Assignment 2

MATT BUBERNAK
3/4/2015

Test Results

Test Number	Precision
1	91.508656%
2	91.653189%
3	93.422978%
4	93.115211%
5	94.422062%
6	92.883581%
7	93.401436%
8	93.151429%
9	94.153170%
10	85.547009%

[FIGURE 1]

Overall Estimated Accuracy: 92.31%

Accuracy on provided test set: 91.55%

Analysis

Training a Model

Before I could go about calculating the Viterbi, it was necessary to be able to build a model of probabilities based on a set of training data. I do this by reading in each word of the data, and adding additional "START" and "END" words to the respective beginning/end of each sentence. I mark these words with the appropriate "START" and "END" tags. These will allow me to better approximate values such as a0,s (probability of a state given it is coming after the start), and a(s,qf) (a state given it's the final state).

For each tag I read in from the training data, I store key statistics. These statistics include calculations that allow me to later access probabilities *P(t,t-1)* (probability of a tag, given this tag) and *P(o,t)* (probability of a word, given this tag). These are necessary for calculating the Viterbi algorithm.

Dealing with Unknown Words

My original implementation did not deal with unknown words. This meant that many of my probabilities would quickly become zero, and the results were very inaccurate. It was very obvious I would need to implement some sort of smoothing.

I decided to use +1 smoothing, or Laplace smoothing. This meant that, I needed to increase the count of each word/tag that I had seen by one (increasing the total count as well), and consider any "unknown" word to have occurred once. This proved moderately effective, but still gave too high of a probability to unknown words/tags. I found, through using my modified evaluation script, that some uncommon tags were being labeled far too often. I subsequently reduced the weights given to unseen words/tags, and ended up finding my maximum success at about these values.

- .05/totalTagOccurances (for unseen words)
- .15/totalTagOccurances (for unseen tags)

System Evaluation

In order to evaluate the strength of my system, I decided to utilize a 10-folder cross validation method. This meant that I broke the training corpus into 10 parts, and evaluated it in 10 different scenarios. Each scenario trained on a training set consisting of 9 parts, and tested with the other 1 part. This meant that no individual segment of the training data was left out of testing. My results, displayed in figure 1, show that overall, my system proved 92% efficient.

In order to better understand my results, I modified the evaluation script. I included output for each tag that was most mislabeled. This revealed some interesting results. Common tags (i.e. NN, VB,IN), were almost always labeled correctly(Nearly 100%). However, some less common tags (i.e. LS,RBS, :), were 100% missed. I believe this could be the result of just not having enough data that includes these tags, and also possibly the result of assigning a very low probability to unseen values.

System Tuning

While I am very pleased with the final performance of my system, it was not without some tuning. There were a few key design decisions that improved my overall efficiency. These include:

- Modifying my original method of calculating the "initial" column of the viterbi matrix from using simply the probability of a tag, given the sum of all tags, to instead calculate the probability of a tag given the "start" of a sentence. This was done by adding the "START" and "END" tags, as described above.
- Modifying my smoothing to lower the probability of unseen words/tags.
- Knowing that every period must map to a period, I made my system always map period to period at the end of sentences.

Conclusion

According to lecture notes, humans can only tag at a rate of about 96-97% accuracy. With my system coming in at about 92%. I know I am getting close to this number. Obviously there are semantic limitations of the Viterbi algorithm, as it cannot understand any sort of sentence context.

It is also limited to a test set that is closely related to the training data, so a sentence cannot be accurately tagged without a large pool of similar sentences.