ECS763P: Natural Language Processing
Assignment_1_CRF_tagging_in_Movie_Queries

Solution 1: A new training dataset has been created by extracting the first *80%* samples of *training_data* imported from the given file *'trivia10k13train.bio.txt'* so that our model is trained on enough samples and the *new_development_data* is created from the rest 20% of the *training_data* which needs to be used for testing purposes. List slicing is being used to split the original training data to avoid a random split. Subsequently, the classification matrix generated by testing the model on *new_development_data* illustrates that the precision and recall values have increased marginally for most of the classes whereas, for certain classes, it has vaguely decreased. However, the overall accuracy and the average f1-score have slightly increased.

Solution 2: To analyze the errors, the top five classes with the least precision values are extracted from the classification report, generated by testing the model on *new_development_data* by converting it into a python dictionary, sorting it on precision values and fetching the top five classes. As a result, 5 classes with the least precision values are *B-Soundtrack, I-Soundtrack, I-Opinion, B-Opinion, and B-Plot*.

To identify all the sentences having false positives for these classes, we are looping over all the sentences and examining if there is any word that is predicted as one of the above five classes and the true value is contrary to the predicted class. Also, recording the words that are contributing towards false positives into a dataframe *df_fp_word_frequency* for further evaluation. The result for each sentence is displayed in a table using *tabulate* function. In summary, *619 sentences* have been identified which contain false positives. On further analysis, we explored that the model falsely predicts 586 words into B-Plot class, 102 words in I-Opinion and 41 words in B-Opinion. These numbers have been verified against the confusion matrix heatmap as well to trust the calculation of false positives. These errors can be reduced by using the n-previous/next words in the features as the model needs to understand the context well for better performance.

Solution 3: Similar to the second task, the top five classes with the least recall values are extracted which are identified as follows: *B-Soundtrack, I-Soundtrack, I-Opinion, I-Character_Name, and B-Character_Name* To identify all the sentences with false negatives for these classes, each sentence is checked if there is any word which has true value among these classes and is predicted differently. On analyzing the results, *201 sentences* have been identified which contain false negatives. Additionally, we explored that the model, falsely predicts 184 words from B-Character Name, 145 words from I-Character Name, 76 words of I-Opinion, 41 words of I-Soundtrack, and 13 words of B-Soundtrack class. These errors can be reduced by using the previous and next words in the features with their POS tags as the model needs to understand the context well for better performance.

Solution 4: In the fourth problem, we are trying to leverage the POS tagging feature in CRF to facilitate the use of linguistic criteria. Current training data is a combination of word and its BIO tag. The raw *training_data* is processed through a *preProcessUpdated* function which uses a predefined model *crf_pos.tagger* and assigns a POS tag to each word and words are replaced by the combination of word and corresponding POS tag concatenated using a delimiter **'|',** for example, *'steve|PRPVBP'.* To extract features including POS tags, the get*_features_withpos* function has been updated to split the token using the predefined delimiter. The first part of the split is used as the feature word and the latter is used as the POS tag. Newly pre-processed training data is then split into 80% training data (*POS_new_training_data_80*) and 20% test data(*POS_new_development_data_20)* for training and testing the model. The resulting classification matrix is used to analyse the performance of the updated model. Accuracy factors precision, recall, and f1 score depict that for some of the classes' accuracy have slightly increased and for others, it has slightly decreased. There is no significant improvement in the performance of the model on the given data.

Solution 5: The effect of using POS tags alone on the training and development data is minimal, so the *get_features_engineered* function is updated with additional features and the model is trained and tested on these features to check the effect of each factor. The process followed is described as below:
Features addition 1: Added suffixes and prefixes of length four characters; Training and testing the model in this scenario increased the overall f1-score from *0.5584 to 0.5947.*
Features addition 2: Introducing previous two words, a previous word, a next word, and the next two words along with their POS tags helped in improving the accuracy further for the model. The average f1-score achieved for the model with the above features is *0.6244*.
Features addition 3: Identifying Numeric words, stop words, the start of a sentence and the end of a sentence. First, to add a new feature *NUM_WORD*, a list of numeric words like "zero", "one", "million", etc. is created and each token is compared against this list to identify the numeric words. Another feature for checking the stop words is added. Stop words are commonly used words like 'the', 'a', 'an', 'are', etc. Using *spaCy* library, the language model "*en_core_web_sm*" is imported to download the list of commonly recorded stop words. In the feature extraction function, all the tokens from our sentences are compared to the list of stop words and recorded as '*STOP_WORD*' in the feature list. Two more features '<START>' and '<END>' are added to identify if the token is at the start or end of the sentence. Subsequently, the model is trained and tested with the addition of the above features, resulting in a slight increment in the average f1-score to *0.6332*. Although the increase in the average score is marginal, the precision of the B-Soundtrack and I-Soundtrack increased dramatically.
Features Addition 4: Spacy library also provides Named Entity Recognition for entities like a person, org, etc. An idea to implement NER is to identify words that are entitled and can be better classified. There is no significant improvement observed by using NER and thus removed from the pre-processing to avoid complexity.
Hyperparameter Optimization: After using the applicable features for the training data, the main motivation of tuning these hyper-parameters is to minimize a predefined loss function for better results. The first step in the process is to decide the parameters to be tuned, here our defined parameters space consists of three parameters namely, the minimum frequency of features (*feature.minfreq*), Coefficient of L1 regularization (*C1*), and Coefficient of L2 regularization (*C2*). Model is trained with different values for these parameters to find an equivalent or better solution. Tuning the hyperparameters is an exhaustive approach as there can be infinite possibilities of the combination of these values. The approach used is to create an initial set of values for these parameters and test the model's average f1-score. The first set of these parameters used is the {"feature.minfreq":3, "c1":0.1 , "c2":0.25}, resulting f-score value observed in this case is *0.6488*, on tuning with the values as {feature.minfreq":2, "c1":0.1, "c2":0.5}, f-score observed in the similar range i.e. *0.6444*. An interesting result was observed on using the below values for these parameters {feature.minfreq":1, "c1":0.1, "c2":1}, f1-score shoot up to *0.85* whereas the performance of this model on full testing data was observed as *0.63* only. Thus, rejected these parameters as this can be a case of overfitting on training data. Following this exhaustive approach, *skopt* library of scikit-learn package is used to tune the parameters for a maximum value of f1-score. The parameters resulting in the max f1-score values are {feature.minfreq":5, "c1":0.93, "c2": 0.01} which gives the f1-score of *0.6797* on 20% of the training data.
Finally, the *CRFTagger* is trained on the complete training data and complete test data from the file '*trivia10k13test.bio.txt*" and the resulting f1-score achieved for the given data is *0.6513*.
In conclusion, the addition of relevant features in the model increases the accuracy of the classification but needs a manual definition of these features and regularization parameters.