COMP 150 NLP

Fall 2016

Problem Set 3

POS Tagging with Hidden Markov Models

Mingzhe Li

1. Preprocessing

   This part is very similar to the last time. Just initialize the vocabulary from the training set and do the work. First sentences of the training set and test set are printed out.

   Program output:

   ```
   <S> <UNK> Vinken , 61 years old , will join the board as a
   nonexecutive director Nov. 29 . </S>

   <S> At Tokyo , the <UNK> index of <UNK> <UNK> issues , which
   gained <UNK> points Tuesday , added <UNK> points to <UNK> . </S>
   ```

2. Baseline

   2.1 The most frequent class algorithm is quite simple, or maybe naïve. Just tag every word with its most frequent tag, and tag unknown words as nouns. This algorithm is not perfect, but does give us somewhat satisfactory results.

   2.2 ComputeAccuracy reports the sentence accuracy and tagging accuracy in percentage. The implementation is just compare and count for discrepancy.

   Program output:

   ```
   --- Most common class baseline accuracy ---

   Tagging accuracy: 85.21%.

   Sentence accuracy: 7.00%.
   ```

3. Training

   3.1 Built the BigramHMM class according to the provided template

   3.1.1 I pretty much followed the design of the given template. The train method is also very similar to the last one with bigrams, which does a lot of counting and mapping

   3.1.2 Counted the ambiguous tags and calculated the percentage

3.1.3 Calculated the probabilities in log first and then converted back to normal probabilities. The result is in line with the given sanity check.

Program output:

```
Percent tag ambiguity in training set is 39.54%.

Joint probability of the first sentence is 2.13086363871e-49.
```

4. Testing

4.1 The Viterbi method is quite complicated. Many thanks to the pseudocode, the implementation was not too hard. I noticed a problem, however, is that a lot of path would have zero probabilities. I believe this is because it's without any smoothing, which would have improved the results even more.

4.2 The hidden Markov model brings the accuracy of tagging by 5% and sentence accuracy by 10%. It proves to be very effective.

Program outputs:

```
--- Bigram HMM accuracy ---

Tagging accuracy: 90.03%.

Sentence accuracy: 16.96%.
```

5. Extra Credit:

I built a matrix to count the "confusion," and another function to report the most confused token-tag pair. NN is confused as JJ by 156 times in the testing. NN stands for nouns and JJ stands for adjectives. This happens because nouns are often used as attributive words just like adjectives. A possible solution that I thought of to solve this confusion is to further develop the model using n-grams.

Program outputs:

```
Most confused token:  JJ

True tag should be:  NN

Confused by 156  times

JJ: adjective or numeral, ordinal

      third ill-mannered pre-war regrettable oiled calamitous
      first separable ectoplasmic battery-powered participatory
      fourth still-to-be-named multilingual multi-
      disciplinary ...

NN: noun, common, singular or mass
```

common-carrier cabbage knuckle-duster Casino afghan shed thermostat investment slide humour falloff slick wind hyena override subhumanity machinist ...