Alex Book

CSCI 5832 – NLP

Prof. Jim Martin

Homework 3 - Named Entity Recognition

When exploring the various options available to use for this homework, I decided that using my solution from Homework 2 (sentiment analysis using feature vectors) as a starting point was my best bet in terms of efficiency, accuracy, and understanding of what is happening (I'm not as familiar with BERT systems, and an HMM-based approach could likely be less accurate).

I explored various possibilities for how I would go about extracting features from each token. An approach similar to that in Homework 2 would limit what kind of features I could have, so I did some more digging about how I could include features such as the token itself, the previous and following tokes, the previous IOB tags, and various other non-numerical features. The best solution I could find was that of sklearn's feature_extraction.DictVectorizer. It takes any features that the user desires (and has stored as a dictionary) and turns them into numerical feature vectors using one-hot encoding. Fortunately, this proved to be exactly the kind of tool I was looking for.

In determining the type of classifier I would use, I started with the idea of expanding upon the Stochastic Gradient Descent method used in Homework 2 but ended up shifting to trying various classifiers available through the sklearn library. I honestly wasn't sure what would work best, so I tried a couple options from sklearn.linear_model (SGDClassifier and PassiveAggressiveClassifier), then tried one from sklearn.svm (SVC). I ran into some issues with both SGDClassifier and SVC in how I stored/vectorized the feature dictionaries, in that the non-sparse vectors were too large for memory to handle (the dimensions were in the hundreds of thousands), and the sparse vectors weren't able to be properly handled at all (odd errors were produced that I was never quite able to find a solution to). Even after changing and cutting down the number of features, these issues persisted, so I went with using the PassiveAggressiveClassifier (in simpler terms, I fell into using this classifier because I liked the features I was using and wanted to use a classifier that could handle them). It turned out that this classifier was a good choice, because it trains incredibly quickly, which is quite nice considering the volume of data on which we were to train our models (at least in comparison to the previous homework). In my understanding, it essentially makes no changes if the prediction is "good enough" (passive), and only updates weights if the prediction is "wrong enough" (as opposed to algorithms such as SGD that update weights every iteration until the predictions are within a margin of tolerance of the correct answer). Additionally, when it adjusts the weight vector, it does so in a way that the prediction for the data that was most recently read in is precisely correct (aggressive). This leads to the proper weight vector being learned quite quickly.

Any sources from which I drew inspiration are listed in my python file, so feel free to check those out if you're curious about where I got the ideas for my chosen features, as well as a better understanding of the Passive Aggressive Classifier!