1. Splitting the data

The data was split into 80:20 ratio. 80% for training set and 20% for development set. No pre-processing was used on this dataset. The total samples in the dataset were 7816. The data was spliced into their respective ratios. The *train_data* contained 6252 samples (80%) whereas the *test_data* set contained 1564 samples (20%).

2. Error analysis: False positives

The sentences associated with false positives were printed by creating a loop which iterated over data from prediction array. The array contained sentences which were tagged by *CRFTagger.* The Conditional Random Field Tagger belongs to the *NLTK* (Natural Language Toolkit) class.

To find the five lowest false positive, a classification report was first created which was then stored into a *DataFrame* from *Pandas* package. The *DataFrame* allows the usage of a method called *sort_values* which can sort out data in an ascending order by *precision.*

The formula for precision is as follows:

- Precision = True Positives/ (True Positives + False Positives)
- Precision = TP/ (TP + FP)

The following is the table of the classes with the lowest precision:

Table 1 - Classes with least precision

Class	Precision
I-Opinion	0.071111
B-Plot	0.215287
B-Origin	0.365239
B-Opinion	0.432584
B-Soundtrack	0.437500

3. Error analysis: False negatives

The sentences with false negatives were printed in a similar way as false positives were printed as mentioned in section 2. However, this time we sort out the data in the *DataFrame* by using *recall*.

The formula for precision is as follows:

- Recall = True Positives/ (True Positives + False Negatives)
- Recall = TP/ (TP + FN)

The following is the table of the classes with the lowest recall:

Table 2 - Classes with least recall

Class	Precision
I-Soundtrack	0.069705
B-Soundtrack	0.070707
I-Character Name	0.090909
B-Quote	0.092199
I-Opinion	0.092843

4. Incorporating POS tags as features

The pre-processing method was modified to *updated_preprocess.* This method was used to attach *CRFTagger* to the dataset. This method concatenates the word and the POS using '@' operator. The dataset was pre-processed using this method and the data was again split in an 80:20 ratio. The *updated_train_data* containing 80% of the samples and *updated_test_data* containing 20% of the samples. The *get_features* method was modified to *updated_get_features* which could now separate the tagged word at operator '@' to correlate the word to corresponding feature. The features were then trained on the dataset to train the tagger. Then classification report was made and stored in a *DataFrame*. The *updated_df* was then compared to the normal *df* to see the improvements.

The F1-score formula is:

F1 = 2 * (precision* recall) / (precision + recall)

Table 3 - Macro averages comparison with and without POS

	No POS Tags	**POS Tags**
Precision	0.642763	0.794219
Recall	0.457124	0.638152
F1-score	0.493850	0.684324

A clear improvement can be seen in the table above because of POS tags.

5. Features experimentation and other optimization for optimal macro average

The pre-processing method was once again modified to *last_get_features* which now takes more features from the *prev_pos_tag_list* and *next_pos_tag_list* which contain previous and next tags. Both POS tag lists contain 4 tags each. The *last_get_features* methods also contain a tag call 'PRE_' for prefix. This takes the sum of added POS tags to 9.

Two hyper-parameters were also incorporated in the tagging process. The hyper-parameters are:

- Minimum Frequency
- L2 regularization – penalises complex models which is equal to the sum of the square of the coefficients. It reduces overfitting.

For the task at hand, the L2 regularization = 0.1 and the minimum frequency = 2. These values were chosen after repeated trials. If other values are chosen, the precision, recall and f1-score starts decreasing.

Table 4 - Macro averages comparison of normal features to Additional features and hyper-parameters

	Normal Features	**Additional Features and Hyper-parameters**
Precision	0.794219	0.803847
Recall	0.638152	0.723029
F1-score	0.684324	0.755658

There is very small improvement over the precision, but good improvement over the recall and f1-score. The scores have increased as the number of POS tags have increased and after because of the optimal valued hyper-parameters which have been fine-tuned after repeat trial.