Assignment 2

Mthodology and Assumptions

- Used Add-k Smoothing
- Added Start and End dummy words in each sentence
- Randomised the dataset before splitting
- For any tag, its first two characters are considered and rest ignored. For example - Merger_NN-HL, then word **Merger** will be considered to be tagged as **NN** and not **NN-HL.**
- **Case folding** is used for word normalisation.
- Words without tag in dataset are tagged as OOV(Out of Vocabulary).
- **SpeedUp - 9x** via Multiprocessing + Using the pypy3 compiler instead of python3
- Time for Bigram - **1.5** min and for trigram - **30 min**
- Used **Hold-out** method for **trigram**.
- Command to run code - **python3 <code.py> <dataset_file_name>**
- Bigram model has **want_parallelism** variable to disable multiprocessing and it takes 3.5 min around for execution.
- Empty lines are removed from dataset

Q1 **Bigram Model**

Following results are based on **weighted-macro**

Fold 1

Accuracy = 0.8820700565019499 **Precision** = 0.9358948348936105

Recall = 0.9364771856118227 **F1- score** = 0.933143581595565

Fold 2

Accuracy = 0.8833151578620 18 **Precision** = 0.9365496910984699

Recall = 0.9369760927122053 **F1- score** = 0.9339699478245068

Fold 3

Accuracy = 0.8822880991486078 **Precision** = 0.935943581336039

Recall = 0.9364604342287322 **F1- score** = 0.9333650508698922

Average of 3-folds

Accuracy = 0.8825632238122533 **Precision** = 0.9361293691093732

Recall = 0.9366379041842533 **F1-score** = 0.9334928600966547

Accuracy = 0.936 when I used formula

Accuracy = correctly_tagged / (correct_tagged + wrong_tagged)

Q1 **Trigram Model**

Following results are based on **weighted-macro**

Accuracy = 0.8680384420167648 **Precision** = 0.9294264089360309

Recall = 0.9284344250811633 **F1- score** = 0.9244502903635906

Accuracy = **0.926135833131034** when I used formula

Accuracy = correctly_tagged / (correct_tagged + wrong_tagged)

Q-2) For markov assumption length = 2

$$P(t_1^n) = \prod_{i=1}^{n} P(t_i \mid t_{i-1} t_{i-2})$$

We need to such a tag sequence t_1^n for a given word sequence $w_1^n = < w_1, w_2 - \text{—} \; w_n >$ that maximises $P(t_1^n \mid w_1^n)$

Now $P(t_1^n \mid w_1^n) = \dfrac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$

since w_1^n is given and it will be same in all possible t_1^n so we ignore it.

$$P(t_1^n \mid w_1^n) = P(w_1^n \mid t_1^n) \, P(t_1^n)$$

$$= \prod_{i=1}^{n} (w_i \mid t_i) \cdot P(t_i \mid t_{i-1} t_{i-2})$$

likilihood assumption — for trigram assumption

$$= \prod_{i=1}^{n} \frac{\text{count}(w_i, t_i)}{\text{count}(t_i)} \times \frac{\text{count}(t_i t_{i-1} t_{i-2})}{\text{count}(t_i t_{i-1})}$$

For $P(t_1 \mid t_0 t_{-1})$ and $P(t_2 \mid t_1 t_0)$ to be valid, we insert two ~~tunny~~ dummy ~~words~~ w_{-1}, w_0 and w_{n+1}, w_{n+2} at back and front of word sequence w_1^n. and then calculate the required emission and trasmission matrix.

Q3

- Words which are uniformly distributed over the tag-set are tagged incorrectly because they have the probability are close to each other for any tag consider and hence they are the ones which are tagged incorrectly.
- Similarly if t_i and t_{i+1} have uniformly distribution then they are the ones which get wrong tags due to same reason as above.
- If the tags are not enough for a particular word in a dataset then also wrong tag is given due to less data available.