# Assignment 2

Navneet Agarwal
2018348

---

## Assumptions:

- ➔ Last word of some of the sentences did not include any tag, when I analyzed this, then I found due to apostrophe s one sentence was incorrectly splitted into two sentences, so I merged them.[9 such instances]
- ➔ Some of the outputs are printed on the .pynb while others are saved as a file. These files can be found in two folders : Output_Files_1_length and Output_Files_2_length
- ➔ The confusion matrix is saved as a .csv file for each fold, which can be found in the two output folders.
- ➔ The values of Precision, Accuracy and F1 score tag wise is stored in a text file named, “NoWordsForTag + {Fold Number} “. This can also be found in the two output folders.
- ➔ The average values of precision, recall and F1 score is also stored in a file as ”AvgPRF”
- ➔ Some of the statistics are printed out while the details such as Number of words for each tag and  words having more than one tags are saved in a text file named as “stat” in the two folders.
- ➔ Weighted average has been calculated for Precision, Recall and F1 score.
- ➔ I have implemented two Viterbi algorithms both for length 2 and length 3. Since my laptop crashed almost every time for the original viterbi , I optimized it on the assumption : I am considering only those tags for a word which are possible according to the corpus instead of taking all the 472 tags. If a word does not appear in my corpus, I have considered all the tags for that particular word.
- ➔ Lets say Navneet can only be tagged as [NN] and [JJ]
    - ◆ So while applying viterbi for this word i consider these two tags only
- ➔ Underflow has been handled using the logarithmic function on probability.
- ➔ Laplace Smoothing has been used to smoothen the zero probabilities.
- ➔ There are two python notebooks one with comments ,one without.
- ➔ COmments were added at last, so there might be some indentation issues[Though I have crosschecked it]

## Preprocessing:

- ➔ Split the whole corpus on a new line “\n” character so as to obtain a list of sentences and store it in a list.
- ➔ Now, this list contains all the given sentences in my corpus.
- ➔ Traverse this list of sentence:
    - ◆ For every sentence, split it on space and store it in a temporary list.
        - Traverse this temp list and further split earache token on “_” to get word and its particular tag.

    - ◆ Store these pairs for every sentence in a list.
    - ◆ The sentences will now be stored as a next list, where each element is [word,tag]
- ➔ Use the kfold method from sklearn to perform 3 fold cross validation.

The methodology is mentioned in the python notebook as comments.

## Outputs

**For length 1**

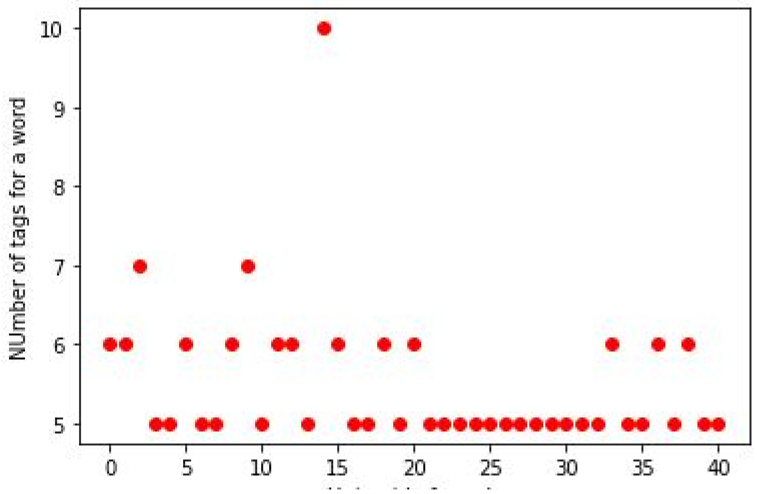
```
******Fold 1************

 Average Values

Precision: 0.9581 Recall: 0.9599 F1: 0.9559

 Statistics

NUmber of words in Training set: 32838
NUmber of tags in Training set: 369
Min,max and average length of sentences : 1 174 20
```



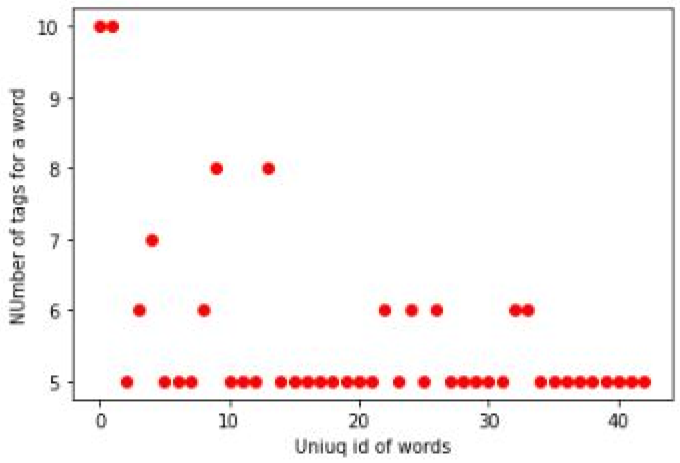
```
******Fold 2************

 Average Values

Precision: 0.9586 Recall: 0.9603 F1: 0.9563

 Statistics

NUmber of words in Training set: 32908
NUmber of tags in Training set: 372
Min,max and average length of sentences : 1 202 21
```



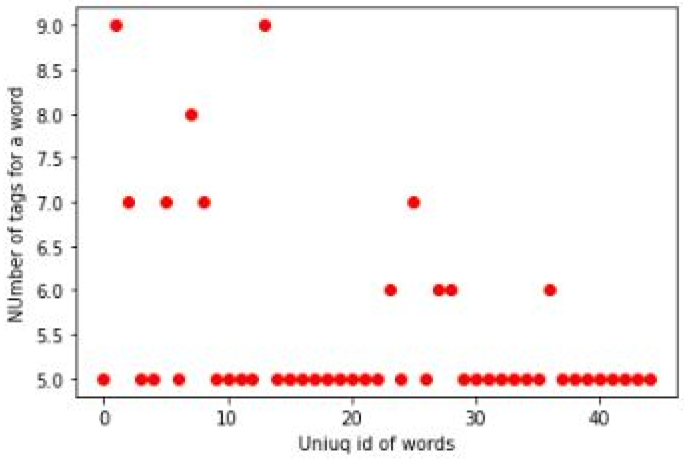
```
******Fold 3************

 Average Values

Precision: 0.9605 Recall: 0.9619 F1: 0.9585

 Statistics

NUmber of words in Training set: 32808
NUmber of tags in Training set: 363
Min,max and average length of sentences : 1 221 21
```



```
Average Scores for 3 folds:
Precision: 0.9591
Recall: 0.9607
F1: 0.9569
```

For length 2

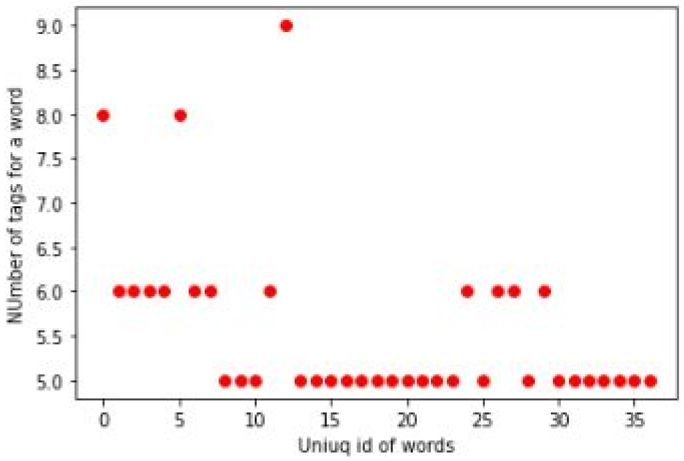
```
******Fold 1************

 Average Values

Precision: 0.8176 Recall: 0.5057 F1: 0.5819

 Statistics

NUmber of words in Training set: 33088
NUmber of tags in Training set: 360
Min,max and average length of sentences : 1 174 21
```



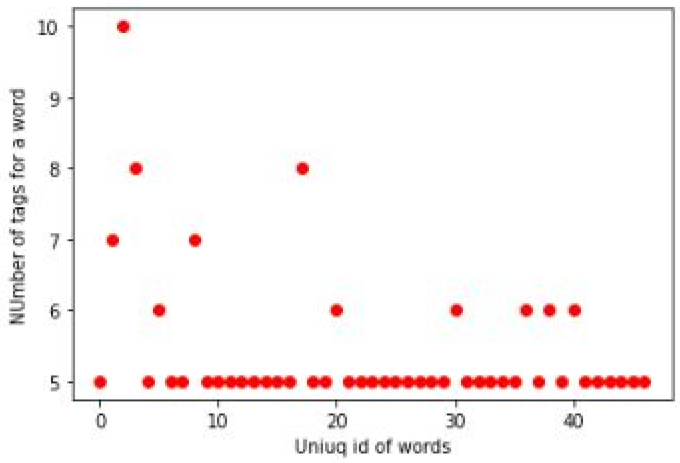
```
******Fold 2************

 Average Values

Precision: 0.818 Recall: 0.5058 F1: 0.5824

 Statistics

NUmber of words in Training set: 32658
NUmber of tags in Training set: 378
Min,max and average length of sentences : 1 221 21
```



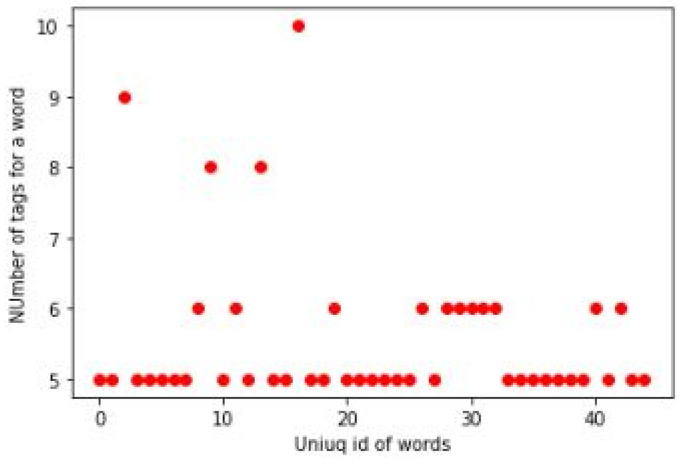
```
******Fold 3************

 Average Values

Precision: 0.8204 Recall: 0.5072 F1: 0.5839

 Statistics

NUmber of words in Training set: 32807
NUmber of tags in Training set: 353
Min,max and average length of sentences : 1 172 21
```



```
Average Scores for 3 folds:
Precision: 0.8187
Recall: 0.5062
F1: 0.5827333333333332
```

**2. Work out the mathematics of HMM for Markov assumption length 2, and include it in your assignment report.**

**->** Reference: Lecture Slides

In HMM

S*: Best possible state sequence

Goal: Maximize P (S|O) tag sequence by choosing best sequence S

So,

S* = argmaxS P(S | O)

P(S | O) = P({s1 , s2 , s3 , s4 , s5 , s6 , s7 .. sn} | {o1 , o2 , o3 , o4 , o5 , o6 , o7 ... on})

[Applying chain Rule]

P(S | O) = P(s1 | O) P(s2 | s1, O) P(s3 | s1 s2, O) P(s4 | s1 s2 s3, O) ... P($s_n$ | $s_1$ $s_2$ ... $s_{n-1}$ , O)

[Apply Markov assumption for length 2,
The current sate depends on the previous two states
$P(s_n \mid s_1\ s_2 \dots s_n) = P(s_n|s_{n-1},s_{n-2})$ ]

$P(S|O) = P(s1 \mid O)\ P(s2 \mid s1, O)\ P(s3 \mid s2 ,s1, O)\ P(s4 \mid s3,s2, O) \dots P(s_n \mid s_{n-1}s_{n-2} , O)$

[ Applying Baye's Theorem
$P(A|B) = P(B|A)*P(A)/P(B)$
where,
$P(A \mid B)$ = Posterior
$P(B \mid A)$ = Likelihood
$P(A)$ = Prior
$P(B)$ = Normalizing constant ]

$S^* = \text{argmax}_S\ P(S \mid O) = \text{argmax}_S\ P(O|S) . P(S)$

[ P(O) same for all sequences so ignored ]

**Prior**
$P(S) = P(s1) . P(s2|s1) . P(s3|s2,s1) \dots P(s_n|s_{n-1}s_{n-2})$

**Likelihood**
$P(O \mid S) = P(o1 \mid S)\ P(o2 \mid o1, S)\ P(o3 \mid o2, S)\ P(o4 \mid o3, S) \dots P(o_n \mid o_{n-1} , S)$

[We know that for the state of any observation depend upon current state only]

**Likelihood**
$P(O \mid S) = P(o1 \mid s1)\ P(o2 \mid s2)\ P(o3 \mid s3)\ P(o4 \mid s4) \dots P(o_n \mid s_{n-1})$

Hence,
$P(S \mid O) = P(O|S) . P(S)$

$= P(s1)P(s2|s1)P(s3|s2,s1) .. P(s_n|s_{n-1}s_{n-2}).P(o1 \mid s1)\ P(o2 \mid s2)\ P(o3 \mid s3)\ P(o4 \mid s4) \dots P(o_n \mid s_{n-1})$

**Adding two start and observations, $s_0,s_{01},o_0,o_1$**

$=P(s1|s_0s_{01})P(s2|s1s0)P(s3|s2,s1) .. P(s_n|s_{n-1}s_{n-2})P(s_{n+1}|s_ns_{n-1})P(s_{n+2}|s_{n+1}s_n).P(o_{01} \mid s_{01})P(o_0 \mid s_0)P(o1 \mid s1)\ P(o2 \mid s2)\ P(o3 \mid s3)\ P(o4 \mid s4) \dots P(o_n \mid s_{n-1})$

Where $P(o_{01} \mid s_{01})$ ,$P(o_0 \mid s_0)$ are both 1 so can be ignored [ initial emission probability]

$$P(S|O) = \pi_{i=1}^{k}\ P(o_i \mid s_i).P(s_i|s_{i-1}s_{i-2})$$

Where, $P(s_i|s_{i-1}s_{i-2}) = \text{count}(s_{i-1}s_{i-2}s_i)/\text{count}(s_{i-1}s_{i-2})$

$$S^* = \text{argmax}_S \{ \pi_{i=1}^{k} \ P(o_i \mid s_i).P(s_i|s_{i-1}s_{i-2}) \}$$

**Q3.**
The word types which have only a small number of occurrences in our corpus are wrongly tagged. It might also happen that the occurrence of particular word_tag in training corpus are very low and due to which the words can get wrongly tagged as HMM is based on probability, which is based on the number of occurrence.
SOme of the tags which had zero precision after 3-fold crossvalidation of model are:
**CD-NC**
**DO*-HL**
**WDT-HL**
**HVZ-TL**
**NR-TL-HL**
***-NC**
**NR+MD**
**JJ-TL-HL**
**AT-TL-HL**
**FW-RB-TL**
**FW-VBZ**

And the minimum precision of 0.2 was for tag **)-HL**