dictionary. If not, we denote it as an 0. If it's denoted as a B tag in our dictionary, then we mark it as such. If it's denoted as an I tag, we check to see if the previous token was a B or I tag of the same entity type. If they match, we mark it as an I tag of that type, and if not then it is marked as an O.

We checked our baseline model based on their precision, recall, and Fscore below...

- Precision: 0.748801804342
- Recall: 0.776608187135
- Fscore: 0.762451557342

We ran the baseline prediction model on the test set and submitted the result to Kaggle, where we received .17354 accuracy score. Perhaps one reason for a low accuracy score is that our baseline is able to identify parts of named entities, but unable to find the entire entity.

III. Hidden Markov Models

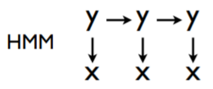
A Hidden Markov Model (HMM) is a probabilistic sequence model whose job is to assign a label to each unit in a sequence. It computes a probability distribution over possible sequences of labels and chooses the best label sequence. It uses the Markov assumption, which assumes that the current state is the only thing that matters to predict a future state.

The states are represented as nodes in the graph, and the transitions (with their probabilities) as edges. The states represent the possible lexical categories (PER, LOC, ORG, or MISC). The hidden variables are the part of speech tags, and the observed variables are the words that we see as input.

The tag transition probabilities, which represents the probability of a tag occurring given the previous tag. These probabilities is calculated using Maximum Likelihood Estimations (MLE), which involves counting the number of times a tag is followed by a specific tag, divided by that specific tag.

At each state we find the maximum probability of the current state by calculating probability of the previous state at a specific tag * transition probability * probability of current state being this tag. We do this for each tag, and the maximum probability will determine what tag the current token should be labeled as.

x is tokens y is BIO



$$p(\vec{y}, \vec{x}) = \prod_t p(y_t|y_{t-1})p(x_t|y_t)$$

Figure 1: HMM Figure

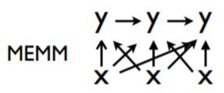
In the figure above, x represents the tokens of the corpus, while y represents the corresponding tag. This figure is demonstrating the generative nature of the model, where the probability of the

HMMs are able to look at the previous state as an input, which makes it more effective than our baseline model which only considers the current state. HMMs are also flexibile in the sense that the transition probabilities and observations are configurable given the context of the problem. For BIO tagging, HMMs work really well.

However, HMM models capture dependencies between each state and *only* its corresponding observation. There may be some non-local features that HMMs is not considering. Additionally, HMM learns a joint distribution of states and observations P(Y,X), but in a prediction task, we need the conditional probability P(Y—X). Furthermore, with a fully connected diagram, this can lead to the sequences being very closely mapped to the training data, which can lead to severe overfitting.

IV. Maximum Entropy Markov Models

One big issue with HMMs is that they only look at the previous state, whereas the Maximum Entropy Markov Models (MEMM) can have features from the input and the previous state. In this project, one of the features we are provided with is the part of speech of each token.



$$p(\vec{y}|\vec{x}) = \prod_t p(y_t|y_{t-1}, \vec{x}, t)$$

$$p(y_t|\vec{x}, t) \propto \exp(\theta^{\top} f(\vec{x}, t, y_{t-1}, y_t))$$

Figure 2: MEMM Figure

In the figure above, y is represents the previous state and x is the part of speech tags which our MEMM will take as a feature. We are given this feature in our data set. This means, that our MEMM system will make use of the word tokens, part of speech tag, as well as the BIO tags from the prior state to construct it's features

MEMMs can condition on any useful feature, which is extremely powerful because this feature can clean up the model. MEMMs are more expressive than HMMs because they can explicitly model the dependencies between each state. It also saves the modeling effort because it can ignore the P(X) term that HMMs need to account for,

However, MEMM are much more difficult to implement than HMMs. Additionally, MEMMs may over-complicate some problems, and you always want to use the simplest model that is able to solve the problem