Part-of-Speech Tagging with HMMs

Mahsa Shafaei
University of Houston
user in CodaLab: mahsa
mahsa.shafaei@gmail.com

1 Introduction

Part of speech tagging is a prediction process in which all words in a sentence will be assigned by a corresponding part of speech tag. There are several methods to do this task automatically. According to (Adhvaryu and Balani) we can categorize tagging methods into two high-level groups. Supervised and unsupervised methods. In supervised methods, we have a pre-trained dataset in which all the words are assigned by a correct tag. On the other hand, in the dataset for unsupervised methods we do not have corresponding tags. As a result, we have to train on the whole dataset then tag the input sentence. This process makes unsupervised methods more difficult than supervised ones. Also, we can split each of these methods to three other sub-methods: Rule-based Technique, statistical technique and transformation based technique.

There are several challenges which made POS tagging problem a difficult task. Having ambiguous words is one of them; this issue refers to words which have different roles in a certain context. For example, "work" can be both noun and verb. Another challenging factor is new words or a new role for old words. We build our model based on the words and their tags in the training dataset. Thus, having a new word or new role for an old word in the test dataset may produce an error in the result.

In this study, as we have tags for words in the training dataset, we use a supervised method. To predict the tags and evaluate our result, in the first step, we train our model based on the training dataset. Then we run it on development dataset and tweak the parameters and improve the features. In the final step, we apply our model to the test dataset and compare the result with the gold dataset. The best accuracy we accomplished is 0.9145% for the test dataset.

2 Dataset:

Our data set was divided into 3 parts: train, development and test. In train and development datasets each line consists of three columns. The first column is a word, the second one is language tag for the word and the last one is corresponding POS tag. Test dataset has the same structure but does not include the third column. Table 1 shows details of the datasets.

	# words	**# sentence**
train	298720	30214
dev	48799	6397
test	45980	6300

Table 1: Dataset Statistics

3 Methodology

In this work, we use the Viterbi algorithm which is a statistical technique. Viterbi algorithm is used for finding the most likely tag sequence for each input sentence. For this method, we need three probability value. First, we need to find the probability of transition from the previous tag to the current tag, second calculate the probability of having a specified word for the current tag and finally the probability of the previous state. Then we should select a tag for the input word which maximizes the multiplication of three aforementioned parameters. Also, in this method we consider very small amount value for unseen tag sequences instead of zero.

We implemented Viterbi algorithm as a baseline system then improved it by different methods. In order to improve the accuracy we added these steps: (1) deciding about unknown words , (2) employing language tags and (3) calculating trigram probabilities rather than bigram for the Viterbi al-

gorithm in order to decrease errors in ambiguous words' tag.

3.1 Handling Unknown Words

As we explained in the section 1, there are some words in the test which do not exist in the train set. The first and easiest solution for this problem is assigning the tag "Noun" or most probable tag to these words. On the other hand, we can apply a wiselier solution. What we do in our system is using the suffix of the word to decide what is the most probable tag for this input.

3.2 Using Language Tag

Our data set is a combination of English and Spanish words. Thus, some errors in tag prediction occur due to the different tags for the same word in English and Spanish. To check if this problem can be smoothed or not, we tried new definition for tags in our system by combining language and POS tag. With the help of this technique, we could discriminate between words in different languages. For example, we have different transition probability for changing the state from (Spanish, Verb) to (English, Noun) compared to changing the state from (English, Verb) to (English, Noun).

3.3 Using Trigram HMM Tagger

For the baseline system, we used bigram probability in the Viterbi algorithm. As we explained, for implementing Viterbi with bigram probability table, we need to calculate the probability of transition from the previous tag to the current tag. e.g. we want to estimate the probability of transition from noun to verb. In other words, if we have a noun in the previous state with what probability we will have a verb in the current state. However, in trigram method, we should keep track of two last states. For example, if we have noun and verb in two last states, what is the probability of having an adverb in the current state. This method in some cases can help us to solve the problem of ambiguous words. As an example, the word "like" can be an adverb or a verb if we only check the previous word, but if we check the two words before "like" we may predict the correct tag.

4 Experimental Results

In the baseline method, we use the bigram HMM algorithm, the accuracy of this version is 0.873%. Then, by adding some features to the model we reach to better accuracy.

For handling unknown words, we use words' suffix to decide what is the more probable tag for this input. For example, we assign "VERB" tag to words which end with "ing" or "ed". We also tried other patterns, but some of them had a reverse effect on the accuracy. e.g. tag all words ends with "ly" as an "ADV". So, we only used patterns that increased our result and tag other words with "NOUN". This technique helps us to achieve the accuracy of 0.891%.

Checking the prediction results, we found out there are some specific words with same patterns that are assigned by a wrong tag and made our prediction noisy. For example, if the words start with a capital letter, with high probability, they are a proper noun. Another case we found interesting was the word "a" when it appears in a Spanish sentence. We have this word both in English and Spanish but with a different tag, and the train data set consists more English version of this word. As a result, our model in most cases labels "a" with a wrong tag for Spanish sentences. To solve these cases we defined rules in our model and improved accuracy to 0.911% .

In the next step, we changed the tag definition in our system by using language tags in the dataset. This Version of the system reached to the best accuracy which is 0.9135%.

Finally, we run trigram model on the dataset. Although we expect this method improve the result, in our case, this technique did not help us to improve the system and accuracy reduced to 0.9127%. This may happen because of over-fitting to the data in the training dataset and having different patterns in the train and test dataset.

Table 2 will show results of all methods we described above.

Method	**Accuracy**
Baseline	0.873%
Handling Unknown Words	0.891%
Adding Rules to the Model	0.911%
Using Language Tag	**0.9135%**
Using Trigram	0.9127%

Table 2: POS tagging Accuracy for all methods-each method is applied to the previous version of the system

Figure 1 shows 11 most frequent words in development dataset which our system could not predict correctly.

All of these words have more than one possi-

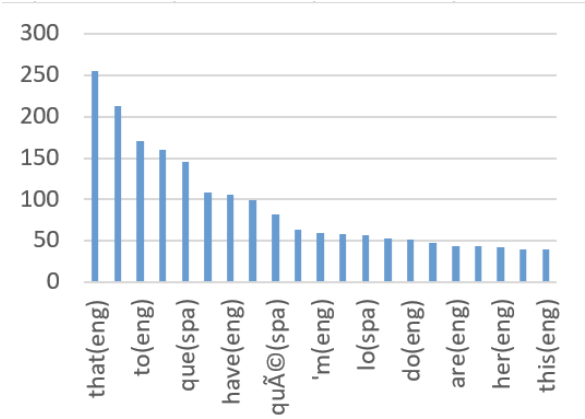

Figure 1: words with most frequent miss classified tags

ble tag. And, for some of them we are not able to predict correct tag only by looking at previous word's tag. For example, word "is" can be "auxiliary verb" and "verb", and we can not detect correct tag unless we investigate next word after "is".

The last trick we used to improve the accuracy is combining train and development datasets. We trained on this new dataset and then tested on the test dataset. The accuracy of this system is 0.9145%.

5 Conclusion

POS tagging systems try to automatically tag each word in a sentence. In this work, we used Hidden Markov Model (HMM) to tag the input words. We improved our system by adding some features like some rules to handle special cases and also employing language tag to discriminate between words in English and Spanish.

References

Nidhi Adhvaryu and Prem Balani. Survey: Part-of-speech tagging in nlp. *International Journal of Research in Advent Technology (E-ISSN: 2321-9637)*.