1 Architecture

1.1 Hidden Markov Model

First order hidden Markov model (i.e. *bigram* model) is used to model the sequence of tags in a sentence. Transition probability $P(t_i|t_{i-1})$ and observation likelihood $P(w_i|t_i)$ are calculated from the training data. These sequence of states (i.e. tags) respresents the hidden part of the Markove model. Using these 2 probabilities , the objective is to find the following sequence of tags T:

$$T = \arg\max_t \left\{ \left(\prod_{i=1}^{T} P(t_i|t_{I-1}) \cdot P(w_i|t_i) \right) \cdot P(</s>|t_T) \right\}$$

The *Viterbi algorithm* is used to find such sequence of tags.

1.2 Smoothing

The *sparse data problem* means not all combination of bigram sequence will be seen in the training data. Consequently when tagging new data, there will be cases where the bigrams have not been seen before. Setting such probability to zero will cause undesirable effect because the probability of the whole sequence will become zero.

1.2.1 Absolute discounting

This method works by subtracting a fixed fraction from the maximum likelihood estimate. For counts with high value, subtracting a small discount will not affect the probability value by much. For counts with low value, even though discount might shift the value by much more, it is still quite acceptable since those with low counts are normally not as important as those with high counts.

$$P_{absolute}(w_i|w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})} & if\ C(w_{i-1}w_i) > 0 \\ \alpha(w_i) P_{absolute}(w_i) & otherwise \end{cases}$$

1.2.2 Minimum probability

A bigram sequnce that has never been observed may approximates the bigram sequence that *occurs only once*. Hence, for these unobserved bigram, we can assign it the same probability value as the probabiliy singly occuring bigram. Some weight may be used to ensure that the sum of probability is still 1.

1.2.3 Constant value

Another possible approach is to assign a constant value to the probability of unobserved bigram. An approximate good value can first be obtained by looking at the distribution of low count bigrams and then from *trial and error* more values can be used to find the better one.

1.3 Handling unkown words

Since the whole corpus is not available in this assignment, we are dealing with the case of *open vocabulary* where new words may be encountered during testing (Of course, we can try to use all words from English dictionary as the corpora, but even then there is no guarantee that there will be no new words based on the information in the assignment).

1.3.1 Using affixes

The textbook suggests using affixes to handle unknown words since English is an *inflectional* language such that prefix and suffix can be used as a good indicator of POS (e.g. *–able* is usually an adjective). This approach can be based on rule created at the start for instance unkown word ending in *–ation* is a NN or based on the probabiliy of these affixes given each tag in the data. The later is used in this case, where $P(affix|tag)$ is collected from each word.

A *normalised sum of the probability* of matched affixes is used as an estimate of unknown word probability

$$\sum_{i=2}^{4} \frac{P(affix_{len=i}|tag)}{4^{(4-i+5)}}$$

The number in the denominator (why 4 power (4-i+5) is used) is simply a good value found by trial and error.

2 Evaluation

Performance of the model is measured by the percentage accuracy, the proportion it gets the correct tag, on the training data split using *10-fold cross validation* (the accuracy is the *average* of the 10 tests).

Smoothing	Parameter	Accuracy
Fixed value (Assign p of unkown bigram to this value)	1.00E-05	0.92000
	1.00E-06	0.93254
	5.00E-07	0.93317
	1.00E-07	0.93329
	1.00E-08	0.93317
Minimum probability		0.92218
Absolute discounting (not tuned such that ∑p=1)	D=0.75 α=0.5	0.93272

Unknown words	Affix length	Normalisation	Accuracy
Affix probability	2 <= len <= 5	$4^{(4-i+5)}$	0.93728
		$4^{(4-i+3)}$	0.89702
	3 <= len <= 5	$4^{(4-i+5)}$	0.93811
		$4^{(4-i+3)}$	0.90978
	3 <= len <= 6	$4^{(4-i+5)}$	0.93624
		$4^{(4-i+3)}$	0.89382
For trials, here, the smoothing used is fixed value with param 1.00E-7 because it appears to perform best			

3 Conclusion

Hidden Markov model appears to be a good model for predicting POS tag, with accuracy of more than 90 % for reasonably tuned smoothing and unknown word parameters.

Of particular interest is the fact the smoothing unknown bigram by assigning it a *fixed value* appears to perform best in this implementation. This, I believe, is largely due to *improper implementation* of absolute discount smoothing, particularly the assigning of value of D and α such that the sum of P is always 1.

The use of *affixes* to estimate the probabily of unkown words also appear to be beneficial, giving the *highest accuracy* on the experiment. This most likely is because the *inflectional* nature of English language, such that word with certain affixes have more probability of having a particular POS tag.

More validation needs to be done to confirm the result. Perhaps by using the *development set* to tune the parameter rather than the training set (so that the data set used to tweak and test/train are independent). Even though literature have shown interpolated Kneser Ney smoothing to perform best in most cases, it is good to try out compare different smoothing methods such as Turing approximation.

4 References

1. Speech and Language Processing 2nd edition (2007) chapter 4 and 5, Jurafsky and Martin.
2. TnT – A Statistical Part-of-Speech Tagger, Brants T.