Akash Chaurasia
achaura1@jhu.edu
JHU CS465 Fall 2019
Homework 5

Problem 4:

Output for vtag using entrain and entest:
Model perplexity per tagged test word: 3924.111
Tagging accuracy (Viterbi decoding): 92.20% (known: 96.30% novel: 49.72%)
Tagging accuracy (posterior decoding): 92.37% (known: 96.25% novel: 52.23%)

The improved tagger increased the model perplexity per tagged word, with a
lambda of 5. This is likely because it biases probabilities in favor of new words and thus sacrifices the probabilities of words it knows. However, since there are many novel words, this improves accuracy compared
to the baseline result.

Increasing lambda definitely increased the novel accuracy while decreasing the known
accuracy, which is a worthwhile sacrifice in this situation, where we have
many novel words that need to be tagged accurately for good overall accuracy.

Also, it seems that model perplexity is not necessarily a good inidcator
of accuracy, since here the perplexity increases (which indicates that the model does not fit the data as well), even though we have a better accuracy.

Problem 5:

Output for vtag_em using entrain25k, entest, and enraw:

```
Model perplexity per tagged test word: 1945.767
Tagging accuracy (Viterbi decoding): 88.07%       (known: 96.46%
seen: 0.00%  novel: 46.54%)
Iteration 0:  Model perplexity per untagged raw word: 1476.6242
Model perplexity per tagged test word: 1765.622
Tagging accuracy (Viterbi decoding): 87.66%       (known: 96.08%
seen: 51.03%  novel: 40.37%)
Iteration 1:  Model perplexity per untagged raw word: 1190.3135
```

```
Model perplexity per tagged test word: 1766.122
Tagging accuracy (Viterbi decoding): 87.13%       (known: 95.43%
seen: 51.46%  novel: 40.06%)
Iteration 2:  Model perplexity per untagged raw word: 1177.5956
Model perplexity per tagged test word: 1770.868
Tagging accuracy (Viterbi decoding): 87.02%       (known: 95.29%
seen: 51.55%  novel: 40.01%)
Iteration 3:  Model perplexity per untagged raw word: 1172.5235
Model perplexity per tagged test word: 1776.063
Tagging accuracy (Viterbi decoding): 86.90%       (known: 95.16%
seen: 51.60%  novel: 39.80%)
Iteration 4:  Model perplexity per untagged raw word: 1169.9494
Model perplexity per tagged test word: 1780.450
Tagging accuracy (Viterbi decoding): 86.83%       (known: 95.08%
seen: 51.69%  novel: 39.75%)
Iteration 5:  Model perplexity per untagged raw word: 1168.5376
Model perplexity per tagged test word: 1783.523
Tagging accuracy (Viterbi decoding): 86.83%       (known: 95.06%
seen: 51.79%  novel: 39.80%)
Iteration 6:  Model perplexity per untagged raw word: 1167.7573
Model perplexity per tagged test word: 1785.416
Tagging accuracy (Viterbi decoding): 86.82%       (known: 95.05%
seen: 51.74%  novel: 39.80%)
Iteration 7:  Model perplexity per untagged raw word: 1167.3303
Model perplexity per tagged test word: 1786.508
Tagging accuracy (Viterbi decoding): 86.83%       (known: 95.05%
seen: 51.84%  novel: 39.85%)
Iteration 8:  Model perplexity per untagged raw word: 1167.0971
Model perplexity per tagged test word: 1787.126
Tagging accuracy (Viterbi decoding): 86.83%       (known: 95.05%
seen: 51.84%  novel: 39.85%)
Iteration 9:  Model perplexity per untagged raw word: 1166.9676
Model perplexity per tagged test word: 1787.483
Tagging accuracy (Viterbi decoding): 86.83%       (known: 95.05%
seen: 51.89%  novel: 39.85%)
Iteration 10:  Model perplexity per untagged raw word: 1166.8933
Model perplexity per tagged test word: 1787.697
Tagging accuracy (Viterbi decoding): 86.83%       (known: 95.05%
seen: 51.89%  novel: 39.85%)
```

a) Figure 2 initializes a###(0) and b###(n) because the model defines these as having a probability of 1 since we are certain that a sequence will start and end with ### as opposed to any other tags.

b) The perplexity per tagged test word is higher than that per untagged raw word because the raw perplexity doesn't include the probability of being

in a specific
state (having a specific tag at that time), but rather includes the probability
of being in any of the states at that time, which will by definition be higher
since it is a sum of probabilities. Thus, the perplexity will be lower due to
this higher probability.

The untagged raw perplexity should be more important because it indicates how
well the model is reestimating its parameters to maximize the likelihood of
the train + raw data; the tagged test perplexity just indicates how surprised
this model is to see such a sequence.

c) V doesn't count the words from test as well because these words aren't being
used to estimate parameters for the model (counts of observed words and tags).
V is used for smoothing, but if a word from test isn't being considered to update
the counts, it's not really considered an observation but rather just an evaluation
of our model.

d) EM resulted in pretty much monotonically non-increasing overall accuracy on
the test data. It resulted in exactly monotonically non-increasing accuracy
on known data, almost monotonically increasing accuracy on seen data (from raw)
and an almost parabolic accuracy on novel words.

e) The EM procedure mainly helped with accuracy on the seen data (from raw). This is because
the raw data is much larger than the train data (almost 100k for raw vs 25k for train),
and thus the model fits the raw data in every successive iteration, which almost
overshadows its initial fitting of the training data. This also explains the mostly
decreasing tagging accuracy on the known words. Regarding the novel words, it likely
initially decreased novel word tagging accuracy because it was reestimating parameters to fit
the seen data, and at first this was pushing it away from being able to accurately tag new words.
However, as it better learned the raw data with successive

iterations, this fit also helped
   it to tag novel words due to similarities between the two sets of words, or at least sequences.

   The EM algorithm got value out of the raw data because it provided a sample dataset
   to refine its counts based on what real data would look like. This results in refined
   probabilities for observed word sequences. Thus, it finds new parameters at each iteration
   that better describes the data it may expect to see in the future.

f) One reason EM didn't always help is because it moves parameters away from those
   that pretty accurately describe the known data (words from train), which
   decreases its accuracy on the known words similar to smoothing. This results in
   a decreased overall accuracy.

   Another reason EM didn't always help is because it is using the training parameters
   to get new counts and reestimate parameters. Thus, the algorithm is only as good
   as the parameters it gets from this training data. This makes sense because we used
   a semi-supervised model, and you need robust training data so that the EM algorithm
   can get reestimate parameters that work well on the test data.

g) One day I probably had a solid 10 scoops of ice cream at Ben & Jerry's. They have
   this nefarious thing called a Vermonster which is probably like 20 scoops of ice cream,
   and naturally my friends let me down so I had to put the team on my back. Honestly, 8/10
   would do again this is the joy of living in a 1st world nation. Definitely didn't get sick
   I doubt there are many types of ailments that 10 scoops of phish food won't cure. Also miss
   me with that weak stuff my parents might not have taught me a lot but they did teach me
   to kill large amounts of nutritionally deficient food when given the opportunity.