Movie genre prediction using apache spark

Sankeerth Tella
50317364
Stella3

Sai Charith Reddy Pasula
50320452
saichari

Prudhveer Reddy Kankar
5032012
prudhvee

Introduction:

In this project, we perform text classification on movie dataset to predict the genre of a movie given the plot of the movie.In the dataset, we are given 31108 movieid,movie names,plot and genres that correspond to the movie.
Each movie can have multiple genres associated with it. So this project is an example of a multi-label classification project.

Preprocessing:

As part of preprocessing, we used pandas to create a new csv file called genres.csv. In this file, we took 20 columns each corresponding to a single genre and 1 and 0 for if a movie has that particular genre or not. We changed the java version in our virtual box to '1.8.0'

Task 1- Using Term Document matrix to perform classification:

In this task, we build a term document matrix for the plots of all the movies. From the document matrix, we use logistic regression to classify the data.

Processing:

We use the regexTokenizer(function present in spark.ml library) which splits the sentences into tokens/words.This acts upon the plot column of the dataset and outputs a set of words for each row.
Next we remove the stopwords by using the stopwordsRemover function given in spark.ml library. To do this, we manually created a list of 150 frequent words and fed that to the function. Some of those stopwords are: "i", "me", "my", "myself", "we", "our" etc. This acts upon the words column and outputs the filtered column.
Next we used the hashingTF function present in the spark.ml library to build the term document matrix. This acts upon the filtered column and outputs the features column.
We included all the above three functions into a pipeline object from the spark ml library and fit and transform our training and test data frame using the pipeline.

Training:

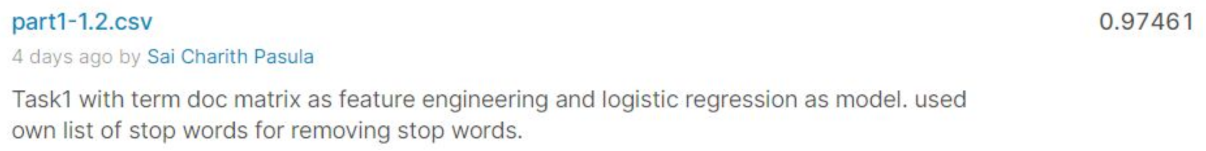
We used two models to train the above obtained features to classify the genres.

Logistic Regression:

The logistic Regression model present in spark directly takes the features and label and fits the model. We then use transform for getting predictions on the test data. Finally we append the movie id of the test case and the prediction for the movie in a dataframe.
We use a for loop to iterate through all the labels to train 20 different models and keep adding the predictions to the dataframe.

Finally after training all the models, we concatenate all the columns of the dataframe convert it into a pandas dataframe to put the movie id and predictions into a csv file

Kaggle competition F1 score:



part1-1.2.csv 0.97461
4 days ago by Sai Charith Pasula
Task1 with term doc matrix as feature engineering and logistic regression as model. used own list of stop words for removing stop words.

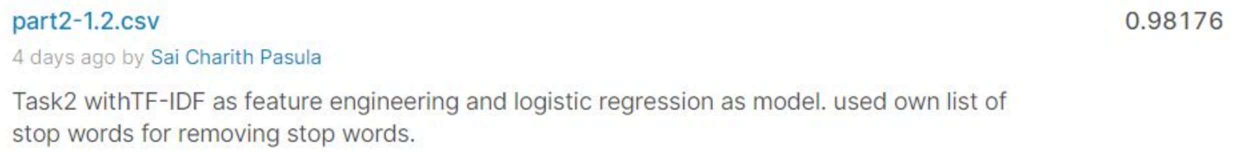
Task2- Using TF-IDF to perform classification:

In this task, we build a TF-IDF matrix for the plots of all the movies. From the TF_IDF matrix, we use logistic regression to classify the data.

Preprocessing:

We used a similar method as above and built a hashing TF raw features. These raw features are fed into the IDF function present spark.ml library that calculates the inverse document matrix for all the rows of plot data. We finally obtain TF-IDF values for the plot. Using this we train the model in a similar way as above.

Kaggle competition F1 score:



part2-1.2.csv 0.98176
4 days ago by Sai Charith Pasula
Task2 withTF-IDF as feature engineering and logistic regression as model. used own list of stop words for removing stop words.

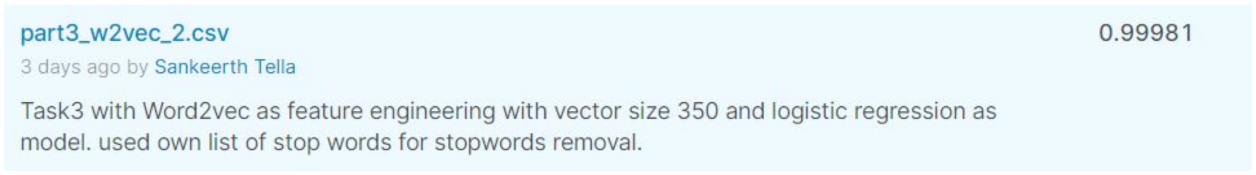
Task3 - Using advanced feature engineering to perform classification:

In this task, we calculate vectors for the plots of all the movies. From the vectors, we use logistic regression to classify the data.

Preprocessing:

We use regextokenizer to obtain the tokens and Stopwordsremover to remove the stopwords. Later we use the word2vec function present in spark.ml to calculate the vector representation of the words. THe main advantage of the vector representation is that similar words are close together in the vector, which makes generalization to novel patterns easier. From these vectors, we train the logistic regression model in a similar way as mentioned above.

Kaggle competition F score:



part3_w2vec_2.csv 0.99981

3 days ago by Sankeerth Tella

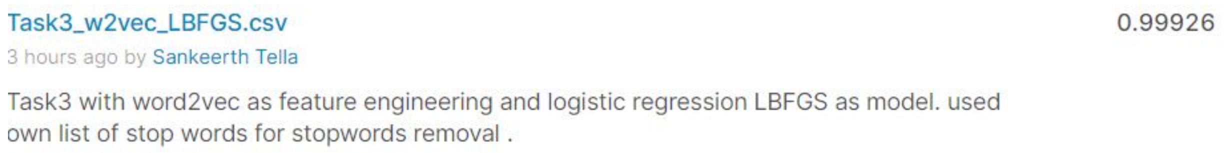
Task3 with Word2vec as feature engineering with vector size 350 and logistic regression as model. used own list of stop words for stopwords removal.

Additional trails:

LogisticRegressionWithLBFGS:

The LogisticregressionWithLBFGS model expects an rdd labelled point as input. TO create a labelled point, we used the rdd.map function that takes in the input of each row and outputs a labelled point<labels,features> . We use the train function with the above created labelled point as input. After the model is trained, we use the predict function on the test labelled point.
The predictions are then converted to a struct schema and concatenated into the final df.
After training all the 20 models, we concatenate all the columns of the testdf, convert it into a pd dataframe and put them in a csv file.

Kaggle competition F1 score for task1:



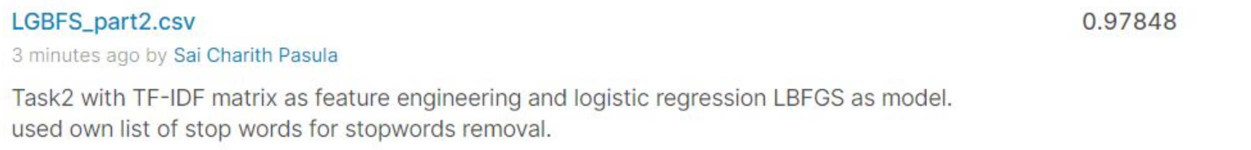
Task3_w2vec_LBFGS.csv 0.99926

3 hours ago by Sankeerth Tella

Task3 with word2vec as feature engineering and logistic regression LBFGS as model. used own list of stop words for stopwords removal .

The filename was wrong for the above csv file.

Kaggle competition F1 score for task2:

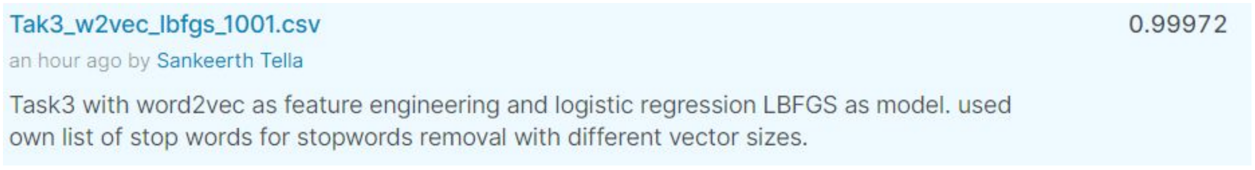


LGBFS_part2.csv 0.97848

3 minutes ago by Sai Charith Pasula

Task2 with TF-IDF matrix as feature engineering and logistic regression LBFGS as model.
used own list of stop words for stopwords removal.

Kaggle competition F1 score for task3:



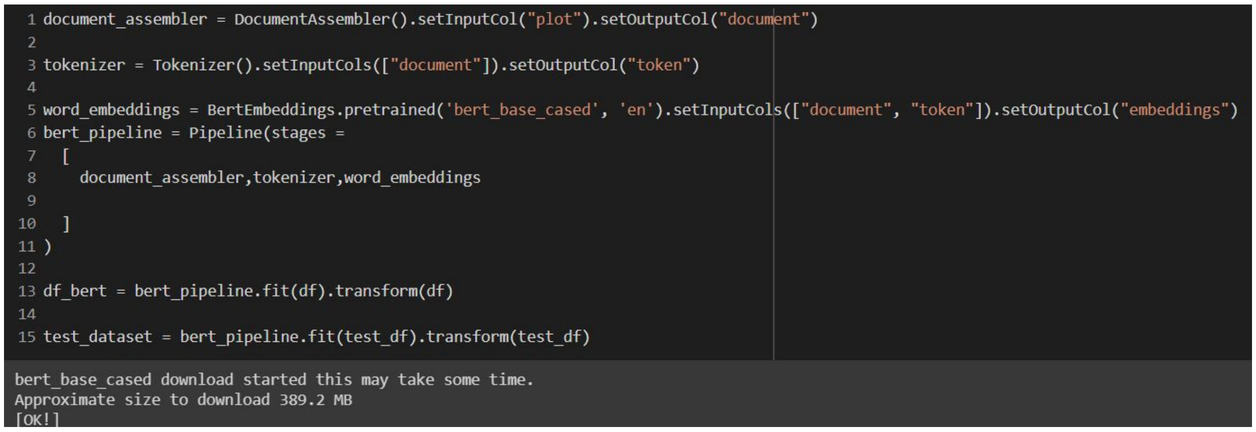
Tak3_w2vec_lbfgs_1001.csv 0.99972

an hour ago by Sankeerth Tella

Task3 with word2vec as feature engineering and logistic regression LBFGS as model. used
own list of stop words for stopwords removal with different vector sizes.

Bert word embeddings:

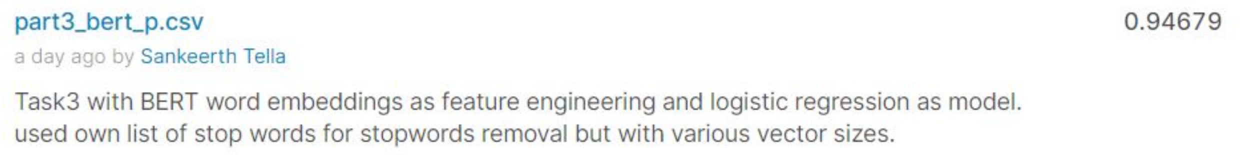
We used the bert embedding function present in the sparkNLP library. For this we first installed the sparkNLP library version -2.5.0.
Bert embeddings has a pre-trained pipeline that transforms the words into bert embeddings.
Using these embeddings, we train the logistic regression model but it didnt give us a better score then by using word2vec model



```
document_assembler = DocumentAssembler().setInputCol("plot").setOutputCol("document")

tokenizer = Tokenizer().setInputCols(["document"]).setOutputCol("token")

word_embeddings = BertEmbeddings.pretrained('bert_base_cased', 'en').setInputCols(["document", "token"]).setOutputCol("embeddings")
bert_pipeline = Pipeline(stages =
  [
    document_assembler,tokenizer,word_embeddings

  ]
)

df_bert = bert_pipeline.fit(df).transform(df)

test_dataset = bert_pipeline.fit(test_df).transform(test_df)
```

```
bert_base_cased download started this may take some time.
Approximate size to download 389.2 MB
[OK!]
```

Kaggle competition F1 score:



part3_bert_p.csv 0.94679

a day ago by Sankeerth Tella

Task3 with BERT word embeddings as feature engineering and logistic regression as model.
used own list of stop words for stopwords removal but with various vector sizes.