CS4248 Assignment 2

Li Xiangqun A0091794R

1. Introduction

This assignment requires us to build a Part-of-Speech (POS) bigram tagger, which is based on a hidden Markov model. The Viterbi algorithm will be used to find the optimal sequence of most probable tags. We are provided with a training set to gather all statistics that we need to build the tagger. During the training process, some smoothing method will be used to handle zero probability. With all the statistics, the tagger is supposed to tag the test data with Penn Treebank tags.

The training set "sents.train" contains 39832 sentences. Each word is annotated with a correct Penn Treebank Tag. There are totally 45 Penn Treebank tags. In order to handle zero probability, I used three smoothing methods: Witten-Bell smoothing, Add-One smoothing, and One-Count Smoothing.

In the following sessions, I will describe in details the training process and the performance of three smoothing methods.

2. Training

Since we are using Hidden Markov Model, the transition probability and emission probability should be collected to calculate the best tags combinations. The following is the data I collected.

Data Name	Description
$C(t_{i-1}\ t_i)$	The number of occurrence of $Tag_{i-1}\ Tag_i$
$C(w_i\ t_i)$	The number of occurrence of $Word_i\ Tag_i$
$C(t_i)$	The number of occurrence of Tag_i
$T(t_i)$	The number of seen word types followed by Tag_i
$S(t_i)$	The number of singletons (occur once) followed by Tag_i
V	Total number of word types
N	Total number of word tokens

During the processing, I change all the instances of numeric tokens into "#NUM#", which match the format `"-?\\d+(\\.\\d+)?"`. Most of the numeric tokens have different forms, but they should be treated as one single type. The following shows the contrast:

	Word Types	Singletons (occur once)
Before change to "#NUM#"	44389	20622
After change to "#NUM#"	40722	18484

This reduces the many different combinations of numbers down to a common type that have higher probability given the tag "CD".
With all the data collected, we can now perform different smoothing methods.

3. Smoothing and Evaluation

Transition probability smoothing

Since the tag types are closed and the number of types is only 45, I only apply Witten-Bell smoothing for transition probability.

$$p(t_i|t_{i-1}) = \frac{C(t_{i-1}\ t_i)}{C(t_{i-1}) + T(t_{i-1})} \quad if\ C(t_{i-1}\ t_i) > 0$$

$$p(t_i|t_{i-1}) = \frac{T(t_{i-1})}{(V - T(t_{i-1}))(C(t_{i-1}) + T(t_{i-1}))} \quad if\ C(t_{i-1}\ t_i) = 0$$

Emission probability smoothing

For emission probability smoothing, I used three smoothing methods: Add-One smoothing, Witten-Bell smoothing, and One-Count Smoothing.

One-Count Smoothing[1]

Since the last one is not taught in class, I will briefly describe the idea. It is basically just add-λ smoothing with back-off, but λ is set higher in contexts with a lot of "singletons" words (that have only occurred once), because such contexts are likely to have unknown words in test data. This is called One-Count smoothing.
First define the back-off estimate:

$$P_{wt-backoff}(w_i|t_i) = P_{w-addone}(w_i) = \frac{C(w_i) + 1}{N + V}$$

This back-off estimate uses add-one-smoothing. N and V denote the number of word tokens and types respectively. According to this formula, any unknown word has back-off probability $\frac{1}{N+V}$.

$$P_{wt}(w_i|t_i) = \frac{C(w_i,\ t_i) + \lambda * P_{wt-backoff}(w_i|t_i)}{C(t_i) + \lambda}$$

$$where\ \lambda = S(t_i)\ number\ of\ singleton\ words\ tagged\ by\ t_i$$

Here λ for "NN Tag" will be high, because many nouns only appear once. This suggests that the class of nouns is open to accepting new members and it is reasonable to tag new

[1] Appendix of http://www.cs.jhu.edu/~jason/465/hw-hmm/hw-hmm.pdf

words with “NN Tag” too. By contrast, λ for “DT Tag” will be 0 or very small because the class of determiners is pretty much closed.
But if $\lambda = 0$, there are no singleton words followed by tag_i , we still get zero probability. In order to avoid this problem, add a small value to λ before using it. Here I simply add 1.

Unknown Words

	Witten-Bell	Add-One	One-Count
$p(unkown\|t_i)$	$\frac{T(t_i)}{(V - T(t_i)(C(t_i) + T(t_i))}$	$\frac{1}{C(t_i) + V}$	$\frac{S(t_i) * \frac{1}{N+V}}{C(t_i) + S(t_i)}$

Problems

When computing the probability in Viterbi algorithm, the probability may become negative due to small numbers multiplication. To overcome this problem, I use log probability instead in the actual computation, as suggested in lecture.

Evaluation

$$Accuracy = \frac{C(correct)}{C(total)}$$

10-fold cross validation accuracy on training set:

	Witten-Bell	Add-One	One-Count
1	0.948573	0.923438	0.950582
2	0.948346	0.922976	0.950514
3	0.951874	0.926482	0.953686
4	0.951559	0.926515	0.953687
5	0.94868	0.922827	0.951312
6	0.951001	0.925027	0.953147
7	0.947234	0.924309	0.949476
8	0.947945	0.924266	0.951048
9	0.953913	0.930630	0.955701
10	0.948537	0.922879	0.950207
Average	0.949766	0.924935	0.951936

Witten-Bell Smoothing and One-Count Smoothing both have much better performance than Add-one smoothing. This suggests that Add-one Smoothing move too much probability mass to zero bigrams.
One-Count Smoothing has a slightly better performance than Witten-Bell, since One-Count Smoothing distribute probability differently for different Tags.

4. Conclusion

With the evaluation result, I will select One-Count Smoothing as the smoothing methods. However, the accuracy is still rather short of 97%. This may because that the training set is still relatively small; we are only using bigrams instead of trigrams; and we didn’t using any Morphological suffixes to handle unknown words.