Model

The NER model I implemented was inspired by work done by Chui and Nichols [1]. In particular, I trained a word embedding model (Word2Vec) on the data corpus to generate continuous, 100-dimensional vector representations of the words in the training and testing corpus. I also experimented with Stanford's GloVe embeddings but found that many of the words in the training set (especially names) were missing in the embedding, so I used Word2Vec instead. I used the word embedding to create an Embedding layer. I then used a Bidirectional LSTM layer. My intuition behind choosing the LSTM layer was to be able to learn long-distance relationships between word vectors. Additionally, making the layer Bidirectional rather than feed-forward meant that the model would learn word context in both directions (i.e.: learn about the word based on what came before it and after).

Finally, I applied a Dense layer with a softmax function to perform the classification (categorical cross entropy loss function). This gave a probability of belonging to a particular class, from which a class could be inferred. I also attempted to use CRF to predict the onn-hot encoded classification sequence rather than a probability. However, the use of CRF led to very poor results since the model kept predicting the none class (i.e.: it predicted that almost all the words weren't named entities).

The overall F1 score was 74.6; however, the model performed quite poorly in terms of classification-it was able to accurately decide when a word is or isn't a named entity, but it has trouble classifying.

I only performed classification for the root labels due to time constraints. My plan to classify the leaf nodes would be to first classify the roots (i.e.: decide if a word is a person or an organization). Once the roots are classified, train the same model but only on the predicted root class. For instance, if the model predicted "organization", train the same model only on training data classified as "organization", with the new goal of classifying the child nodes of the organization root. This algorithm can be generalized for any N number of layers: keep pruning down the NER tree until you reach a leaf.

Model Improvements/Changes

Due to time constraints, I was unable to explore a variety of novel techniques and interesting avenues. In particular, I would have liked to explore Poincare word embedding techniques. The general motivation behind this is that standard word embedding techniques like Word2Vec aren't able to take into account hierarchical information since the area required to represent the branching structure of a hierarchy isn't available in Euclidean space, but is available in hyperbolic space (on a D-dimensional Poincare ball) [2]. I suspect that using this word embedding may also make training far easier since, instead of using the algorithm above where you have to prune the tree and keep training new models, you may be able to train all levels at the same time (i.e.: you can train "organization" and "organization/political" at the same time). Then this embedding could have been fed into the LSTM model described above for classification, or I may have performed a dimensionality reduction (CNN or VAE) and then fed into the LSTM.

References

[1] Jason P.C. Chiu, and Eric Nichols, Named Entity Recognition with Bidirectional LSTM-CNNs
[2] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea, Poincaré GloVe: Hyperbolic Word Embeddings