Natural Language Processing Homework 6

Katie Chang + Willis Wang

April 19, 2016

README

1 Q1

1.1 a

i. Initially, the probability that day 1 is H was 0.871. With the change, the probability is now 0.491.

ii. With this change, the probability that day 2 is hot is 0.918. This shows a decrease in probability of 0.059. We look at the $p(\rightarrow H)$ column for this information.

iii. Originally with two ice creams on day one, after 10 iterations the probability of the first day being hot decreased ever so slightly, but the probability was always around 1.0 (specifically, p(H) = 1.0 and 0.995 on day 1 and 2, respectively)

After the change to one ice cream on day one, however, the data shows that p(H) in general has decreased significantly. Day 1 p(H) is now 0, and at day 2, p(H) = 0.557.

1.2 b

i. Before the change, the graph shows that days 10-13 had a relatively high chance of being hot.

Day 10: 0.918 Day 11: 0.752 Day 12: 0.856 Day 13: 0.779

After this change, we see that the probability of these days being hot decreases quite a bit.

Day 10: 0.764 (doesn't decrease too much because 3 ice creams were eaten) Day 11: 0 (only one ice cream was eaten) Day 12: 0.441 Day 13: 0.441

Since we're restricting our model such that it can never be a hot day if only 1 ice cream was eaten, then the probability of it being hot on day 11 is 0, which affects everything that comes after it.

ii. After 10 iterations, the probability of it being hot on days 12 and 13 drop to nearly 0.

Before the restriction we made of $p(1|H) = 0$, the graph actually looks pretty much the same as it does after we added this restriction. For day 12, the difference of p(H) from before the change to after is a drop of 0.001.

This is because after 10 iterations, this model should learn that the probability of one ice cream given that the day is hot is very low, regardless of our initial restriction. In fact, without this restriction, at the 10th iteration $p(1|H) =$ 1.6E-04, which is basically 0.

iii. With the change of $p(1|H) = 0$, it still equals 0 after 10 iterations. Based on the excel, $p(1|H) = p(\rightarrow H, 1)$ / $p(\rightarrow H)$ from the previous iteration. But if we initially had set $p(1|H) = 0$, then $p(\rightarrow H, 1)$ is also always 0, so this value is always 0.

1.3 c

i. Since the backward algorithm is analogous to the inside algorithm and the inside algorithm finds the probability of a sentence by summing over all possible parses, this should mean that the β probability of the EOS character sums all possible parses, therefore equaling the total probability of the sentence.

ii. In the tree on the left, the H constituent means that the day is hot. The specific day corresponds to the depth at which the H is at in the tree. The probability of the rule H $\rightarrow$ 1 C is equal to $p(1|C)$ * $p(C|H)$. The probability of H $\rightarrow \epsilon$ is equal to $p(STOP|H)$. The tree on the right may be preferred because instead of having to calculate p(H $\rightarrow$ 1 C) as stated above, we can instead get p(H $\rightarrow$ EC C) from $p(C|H)$, which is already in our excel spreadsheet. Also, p(EC $\rightarrow$ 1) = $p(1|C)$, which is also in the spreadsheet.

2 vtag

Question 2 implemented in vtag.py

3 vtag

Question 3 implemented in vtag.py

4 extra

Question 4 one-count smoothing implemented in vtag.py

Output:

Tagging accuracy (Viterbi decoding): 94.09% (known: 96.81% novel: 65.84%)

Perplexity per Viterbi-tagged test word: 862.377

We got an accuracy improvement of 1.61% over the baseline tagger, an improvement of 0.82% in known words, and an improvement of 9.77% in novel words. Perplexity was improved by 715.122.

5 forward backwards

One-count smoothing is included. Including Question 5, our output is:

Tagging accuracy (Viterbi decoding): 94.09% (known: 96.81% novel: 65.84%)

Perplexity per Viterbi-tagged test word: 862.377

Tagging accuracy (posterior decoding): 94.15% (known: 96.83% novel: 66.41%)

The posterior decoder does slightly better in accuracy, in comparison. There's an improvement of 0.06% in overall accuracy, 0.02% better in known word accuracy, and 0.57% better in novel word accuracy.

6 em

Output:

Katies-MacBook:Homework 6 katiechang$ python vtagem.py entrain25k entest enraw

Tagging accuracy (Viterbi decoding): 94.17% (known: 96.54% seen: 62.86% novel: 65.65%)

Perplexity per Viterbi-tagged test word: 949.669

Iteration 0: Perplexity per untagged raw word: 988.75

Tagging accuracy (Viterbi decoding): 94.28% (known: 96.52% seen: 68.83% novel: 66.84%)

Perplexity per Viterbi-tagged test word: 920.176

Iteration 1: Perplexity per untagged raw word: 612.79

Tagging accuracy (Viterbi decoding): 94.19% (known: 96.53% seen: 68.93% novel: 65.59%)

Perplexity per Viterbi-tagged test word: 915.686

Iteration 2: Perplexity per untagged raw word: 608.62

Tagging accuracy (Viterbi decoding): 94.20% (known: 96.54% seen: 68.93% novel: 65.70%)

Perplexity per Viterbi-tagged test word: 914.724

Iteration 3: Perplexity per untagged raw word: 608.00

Tagging accuracy (Viterbi decoding): 94.21% (known: 96.55% seen: 68.64% novel: 65.70%)

Perplexity per Viterbi-tagged test word: 914.448

Iteration 4: Perplexity per untagged raw word: 607.79

6.1 a

α###(0) is initialized to a 1 because we have to start with three pound signs. All paths from the start have to end in the state of three pound signs, so that must have a probability of 1.

β###(n) also has to end with three pound signs regardless of where it comes from, so the probability of all paths from n to n has to be 1.

6.2 b

Viterbi-tagged word perplexity is higher because that perplexity is calculated taking into account the sentence's tags. Posterior perplexity does not account for tags, and therefore variations of a sentence (where one sentence can have different tags) are ignored and therefore perplexity is lower.

6.3 c

We were never to use the test data to train. We should be able to use training data to train our model, and then use that model to give us results with the test data. If we trained on test data, and then actually tested on test data, we would not run into OOV situations, which is counterproductive with regards to test data.

6.4 d

After 5 iterations, all accuracy improved slightly, however with fluctuation.

Overall accuracy went from 94.17% to 94.21%, with a high point of 94.28%.

Overall known word accuracy went from 96.54% to 96.55%.

Overall seen word accuracy went from 62.86% to 68.64%, with a high point of 68.93%.

Overall novel word accuracy went from 65.65% to 65.70%, with a high point of 66.84%.

6.5 e

Just like the excel files provided in class show, iterating over all known data improves the predictions. After five iterations, our overall accuracy did improve. EM utilizes the training data and gets initial probabilities to make predictions on the raw data file. This also helps with improving accuracy.

6.6 f

The results will depend on the initial probabilities and initial data that the model gets to train on. If any of the data files (training, raw, testing) are too different from the other two, then we lose the benefit of using the raw data to help improve predictions.

EM also relies on smoothing the data. If we don't smooth correctly, we could overfit or underfit the data and incorrectly predict probabilities.

6.7 g

More notably, Willis has eaten a crap ton of ice cream before, without getting sick (or so he says). Katie has only eaten a maximum of maybe two scoops at once (what a wimp).

Used overleaf.com to generate LaTeX document.