CS4248 Assignment 2
Raymond Hendy Susanto
U096940N

1. Introduction

Part-of-speech (POS) tagging is the task of assigning the correct parts of speech to words in a text (corpus). POS tagging is useful in a large number of applications. It is useful as a preprocessing step of syntactic parsing. It is widely used in text-to-speech (TTS) systems to help disambiguate words that have identical spellings but have different pronunciations and meanings. For instance, the word *bow* may be a verb meaning "to bend in respect" or a noun meaning "a stringed weapon". They are pronounced differently, thus we need to know which word is being used in order to pronounce the text correctly.

Here we describe the implementation of a POS bigram tagger based on a hidden Markov model. We also focus on how to deal with words that are not encountered during the training of the tagger. We will adopt the Penn Treebank tagset in this assignment.

2. HMM Tagging

This section describes how to use Hidden Markov Model (HMM) in part-of-speech tagging. Following Jurafsky and Martin (2008, p. 23), a HMM is specified by the following components:

$Q = q_1 q_2 \cdots q_N$	a set of N **states**
$A = a_{11} a_{12} \cdots a_{n1} \cdots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \; \forall i$
$O = o_1 o_2 \cdots o_T$	a sequence of T **observations**, each one drawn from vocabulary $V = v_1 v_2, \cdots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**; also called **emission probabilities**, each expressing the probability of observation o_t being generated from a state i.
q_0, q_F	a special **start state** and **end (final) state** which are not associated with observations, together with transition probabilities $a_{01} a_{02} \cdots a_{0N}$ out of the start state and $a_{1F} a_{2F} \cdots a_{NF}$ into the end state.

Training a hidden Markov model consists of finding the transition probabilities A and the emission probabilities B. Given this model and a sequence of observations, we can decode the best sequence of states by using **Viterbi** algorithm.

In part-of-speech tagging, the hidden states above are the POS tags and the observations are the words found in the text. Here we describe the computation of both transition probabilities and emission probabilities for our HMM.

2.1 Transition probabilities

With POS tags as the hidden states, a transition probability a_{ij} from a state *i* to state *j* is the probability of a POS tag t_i following POS tag t_j. We can use maximum likelihood estimate to find this probability, simply by counting occurences in our tagged training corpus as given by the formula below.

$$a_{ij} = P(t_j|t_i) = \frac{C(t_i t_j)}{C(t_i)}$$

Given a Penn Treebank tagset consisting of 45 tags, we need to estimate 45^2 = 2,025 different transition probabilities. Without a sufficiently large training data, it is likely that less probable POS bigram sequences are not found in the training data, and consequently receive a zero probability. Any tag sequence that contains unobserved bigram will also get a zero probability and will only be output if the tagger finds no other alternative. This *sparse data problem* is solved by parameter smoothing, where a small fraction of the probability mass is redistributed from unobserved events to observed events. In this assignment, we tried two different smoothing methods: Laplace (add-one) smoothing and linear interpolation by deleted interpolation.

2.1.1 Laplace smoothing

In Laplace smoothing, we simply assign a count of 1 to unseen bigrams. The transition probability formula is then modified as follows:

$$a_{ij} = P(t_j|t_i) = \frac{C(t_i t_j) + 1}{C(t_i) + |T|}$$

where $|T|$ is the number of possible POS tags in Penn Treebank tagset, i.e. 45.

2.1.2 Linear interpolation

In linear interpolation, we estimate the probability $P(t_j|t_i)$ by a weighted sum of the bigram and unigram probabilities:

$$P(t_j|t_i) = \lambda_1 \hat{P}(t_j|t_i) + \lambda_2 \hat{P}(t_j)$$

To set the λs, we use a Brants (2000) version of the deleted interpolation algorithm for tag bigrams. In deleted interpolation, we successively delete each bigram from the training corpus, and choose the λs so as to maximize the likelihood of the rest of the corpus (Jurafsky and Martin, 2008, p. 29).

2.2 Emission probabilities

The emission probability of an observation o_t from a state *i* is the probability of generating a word w_t at time *t*, given a POS tag t_i. Similar to the calculation of transition probability, the maximum likelihood estimate for the probability is given by the formula below:

$$b_i(o_t) = P(w_t|t_i) = \frac{C(t_i, w_t)}{C(t_i)}$$

Another problem arises if a word is not found in the training data, since it will result in all state sequences for an observation containing an unknown word to have a zero probability. We describe some possible approaches for estimating the emission probabilities for unknown words below.

Most probable tag
A very simple approach to handle unknown words is to assign the POS tag that appears the most frequently in the corpus. We then assume that all unknown words have this POS tag.

Hapax legomena
The above method is obviously flawed since it assumes that all unknown words have the same POS tag. A better option is to put unknown and rare words into a class that is treated like a single word. Any word that is occuring only once in the training set (*hapax legomena*) is replaced with a special token "**<unk>**", then the parameters can be estimated as usual. During tagging, we also replace any word that does not appear in the training data as "**<unk>**".

In addition, we also include a simple cardinal number ("**CD**") detection in our tagger. Other possible approaches to estimating the probability distribution of unknown words, but not implemented in this assignment, includes the use of the morphology information in the words. For example, words ending with *able* tend to be adjectives, while words ending with *ness* tend to be nouns. Recent approaches to unknown word handling are based on maximum entropy models, which combines several features of a word (such as its morphology, numbers, letter case, etc.) into a log-linear model.

3. Experiment Results

For our bigram POS tagger implementation, we have used all the provided files to perform training, tuning, and testing. We evaluate the performance of our tagger by using 10-fold cross-validation, where we randomly partition the training data into 10 parts. Of the 10 parts, a single part is retained as the validation data for testing the tagger, and the remaining 9 parts are used as the training data. The cross-validation process is repeated 10 times, with each part used exactly once as the validation data. For each validation data, we try different combinations of smoothing techniques and methods to handle unknown words.

The POS tagger is evaluated using *accuracy*, i.e. the number of correctly tagged words over the total number of words in the test data. Table 1 depicts the result of the experiments with different combinations of smoothing and unknown words handling. The average accuracies over the 10 runs are reported in Table 2.

runs	unknown word handling	Laplace smoothing	linear interpolation
1	Most probable tag	94.50%	94.54%
	Hapax legomena	95.67%	95.69%
2	Most probable tag	94.46%	94.50%

	Hapax legomena	95.62%	95.62%
3	Most probable tag	94.49%	94.56%
	Hapax legomena	95.69%	95.67%
4	Most probable tag	94.42%	94.45%
	Hapax legomena	95.63%	95.62%
5	Most probable tag	94.57%	94.59%
	Hapax legomena	95.66%	95.65%
6	Most probable tag	94.44%	94.50%
	Hapax legomena	95.67%	95.66%
7	Most probable tag	94.46%	94.56%
	Hapax legomena	95.58%	95.62%
8	Most probable tag	94.52%	94.58%
	Hapax legomena	95.64%	95.65%
9	Most probable tag	94.53%	94.60%
	Hapax legomena	95.68%	95.67%
10	Most probable tag	94.49%	94.54%
	Hapax legomena	95.66%	95.68%

Table 1: **Evaluation results.** Shown are accuracies of the tagger with different combinations of smoothing techniques and methods for handling unknown words in each run.

unknown word handling	**Laplace smoothing**	**linear interpolation**
Most probable	94.49%	94.54%
Hapax legomena	95.65%	95.65%

Table 2: **Average accuracies** after 10 runs for each combination of smoothing techniques and methods for handling unknown word.

As expected, our approach to modeling unknown words based on *hapax legomena* in the training data outperforms the naïve approach by assuming that every unknown word has the same POS tag. It is also worth noticing that linear interpolation by deleted interpolation algorithm does not help much in improving the overall performance of the tagger, perhaps due to the fact that the sparse data problem in the training data is not as worse as if we have used a higher order grams for our tagger, e.g. trigram, where the use of better smoothing techniques is crucial.

4. Conclusion

We have implemented the bigram POS tagger based on hidden Markov models. The best average accuracy in our 10-fold cross-validation is 95.65%, where the unknown words are modeled based on *hapax legomena* in the training data, and both smoothing methods achieve almost the same performance.

References

Jurafsky, Daniel & Martin, James H. 2008. *Speech and Language Processing, An Introduction to Natural Language Processing,Computational Linguistics, and Speech Recognition*. (2nd ed.). New Jersey: Prentice-Hall, 2nd edition.