NLP Assignment 1 Report

This report provides detailed explanation on how each question in the given assignment is resolved including the methods used.

Q1: Input and Basic pre-processing

To achieve this, the methods parse_data_line and the pre-process are used and updated as required.
parse_data_line: This method uses a predefined method convert_label to convert all different labels to only a binary value of REAL and FAKE. Once achieved, only first and second columns from the file i.e., label and statement are extracted using list indexes.
pre_process: A sklearn module of CountVectorizer is used to pre-process the data. The count vectorizer, extracts the features from the given text. This module by default does some pre-processing, tokenizing and convert the words into lower case.

Q2: Basic Feature Extraction

To achieve feature extraction a 'to_feature_vector' method is updated. This method gets a pre-processed list of tokens as an input. For each given input token, the occurrences of words are counted and maintained in a dictionary (feature_vector_dict). Using a for loop on list of tokens provided, each word is added as the key in dictionary and its value is the frequency of occurrence of that word (Initially the value will be set to 1). If the word is already present as a key in dictionary, then simply increasing its value by 1.
Also, a list (global_feature_list) is being maintained throughout a dataset to understand all unique features exists in dataset and to obtain a count of unique features. This list is initialized globally, and using if condition, for each feature from token this list is updated if that feature doesn't exist in list.

Q3: Cross-validation

This method performs 10 folds cross validation on Training dataset (80% of total dataset) by diving the training dataset into train and test dataset. This is achieved using a 2 for loops.
First for loop: Runs over a total training dataset with step size as fold size given. In this the test dataset will always has a same range i.e., from i-value to fold_size. For train dataset, when i-value is zero, it will start from i+fold_size till end and for all other folds need to concatenate the two parts to form a train dataset. So, this for loop will execute 10 times and for each fold train and test datasets will change according i-value. SVM Classifier will be then trained on this Training dataset.
Second for loop: This will run inside first for loop, once train and test data are formed. This for loop will run on each sample of test data. The 'predict_labels' method will be used to perform the prediction on each test sample. This result of predicted label and actual known label from test dataset are stored in a list. At the end of first for loop i.e., after each fold, precision, recall, f1-score and accuracy are calaulated on actual and predicted labels. These values and classification reports are also saved in a file for each fold. At the end of this method, the average of precision, recall, f1-score and accuracy for all folds is calculated.

Q4: Error Analysis

After training the classier for trained data, we have received the values for first fold's predicted labels. By using these predicted labels and actual labels a confusion matrix is plotted. According to this confusion matrix, below values for False positives and False negatives for first fold are obtained. We have obtained a large number for the values for FP=175 and FN=185. Due to this we received 56% of accuracy. The statements for which predicted and actual labels are different is stored into a file: False_Positives_And_False_Negatives_For_First_Fold.txt. This helped in performing the error Analysis and understanding the incorrect FP and FN values.

Q5: Optimising pre-processing and feature extraction

As the accuracy of the trained model is not that significant, some more techniques for pre-processing and feature extraction are used in this. In the basic model, count vectorizer and Unigram were used. More different methods than those are used and the difference in model accuracy can be observed. Below table shows the details:

Combinations of method used	Observations	Confusion Matrix
TFID Vectorizer with Unigram	Accuracy: 0.55, Percision: 0.55 Recall: 0.55, F1-Score:0.55	TP: 156, TN: 293 FP: 188, FN: 183
Normalizing and lemmatizing with Unigram	Accuracy: 0.58, Percision: 0.49 Recall: 0.58, F1-Score:0.45	TP: 8, TN: 467 FP: 14, FN: 331
Regex with Unigram	Accuracy: 0.58, Percision: 0.58 Recall: 0.58, F1-Score:0.58	TP: 177, TN: 299 FP: 182, FN: 162
Regex and Bigram	Accuracy: 0.60, Percision: 0.60 Recall: 0.60, F1-Score:0.60	TP: 169, TN: 324 FP: 157, FN: 170
Regex and Trigram	Accuracy: 0.58, Percision: 0.57 Recall: 0.58, F1-Score:0.57	TP: 120, TN: 359 FP: 122, FN: 219

From the above observations, Regex with Bigram approach provides better accuracy.

Q6: Using other metadata in the file

Until now, only Label and statement columns were used as features for training the classifier. The other columns in given input file, also hold the important information.
Columns used: Statement, Subject, Speaker, State_info, Party_affiliation, Last Col(col:13)
Condition used: remove statements if context=0 and Party_affiliation is None and Last column is empty.

Combinations of method used	Observations	Confusion Matrix
Regex and Trigram	Accuracy: 0.61, Percision: 0.59 Recall: 0.61, F1-Score:0.59	TP: 133, TN: 364 FP: 117, FN: 206
Unigram and Regex	Accuracy: 0.60, Percision: 0.60 Recall: 0.60, F1-Score:0.60	TP: 184, TN: 304 FP: 177, FN: 155
TFID, Regex with Bigram	Accuracy: 0.62, Percision: 0.61 Recall: 0.62, F1-Score:0.60	TP: 123, TN: 387 FP: 94, FN: 216
counter_vectorizer and bigram	Accuracy: 0.59, Percision: 0.58 Recall: 0.59, F1-Score:0.58	TP: 146, TN: 338 FP: 143, FN: 193
Lammatizing and Unigram	Accuracy: 0.41, Percision: 0.46 Recall: 0.41, F1-Score:0.24	TP: 338, TN: 1 FP: 480, FN: 1

Columns used: Statement, Subject, Speaker, Party_affiliation, Last column(col:13)
Condition used: Remove statements if Party_affiliation is None and Last column is empty.

Combinations of method used	Observations	Confusion Matrix
Bigram and Regex	Accuracy: 0.63, Percision: 0.62 Recall: 0.63, F1-Score:0.62	TP: 169, TN: 344 FP: 137, FN: 170
Unigram and Regex	Accuracy: 0.58, Percision: 0.58 Recall: 0.59, F1-Score:0.59	TP: 181, TN: 298 FP: 183, FN: 158
Trigram and Regex	Accuracy: 0.59, Percision: 0.58 Recall: 0.59, F1-Score:0.58	TP: 130, TN: 357 FP: 124, FN: 209

After verifying all observations, with different columns and conditions, Regex and Bigram combination with above mentioned columns and conditions, provides overall better observations. So, this is the final model will used to perform predictions on remaining Test Data (20% of Dataset).