# Trigram HMM Taggers for English and Chinese

**[Mengmei Chen]**
[New York University]
[mc4522@nyu.edu]

## Abstract

This paper presents a trigram HMM system I developed for English and Chinese part-of-speech tagging. I used a combination of unigram, bigram and trigram to model the prior probabilities and exploit different features of the words to model the emission probabilities of unknown words. My model for English achieves an accuracy of 96.53% and my model for Chinese achieves an accuracy of 90.84%.

## 1 Introduction

In natural language processing, part-of-speech tagging is the process of assigning each word in a sentence of the corpus a corresponding part of speech, usually based on the definition and context of the word. It is indispensable to natural language processing because it is the foundation of many other advanced language-processing tasks. For example, accurate tagging results can help alleviate the syntactic ambiguity problem in parsing.

However, it is not as easy as it seems to be. The difficulty lies in the fact that words can possess many different meanings and part of speeches depending on the context. For example, "does" is usually instantly perceived to be a verb even by human beings, but can actually also represent female deer, which is a noun. Inspired by this problem and the bigram we built for homework 4, I went forward and built a trigram system for the English part-of-speech tagging problem. As I discovered the Chinese part-of-speech tagging is different and even more challenging, I also built a model for Chinese.

## 2 Data

### 2.1 Penn Treebank WSJ Corpus

For the English tagging problem, I used the Penn Treebank WSJ Corpus. The training set contains 39832 sentences and 950028 tags. The validation set contains 1346 sentences and 32853 tags. The test set contains 2416 sentences and 56684 tags.

| | Training | Validation | Test |
|---|---|---|---|
| Sentences | 39832 | 1346 | 2416 |
| Tags | 950028 | 32853 | 56684 |

### 2.2 Penn Chinese Treebank Corpus

For the Chinese tagging problem, I used the majority part of the Penn Chinese Treebank Corpus. I preprocessed the corpus, which contains many HTML tags, to have the exact same format as the WSJ corpus. The training set contains 18784 sentences and 508613 tags. The validation set contains 786 sentences and 23626 tags. The test set contains 659 sentences and 21557 tags.

| | Training | Validation | Test |
|---|---|---|---|
| Sentences | 18784 | 786 | 659 |
| Tags | 508613 | 23626 | 21557 |

## 3 The Bigram HMM model

In homework 4, we implemented a bigram HMM POS tagger, which will be used as my baseline system. It has two important assumptions:

a. If the probability of a word is only dependent on its own POS tag
b. If the probability of a POS tag is only dependent on its previous POS tag

With these two assumptions, it simplifies our goal. Given the word sequence $w_1, ..., w_n$, to find

the tag sequence $t_1, \ldots, t_n$, instead of maximizing $P(t_1, \ldots, t_n | w_1, \ldots, w_n)$, we can now maximize:

$$\prod_{i=1,n} P(t_i|t_{i-1}) * P(w_i|t_i)$$

The model uses bigram for both prior and emission probabilities in the formula.

It uses Viterbi to find out the maximum likelihood and the most likely. Let *Viterbi* be the matrix that stores the maximum likelihood where $Viterbi(q,t)$ represents the maximum likelihood of the current sequence that ends in the word with tag q at position t. Let A be the matrix of prior probabilities where $A[\mathrm{i},j]$ represents the probability of transition from tag i to tag j. Let B be the matrix of emission probabilities where $b_q(w_t)$ is the probability that POS tag q is realized as word $w_t$. Then, after initialization, for each given word w at position t and for each possible tag q for the word,

$$Viterbi[\mathrm{q},\mathrm{t}] = \max_{q'=1,n} Viterbi[q',\mathrm{t}-1] * A[q',q] * b_q(w_t)$$

In the end, it will find the maximum result and the tag sequence along with it by tracing back the backpointers that store the optimal tag in each recursive step.

To deal with out-of-vocabulary items, it treats all unknown words as one big unknown word. When it encounters an unknown word, to estimate the probability of an instance of this unknown word given a tag, it uses the probability of words occurring only once given that tag. For example, if we observe 50K NN in the corpus, of which 1K words occur only once. Then when the model gets a new word *dljfggb*, it estimates *P(dljfggb|NN)* to be 1K/50K = 0.02.

# 4 The Trigram HMM model

## 4.1 English

First of all, I built the trigram model for the prior probabilities. The model now assumes that the probability of a POS tag is only dependent on its previous two tags, instead of one. Because of this fundamental change, the recursive function becomes:

$$Viterbi[\mathrm{t},\mathrm{u},\mathrm{v}] = \max_{q'=1,n} Viterbi[\mathrm{t}-1,\mathrm{w},\mathrm{u}] * A[\mathrm{w},\mathrm{u},\mathrm{v}] * b_v(w_t)$$

in which $Viterbi[\mathrm{t},\mathrm{u},\mathrm{v}]$ represents the maximum likelihood of the current sequence that ends in the word at position t with tag v and previous tag u, $A[\mathrm{w},\mathrm{u},\mathrm{v}]$ represents the transition probability from tags w and u to v, and $b_v(w_t)$ represents the probability that tag v is realized as word $w_t$. Similar to bigram, the model will generate the maximum likelihood for the whole sentence and the tag sequence along with it using recursion and backpointers.

I then tried to use trigram on the emission probabilities as well but it actually decreased the accuracy. Besides the standard left-to-right emission probability, I also tried to incorporate the right-to-left emission probability, which uses the probability of a word given its tag and its next tag, but it also decreased the accuracy. These results show that the probability of a word is probably only dependent on its own POS tag in general.

Because of the complexity of trigram, many of the counts in the model will be zero. For example, to find out the probability of a NN given TO and VB, the model calculates it as the count of TO VB NN over the count of TO VB. However, TO VB NN is very likely to have not appeared in the training corpus, thus making the count zero and decreasing the accuracy of the model. To solve this problem, I used a combination of unigram, bigram, and trigram, so that even when the trigram count is 0, the model can still use the unigram count and/or the bigram count to estimate the probability. The formula is as follows:

$$P(w_t|w_{t-1}w_{t-2}) = \lambda_1 P(w_t) + \lambda_2 P(w_t|w_{t-1}) + \lambda_3 P(w_t|w_{t-1}w_{t-2})$$

in which $w_t$ is the word at position t and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. And then I tuned the three lambdas to get the highest accuracy on the validation set.

I also used various features of the words to estimate emission probabilities for unknown words. Such features include whether the word is a number, whether the word's first letter is capitalized when it's not at the beginning of a sentence, and the endings of a word. The intuition is that a word with certain patterns is more likely to be associated with some tags than with others. For example, a word that ends in "ly" is more likely to be an adverb as supposed to a noun. For each unknown word, for each possible tag, the model calculates

$$P(unknown\ word|tag) = \frac{P(unkonwn\ word, tag)}{P(tag)} = \frac{Count\ (unknown\ word, tag)}{Count(tag)}$$

$$= \frac{Count\ (words\ with\ similiar\ patterns, tag)}{Count\ (tag)}$$

In particular, if a word is neither a number of a capitalized word, then to use the endings of the word to estimate the emission probability, I used the following equation:

$$P(unknown\ word|tag) = \frac{\begin{array}{c}\lambda_1 Count(words\ with\ the\ same\ last\ letter, tag) \\ +\lambda_2 Count(words\ with\ the\ same\ last\ two\ letters, tag) + \\ \lambda_3 Count(words\ with\ the\ same\ last\ three\ letters, tag) \\ +\lambda_4 Count(words\ with\ the\ same\ last\ four\ letters, tag)\end{array}}{Count(tag)}$$

and I tuned the four lambdas in the equation to get the highest accuracy on the validation set.

To avoid assigning a high probability to an unknown word given a closed class such as TO, I only counted the word when it appears less than 10 times in total in the training corpus.

If the emission probability is still zero after examining all the above features, I assign it the average of the probability of words occurring only once given its tag and the probability of words occurring only once given its tag and its previous tag.

### 4.2 Chinese

In the model for Chinese, I used the same trigram method for prior probabilities and re-tuned the lambdas to get the highest accuracy.

To deal with unknown words in Chinese, several changes were made to reflect the differences between the English and Chinese language. While the first letters of proper nouns are capitalized in English, they are not in Chinese, so the feature of capitalized first letter is removed. In addition, when Chinese words contain numbers, depending on the other characters they contain other than the numbers, they have different part of speech tags. Accordingly, I split the feature of number into two categories and treat them as separate features to examine.

For the endings of the words, I only used the last one or two characters because Chinese words are usually much shorter than English and do not normally exceed four characters.

## 5 Results

On English, the bigram HMM model achieves a validation accuracy of 95.18% and a test accuracy of 95.62%. On Chinese, it achieves a validation accuracy of 86.93% and a test accuracy of 89.77%. These are used as my baseline accuracies.

The trigram HMM model improves both English and Chinese accuracy. On English, it achieves a validation accuracy of 96.32% and a test accuracy of 96.53%. On Chinese, it achieves a validation accuracy of 89.74% and 90.84%.

**English**

| | Validation accuracy | Test accuracy |
|---|---|---|
| Baseline | 95.18% | 95.62% |
| Trigram | 96.32% | **96.53%** |

**Chinese**

| | Validation accuracy | Test accuracy |
|---|---|---|
| Baseline | 86.93% | 89.77% |
| Trigram | 89.74% | **90.84%** |

## 6 Conclusion and Future Work

In conclusion, the trigram HMM model I developed was successful. It improved the accuracy of tagging on English from 95.62% to 96.53%, which is very close to the accuracy of the state-of-the-art trigram, 96.7%. It improved the accuracy of tagging on Chinese from 89.77% to 90.84%, while the accuracy of the state-of-the-art trigram is around 94%.

To further increase the accuracy on English, I can choose the lambdas automatically by using some advanced algorithms such as the deletion interpolation algorithm.

To further increase the accuracy on Chinese, several things can be done. First, instead of using only parts of the Penn Treebank as my corpus, I can and should use all of it. This will add more data to my training corpus, which will give a more accurate result. Secondly, I can also include the segmentation task as part of my part-of-speech tagging problem. Lastly, I can use some other advanced methods to deal with unknown Chinese words based on the unique aspects of them. For example, while in English, it is usually the endings of a word that imply its tag, in Chinese, all characters in a word may be important to identify its tag, so a geometric average of the emission probabilities of the characters of words might be helpful.

## References

[1] Zhongqiang Huang, Mary P. Harper, Wen Wang. Mandarin Part-of-Speech Tagging and Discriminative Reranking.

[2] Michael Collins. Tagging with Hidden Markov Models. http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/hmms.pdf