Project 3: Report
CSE 587: Data Intensive Computing

CHARANTEJA KALVA, 50336873
SAI BHARAT KONAKALLA, 50336876
SOUMITH REDDY CHINTHALAPALLY 50336650

Libraries Used

- pyspark.sql.functions
- pyspark.ml.feature
- pyspark.sql.types
- pandas
- pyspark.ml.classification

Some of the functions we have used from the above libraries are:

- RandomForestClassifier
- Tokenizer
- StopWordsRemover for Removal of stop words
- StringIndexer
- IndexToString
- Word2Vec
- HashingTF for TFIDF
- CountVectorizer

Steps involved in the execution of Models :

Some common step used across all models :

- Loaded the train and test data in pyspark dataframe

- Cleaned the plot column using regular expression to remove special characters.
- Divide the plot description into tokens using Tokenizer method
- Removed the stop words from the plot to make the model independent of irrelevant words.
- Converted the genre into Integer using StringIndexer which will make each String an unique integer and by doing so the classification will become multi-class probem We are using RandomForestClassifier algorithm in all our models.

Model using CountVectorizer :

- The dataframe obtained after applying the above steps on data is used for solving the multi-class problem
- We have applied a method CountVectorizer which converts word features into vectors. Also here we are limiting the number of word features so we are taking the most frequent word features among all the word features.
- Now we sent the word features obtained using the CountVectorizer as input(features) and the classes(Integer) to train our model.
- Converted the genre into Integer using String Indexer which will make each String an unique integer and by doing so the classification will become multi-class problem.
- We are using RandomForestClassifier ML model to predict the genres.
- The output predictions we get after applying the model we will convert it back to String labels. We have created a UDF to convert those string labels to the desired format required.
- **Kaggle F1 Score : 1.00000**

Model using HashingTF (TF-IDF) :

- The dataframe obtained after applying the above mentioned common steps on data is used for solving the multi-class problem.
- We have applied methods HashingTF (TF-IDF) which converts word features into vectors. Also here we are limiting the number of word features so we are taking the best word features(word features with high weightage) among all the word features.
- Now we sent the word features obtained using the HashingTF as input and the classes(Integer) obtained while converting genre to integer as output to train our model.
- The output predictions we get after applying the model we will convert it back to String labels. We have

created a UDF to convert those string labels to the desired format required.

- **Kaggle F1 Score : 1.00000**

Model using Word2Vec :

- The dataframe obtained after applying the above mentioned common steps on data is used for solving the multi-class problem.
- We have applied Word2Vec method which creates an average word vector for each feature. We are setting VectorSize and minCount whilch helps us to maintain vector size and the minimum number of times a token must appear to be included in the word2vec model's vocabulary.
- Now we sent the word vectors obtained using the Word2Vec as input and the classes(Integer) to train our model
- We are saving this model and loading the saved model in a different file.
- We are predicting the values using the saved model we will convert it back to String labels. We have created a UDF to convert those string labels to the desired format required
- **Kaggle F1 Score : 1.00000**

Kaggle Team Member Names:

- Soumith Reddy Chinthalapally
- charanteja1997
- Sai Bharat Konakalla

Link to the video explanation of the assignment:

https:// youtu.be/ h8ArAWPx6LA