PART OF SPEECH TAGGING ASSIGNMENT REPORT

NLP LAB OF COMPUTER ENGINEERING

Ufuk Baran Karakaya
ufukbarankarakaya@gmail.com

April 12, 2019

1 Introduction

In this assignment, the project is based on Hidden Markov Model and, Viterbi Algorithm to predict Part of speech (PoS) tags. Pos tags are used for analysis of a content and, disambiguation. The aim of the project is prediction of tags and, gaining the higher accuracy for test set. To implement three tasks of the project, the dataset is divided two separate parts as train set and, test set. Train set is used for creation of Hidden Markov Model. Also, test set is used for evaluation of our model.

In this process, Viterbi Algorithm is used for prediction of tags for every sentence and the code calculates the accuracy of the project and apply some pre-processing steps such as detection of sentences, stemming. In the last step of the implementation includes smoothing methods for unseen data and calculation of right predictions. The steps of implementation is explained in detail below.

2 Algorithm

The algorithm is based on two main steps such as Hidden Markov Model and, Viterbi Algorithm. In the pre-steps, the code divides the dataset to two parts as trainset and, testset. Trainset is used to create a model. The model has three probabilities such as initial probability, transition probability and emission probability. All these probabilities are used to predict tags of the words.

Before the creation of Hidden Markov Model, the code creates word objects and holds tag of word. All words of trainset holds in dictionary. Some words may have more tags than one. Thus, word objects have a tags list. While the code creates dictionary all tags lists are converted to dictionaries with using counter. The code implements Hidden Markov Model. After that according to the model, viterbi algorithms are applied and predicts tags sequence which is the most possible.

2.1 Hidden Markov Model

Hidden Markov model has three components such as initial probability, transition probability and, emission probability.

- Initial probability means probability of a sentence begins with the tag (i.e Verb Phrase).
- Transition probability means probability of tag sequence. For example, probability of coming noun phrase after verb phrase.
- Emission probability means probability of word which is used for a tag. For example, "gelmek" is used for verb phrase.

The code parse the trainset according to / character and first token is name of word and second token is added to tags list of this word. Also, to calculate the initial probability, words are eliminated from sentence and, <s>, </s> tags are added to sentence. **genarate n_gram** function creates a bigram structure for initial and transition probabilities.

After the creation of Hidden Markov Model process, the code has emission_probabilities dictionary which holds probability of emissions as nested dictionary, transition_probabilities_bi which holds transition probabilities of all tags in the trainset and, corpus tags which holds the counts of tags in the trainset.

2.2 Viterbi Algorithm

Viterbi Algorithm is used for prediction according to data of trainset. Hidden Markov Model provides three probabilities for calculation of probabilities. The code applies viterbi algorithm according to these probabilities.

First of all, the code reorganizes testset and splits the words and tags. Tags of testset will be used for evaluation part. Viterbi algorithm traverses the sentence and calculate the initial probability, transition probability and, emission probability for all words.

Crucial part of this algorithm is unseen data of test set. Some words may not be in trainset. When the prediction process this words do not have any probability and, tag. In this case, the algorithm applies **add-one-smoothing**. Thus, all unseen data have a probability. To calculate all probabilities for unseen data. Viterbi algorithm considers all tags. Even if a seen word do not have multiple tags, the algorithm calculates all tags options.

If we consider the thirteen tags, we can say that the viterbi algorithm creates 13x13 probabilities between two words. Step by step this operation repeats itself. back_trace functions is called and most possible tag is chosen for next steps. This process works from end to begin of the sequence.

All probabilities are quite small numbers so calculation of probabilities are implemented logarithmically. create_key function is used for search tag sequences in transition probabilities and tags of corpus.

All steps of viterbi are updated according to maximum probabilitiy for a tag to prevent increasing of complexity.

3 Conclusion and Future Words

After predicition steps, all tags of testset are used for evaluation step. The code counts all correct predictions and, divide the total count. Consequently accuracy of model is roughly **77%**. All predictions are printed on output.txt.

To increase the accuracy, the code implements some pre-processing steps such as start and stop tags, lower case method to ignore the upper letters. However, turkish stemming libraries are in adequate for stemming processing. For example, the stemming operation must detect a lot of inflection suffix geliyorum,geldim,.. all forms should belongs to "gel" word. However, stemming cannot overcome this problem. Also it parses may words wrong. For instance, bir_an is converted to bir_a by stemming. If we modify stemming advantages to the model, we can have higher accuracy in this project.

4 References

- bbm 497 second assignment
- bbm 495 Viterbi Algorithm Course slides