Rianne Lyons
LING 539
11/26/2017

Assignment 3 Written Report

Requirements:
1. Python 2. This version can be found at https://www.python.org/downloads; to install, click Download Python 2.7.13 and follow the instructions.
2. Numpy. To install on Linux, enter sudo apt-get install python-numpy at the command line (Ubuntu) or find the package for the appropriate distribution. To install on Mac or Windows (or Linux) download Anaconda or another distribution for Python 2.7 (links are at https://scipy.org/install.html).
3. DyNet. To install, follow the instructions at http://dynet.readthedocs.io/en/latest/python.html under Installing DyNet for Python (it may be necessary to run sudo apt-get install python-dev first before DyNet installs successfully).

Problem 1:
To train and test respectively in Problem 1, run the following commands in a terminal window after navigating to the directory where problem1.py is located:

```
python2 problem1.py train train.tagged
python2 problem1.py test test.tagged
```

The model is saved as the file model.pkl. Overall accuracy is 87.85% and accuracy on unknown words is 27.36%.

Problem 2:
To train and test respectively in Problem 2, run the following commands in a terminal windowafter navigating to the directory where problem2.py is located::

```
python2 problem2.py train train.tagged
python2 problem2.py test test.tagged
```

The model is saved as the file model2.pkl. The accuracy went up for both overall accuracy and unknown word accuracy after adding add-one smoothing. Overall accuracy is 90.49% and accuracy on unknown words is 27.70%.

Problem 3:
To train and test respectively in Problem 3, run the following commands in a terminal window after navigating to the directory where problem3.py is located::

```
python2 problem3.py train train.tagged
python2 problem3.py test test.tagged
```

The model is saved as the file model3.pkl. The accuracy went down for both overall and unknown words, although the accuracy for unknown words is significantly lower than expected, which may indicate an error. Overall accuracy is 88.86% and unknown word accuracy is 0.05%.

Problem 4:
Much of the code for implementing the RNN with gated recurrent units came from Michael Capizzi's DyNet tutorial at https://github.com/michaelcapizzi/dynet_tutorial, and I attempted saving the model using the DyNet documentation at http://dynet.readthedocs.io/en/latest/python_saving_tutorial.html, but loading the model in resulted in a significant drop in accuracy when testing a small portion of data (from around 80% to about 2%). Because of this, I trained, tuned, and tested on a smaller portion of the data and did not load in a previous model (although the functionality is commented out in my code). I used a batch size of 40 and traversed 50 batches in both training and testing, and tuned for 3, 4, and 5 epochs. I chose an epoch size of 5 as the best model, which is used in testing and results in 84.73%

overall accuracy and 32.63% unknown word accuracy. Because I used a smaller portion of data, the time was comparable to my HMM implementations and the accuracy was slightly worse for overall words and slightly better for unknown words. If I had used all the data, I suspect that the model would have taken significantly longer to train but would h ave given even better accuracy results.
Training: python2 problem4.py train deps.words train.tagged
Tuning: python2 problem4.py tune deps.words train.tagged dev.tagged
Testing: python2 problem4.py test deps.words train.tagged test.tagged

Problem 5:

The different state-of-the-art POS taggers at the current time of the paper achieve best results by careful design of feature sets. As described in the paper, the sequence model tagger that uses bigrams and trigrams over a three-word and three-tag window as features has the smallest number of feature sets and gets the lowest accuracy on both sentences and tokens. The tagger with the next best performance and number of feature sets adds more sets within the three-word and three-tag window, and the replications made of it achieved better performance by minimal adjustments. The higher-performing group of taggers described (on sentences and tokens) use a five-word window, and the better-performing taggers use word shape feature sets and features based on distributional similarity classes.

Manning describes the biggest challenges to boosting and evaluating performance on POS tagging as common error areas in automatic tagging and lack of progress in updating gold standard data. Some of the areas where automatic tagging appears to lose the ability to improve are areas of linguistic difficulty like unknown words, unseen context, and instances where the tag is informed by discourse, as well as areas like incorrect gold labels. The reluctance to update gold data and the lack of progress in that area contributes to those error-prone areas, which prevents automatic tagging from improving.

Some methods that Manning advances in order to increase accuracy are correcting the errors in the Penn Treebank and examining the use of categories in tagging. To correct errors in the gold data, Manning suggests using deterministic rules based on the X'-theoretic syntactic tree structures in the Treebank, which can fix errors (such as past tense tags being exchanged for past participle tags, plurals being tagged as singulars and vice versa, and difficult words like "that" being misclassified) automatically. With regard to the use of categories, Manning describes how annotators are not consistent due to varying semantic influences on syntactic categories, and suggests examining how tags are assigned for words such as "worth," which is listed under multiple categories in multiple sources.