BBM497: Introduction to Natural Language Processing Lab. **(Due: 20/04/2020)**

Assignment #2

Instructor: Necva Bölücü *Name:* MELTEM TOKGÖZ, *Netid:* 21527381

INTRODUCTION

We were given a data set containing tagged sentences. In this project, I first created an HMM model with the train dataset, which includes intial, transition and emission possibilities. Then, using the model, I applied the viterbi algorithm to estimate the tags of the sentences in the test dataset. I will explain the data structures in the Hmm model, the structure of the viterbi algorithm and my results below.

1. **Task 1: Build a Bi-gram Hidden Markov Model (HMM)**

   When creating HMM, tags are used as hidden states and words are used as observation states. First of all, I kept the frequencies in dictionaries. Then, by applying laplace smoothing for intial and transition, I created dictionaries with possibilities. Emission hold the possibilities in a new dictionary without smoothing. Because if there is a word that is not trained in the test data, I should have done smoothing again. So I ran smoothing for emission probabilty inside the viterbi algorithm. Example dictionaries are given below.

Figure 1: Hidden Markov Model

Initial probability : Total 9 tag
{'B-ORG': 0.174, 'O': 0.5807, 'B-MISC': 0.035, ...}

Transition probability
{'B-ORG': {'O': 0.591, 'I-ORG': 0.393, ... }, 'O': {'B-MISC': 0.0168, ...}}

Emission probability
{'B-ORG': {'eu': 0.591, 'european': 0.393, ... }, 'O': {'soccer': 0.0168, ...}}

2. **Task 2: Viterbi Algorithm**

   I made a structure that will hold 4 features (word, tag, probability, back-pointer) for each element of the matrix. I clearly explained this in the figure below. Then, I have done the following for each sentence in the test dataset.

- I created a matrix of (number of tags x word count of test sentence).
- I found the possibilities of each element in this matrix according to certain formulas. It is explained as to how he found the possibilities in the figure below. From the probability values taken from the previous column, I kept the Matrix-Item, which maximizes the value I calculated now, as a back pointer. r-tag and c-word are already the word and tag of that element of the matrix. So, I have completed a matrix.
- After that, I started doing back tracking. I found the tag that made the last column maximum, and went back to the back-pointer. With this process i found the predicted tags.

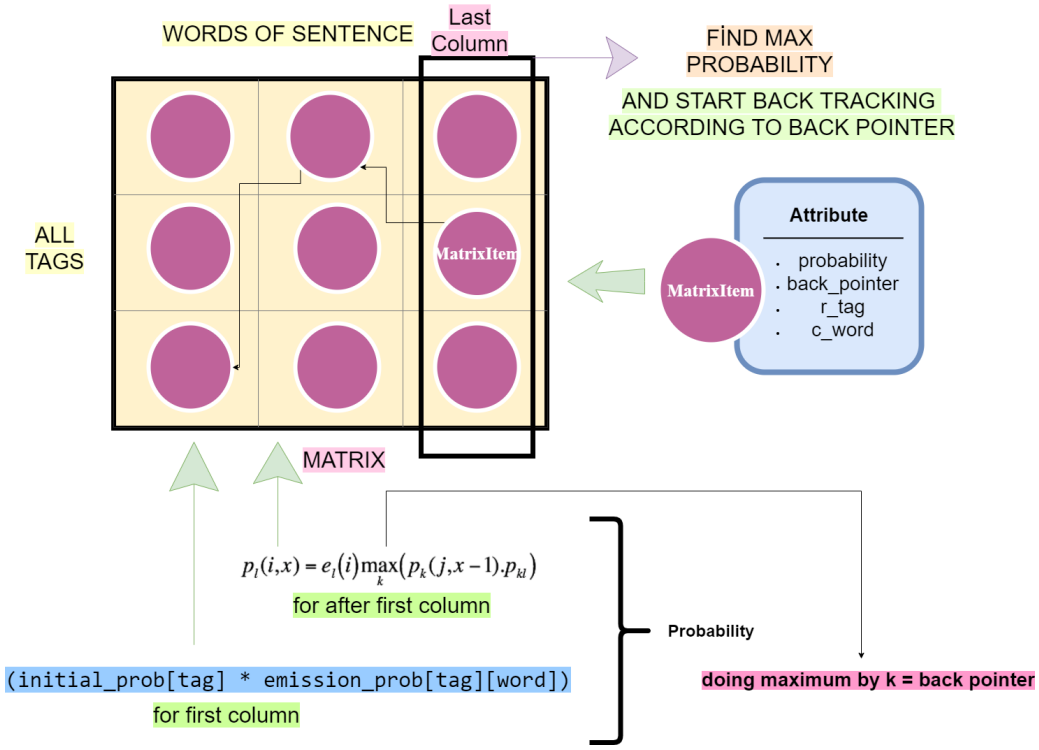

Figure 2: Viterbi Algorithm

- Smoothing
  I applied laplace smoothing for words not in the train dataset but in the test dataset or for words with probability 0.

3. **Task 3: Evaluation and Error analysis**

   After doing these operations for all test sentences, I calculated accuracy by comparing the predicted tags with real tags.

   For all test dataset :

   Acuracy = %80.08586

$$A(W) = \frac{\#of\_correct\_found\_tags}{\#of\_total\_words}$$

Figure 3: Accuracy

- **Analysis**
  When I find the accuracy, I analyzed the following:
  - I apply all of the tokens in lower case in both the train and the test data. I have observed that this slightly improves its accuracy.

- I used bi-gram for the possibility of transition when making the hmm model, it would be higher if I used tri-gram.
- The 'O' label is often guessed correctly because the 'O' label passes too often. The 89 percent 'O' label is estimated correctly.
- When I compared real tags and predict tags, I saw that some tags were mixed. For example, labels starting with the same mark are mixed. Such as, instead of I-PER tags, the I-ORG tag is usually guessed.
- 'B-MISC' and 'B-PER' tags are generally estimated with high accuracy.
- I have observed that I-PER AND I-ORG are generally predicted with low accuracy.
- As the test data grows, the accuracy decreases, albeit a little.