CSE587 Assignment 3

TeamName on Kaggle: *RDS*

Rishabh Joshi (rjoshi4@buffalo.edu, 50316839)

Shivaprakash Hiremath (hiremath@buffalo.edu, 50313811)

Deepanshu Yadav (dyadav@buffalo.edu, 50321285)

Notes to Grader:

- Java version used in Virtual Machine: Java 10.0.2 (Default which was given along with VM) - No changes made.
- Logistic Regression model was used with the same parameters for all parts of the assignment.
- Test data was never fit into any model. All models were trained only on train data.
- Kaggle scores to consider (Comments) :
  - RDS_Task3_Consider - 0.98791
  - RDS_Task2_Consider - 0.96980
  - RDS_Task1_Consider - 0.95689
- For running the tasks, copy the assignment3.zip into VM. Untar the zip file. Start the jupyter and run all three notebooks one after the other.
- Folder structure in submission zip file:
  - task1- task1 jupyter notebook, test and train files and saved models
  - task2- task2 jupyter notebook, test and train files and saved models
  - task3- task3 jupyter notebook, test and train files and saved models
  - results- result_task1, result_task2, result_task3 (csv files to consider)
  - ***Other_experimented_models- word2vec
- Video Link: **https://buffalo.box.com/s/df1q45ibgir4zvxfwmp23zlaqh9vfqr2**

Task 1: Word Frequency

Code Explanation:

1) Train and test files are read using pandas and converted to spark dataframes.
2) Genres are converted to string format by removing '[', ']' and ' ' ' and then converted to lower case to maintain consistency.
3) Multi label genres are split into 20 columns with binary representation of 0 or 1. If present, it is represented as 1 and not as 0.
4) The plot column of train and test files is tokenized using Tokenizer and stop words are removed using StopWordsRemover and output is saved to the 'words_clean' column.
5) Now the model is trained by calculating frequency of each word in the 'words_clean' column using CountVectorizer and output is saved in the 'features' column of the train and test dataframes.
6) Train files are created for each genre so overall 20 files are created which will be used to train 20 logistic regression models corresponding to each genre and predictions will be saved in the predictions column.
7) The columns are renamed to prediction1, prediction2 and so on corresponding to each genre and columns are appended one after the other.
8) The predictions which are in integer format are converted to string format and all predictions are combined to convert the dataframe in the format which can be read by Kaggle.
9) The dataframe is then saved as a csv file.

F-1 score from Kaggle: 0.95689

Task 2: TF-IDF

Code Explanation:

1) Train and test files are read using pandas and converted to spark dataframes.
2) Genres are converted to string format by removing '[', ']' and ' ' ' and then converted to lower case to maintain consistency.
3) Multi label genres are split into 20 columns with binary representation of 0 or 1. If present, it is represented as 1 and not as 0.
4) The plot column of train and test files is tokenized using RegexTokenizer and stop words are removed using StopWordsRemover and output is saved to the 'words_clean' column.
5) Now the model is trained by calculating frequency of each word in the 'words_clean' column using CountVectorizer and output is saved in 'rawFeatures' column of the train and test files.
6) The 'rawFeatures' column is now used as an input for the IDF model.
7) The IDF model is trained on the output of CountVectorizer model and the transformed files are saved as rescaledData and test_rescaledData.
8) Train files are created for each genre so overall 20 files are created which will be used to train 20 logistic regression models corresponding to each genre and predictions will be saved in the predictions column.
9) The columns are renamed to prediction1, prediction2 and so on corresponding to each genre and columns are appended one after the other.
10) The predictions which are in integer format are converted to string format and all predictions are combined to convert the dataframe in the format which can be read by Kaggle.
11) The dataframe is then saved as a csv file.

F-1 score from Kaggle: 0.96980

Task 3 & 4: Customized feature extraction

Approach :

Our customized approach involves use of Ngrams (we are currently using bigrams) on word tokens generated from movie plots and then using their frequencies as our features. We used this approach because it was very intuitive as two words together if come in a movie plot and then again appear in the plot of another movie then the probability of that movie belonging to the same genre will be higher.

Eg: if “intense action” appears in the plot of a movie belonging to the ‘action’ genre, then if this phrase appears in some other plot, the probability of that movie belonging to ‘action’ genre becomes higher.

Code Explanation:

1) Train and test files are read using pandas and converted to spark dataframes.
2) Genres are converted to string format by removing ‘[‘, ‘]’ and ‘ ‘ ‘ and then converted to lower case to maintain consistency.
3) Multi label genres are split into 20 columns with binary representation of 0 or 1. If present, it is represented as 1 and not as 0.
4) The plot column of train and test files is tokenized using Tokenizer and stop words are removed using StopWordsRemover and output is saved to the ‘words_clean’ column.
5) Now we calculate N-Grams from ‘words_clean’. We used n-value as 2 ie we are calculating bigrams for words belonging to a plot.
6) We will pass the ngrams of every plot to the count vectorizer vocab and calculate their frequencies.
7) We will use these frequencies as features for the logistic regression models
8) Train files are created for each genre so overall 20 files are created which will be used to train 20 logistic regression models corresponding to each genre and predictions will be saved in the predictions column.
9) The columns are renamed to prediction1, prediction2 and so on corresponding to each genre and columns are appended one after the other.
10) The predictions which are in integer format are converted to string format and all predictions are combined to convert the dataframe in the format which can be read by Kaggle.
11) The dataframe is then downloaded as a csv file.

F-1 score from Kaggle: 0.98791