Big Data : UE19CS322
Project
Team_ID: BD_279_284_378

1.1 Project title chosen: Sentiment
1.2 Design details:

Most of the units are tackled in a single file and a single function. This project can be explained in the form of stages:
Stage 1:Starting with the streaming, a basic approach was inherited where SparkContext was used to stream the batches with endless=False (not stream in loops).
Stage 2:Preprocessing on text data requires some form of transformations that vectorise the text to numeric data for efficient classification. A function for preprocessing was operated on each RDD. Stage 3: The text is first cleaned and then preprocessed using SparkFeatureExtraction which provides a multitude of functions that transform the data frame.
Stage 4:Transformations using the feature extraction functions. Although pipeline could be easily used, our design applied the transform function after each stage without the pipeline function.
Stage 5: Once the last stage of transformation is performed, we have the features column changed into a vector of numeric data. Perfect!
Stage 6: Here, the various models are initialised. For the purpose of this project, SGDClassifier, Multinomial Naive Bayes and PassiveAggressiveClassifier are the chosen models among the suitable ones. Since this project asks us to resort to incremental learning, only a limited amount of models support incremental learning in sklearn.
Stage 7: 3 chosen classifiers are partially fitted using the features column and tested on the split batch of data frame. The partial fit function requires a 2D data, but the manual transformations applied resulted a Scalar array, hence this called for some reshaping.

Stage 8: The models, once fitted, are saved using the pickle module that is built-in with Python. One each batch that is trained and tested, the saved model is dumped and loaded respectively. The metrics focussed here are accuracy, F1 score and precision. The results of various batch sizes are written into a text file that writes the difference metric scores for 3 classifiers
Stage 9: Clustering was done using MiniBatchKMeans
Stage 10: Plotting using matplotlib

1.3 Surface level implementation details about each unit

Streaming Data: It was realised that the object obtained was a DStream, which needed transformation in order to further work on it. This DStream is transformed into an RDD and further into a Dataframe.
Processing Stream: Once done, the text columns is tokenized, stop words removed, hashingTF applied and finally, Tfidf Vectorizer is applied which changes the text feature to numeric data based on the occurrence, frequency and thus, important features are created.
Building Model: Now, the two columns of importance are the final features in vector form and the sentiment label. To partially fit any model, the data column must be reshaped first since the data is a scalar array than a 2D array.
Learning from data: This project implemented SGDClassifier with loss='log' which implies logistic regression. The next model was Naive Bayes Multinomial model which is useful when features are discrete than a Binomial one. The third was PassiveAggressiveClassifier, which is not a commonly used model but uses the approach of weights given based on the classification performance.
After obtaining the metrics it was observed that hyperparameter tuning was a potential improvement to the scores obtained. For example, the number of features in SGDClassifier gave different results. Setting random shuffle to False gave better results than when set to True.

Testing model: Coming to the plots, the 3 metric scores for 3 classifiers was written to a log.txt file which was later obtained to compute arrays for different metrics for each classifier. Later plots were obtained to visualise the performance.

It was noted that the performance of PAC declined when batch size increased. It was also observed that SGDClassifier performed better when stop words were NOT removed.

Clustering: Coming to MiniBatchKmeans, the clusters were partially fit and the centroids were used to compute the pairwise distance to obtained the predicted labels.

1.4 Reason behind design decisions

Since this a text based project, it is important to use vectorising preprocessing functions to improve the efficiency. The classifiers were trained together on each batch size in the same function. The project is simple but new. Although the models could further be improved through Lemmetizer and Stemming, the focus was more on the incremental learning.

1.5 Takeaway from the project

Incremental learning can substantially improve the way data is processed and trained, makes machine learning all the more fun!