Emily Roller

netid: eroller2

Question 1: letter bigram model

a) For my letter bigram model, I choose to allow some punctuation as "letters", namely apostrophes and hyphens. I chose to include these because some languages incorporate these forms of punctuation within their words, and so they can make for some quite distinct bigrams. For example, the bigram ("'", 's'), as in the last two characters of "it's", are much more distictive to the English language than to French. In addition to removing punctuation during my text pre-processing, I was also careful to leave in letters with accents and other diacritics, since those can also be quite distinctive to a particular language. I also took steps to lowercase each line before calculating the probabilities so that capitalizations wouldn't add any repetitive bigram combinations.

b) For out-of-vocabulary bigrams (bigrams that occurred in the testing data but not the training data), I set the probability equal to 1 / total number of bigrams, in accordance with LaPlace smoothing.

c) I found that implementing LaPlace smoothing improved the accuracy by approximately 4% in comparison to using no smoothing technique, from 88.3% to 92% accuracy rates. However, I still found that a letter bigram model, even with the performance increase due to smoothing, performed worse in terms of accuracy than either word bigram model. Personally, if I had to use a letter bigram model I would use one with plus-one smoothing, but if not then I would use a differenet bigram model altogether.

d) My model's output was correct 276/300 times, for an accuracy of 92%.

Question 2: word bigram model (no / LaPlace smoothing)

a) My text preprocessing was the same as that of my letter bigram model. For each line of input text, both for the training and testing files, I kept all letters (including those with accents and other diacritics) and spaces, apostrophes, and hyphens for the same reasons as question 1a. I also converted all input text to lowercase for the same reason.

b) For out-of-vocabulary bigrams, I simply put their probability as 0, following the MLE probability model.

c) I found that implementing LaPlace smoothing for a word bigram model actually worsened accuracy rates. I saw an increase from 95% to 99.7% accuracy rates when I removed my LaPlace smoothing implementation, suggesting that the algorithm is sufficient on its own and doesn't need to rely on plus-one smoothing for higher accuracy. However, plus-one smoothing seems to have too negative of an impact on its performance, though the decrease in accuracy is obviously not ideal.

d) When using plus-one smoothing, the model was accurate 285/300 times. Without it, the model was accurate 299/300 times.

Question 3: word bigram model (Good Turing smoothing)

a) I actually found that using Good Turing smoothing did not improve the accuracy in comparison to using no smoothing technique. However, it did perform better than using the word bigram model with LaPlace smoothing, with a 96.6% accuracy compared to a 95% accuracy. So again, I think that the bigram model can actually be implemented without any smoothing technique, or with add-one smoothing if desired.

b) To implement Good-Turing smoothing, I needed to account for both unseen bigrams and unfrequently seen bigrams in a given line of test input text. To account for both, I iterated through line and kept track of all bigrams that had occurred between 0 and 9 times in the training model. Then, when calculating the probability for any unreliable bigrams (i.e. seen at least once but less than 10 times), I used the formula Prob_GT = ((n+1) * (# bigrams seen n+1 times) / (# bigrams seen n times)) / (total # of bigrams), where n is the number of times the bigram was seen.

c) To account for unseen bigrams, I used the formula Prob_GT = (# bigrams seen once) / (total # of bigrams).

d) With Good-Turing smoothing, the model was accurate 290/300 times.

Question 4:

In summary, I found that a word bigram model using no smoothing performed the best, i.e. had the highest accuracy rate. The letter bigram model without any smoothing had the lowest accuracy, and the letter bigram model with LaPlace smoothing had the second lowest. Both the word bigram model with LaPlace and the word bigram model with Good Turing smoothing were nearly tied for the next most accurate, with Good Turing performing slightly better. Finally, the word bigram model with no smoothing surprisingly beat them all.

I found that the primary disadvantage of the letter bigram model was that the probabilities for each language were often too close. Because of this, a language could falsely be selected as the correct language because its probability was a trivial percent higher than the correct language. I believe this is because some letter bigrams are often not so different between languages, and therefore don't make very good distinguishing features. For example, the following were the probabilities for an English sentence "Madam President , on a point of order ." incorrectly identified as French:

```
English:  2.313098669083296
French:   2.7022255024269404
Italian:  2.5902141714803544
```

The difference in probability between English and French is nearly trivial, but because French was slightly higher, it was incorrectly chosen. Additionally, the main accuracy mistakes made by the letter bigram model were typically between English and French or Italian. My hypothesis is that this is because English borrows many words from French and Latin stems, so there are many letter bigrams shared between English and the other two languages. Using the aforementioned sentence, "Madam" is borrowed from French, and "President" is derived from Latin. When a sentence is only seven words long, it seems that two of those words being closely related to another language can be quite consequential.

I found that the word bigram models, both with LaPlace and Good Turing smooth, much less of the accuracy issues due to repetition between languages. The inaccurate results seemed to be more evenly distributed between all combinations of languages, which I would guess is due to less words being the exact same between any two languages. Therein lies the primary advantage of the word bigram model; since it is less likely that a word or word bigram in one language will also be a word or word bigram in another, there is less similarity in the bigram models for each language, and so the calculated probabilities are more distinct. Therefore, the word bigram models without smoothing, with LaPlace smoothing, AND with Good-Turing smoothing overall had significantly higher performance than the letter bigram models, with or without smoothing.