

Movie Genre Prediction

Team Tacklers

Sai Shanthan Venreddy

Teja Akasapu

Vamshidhar Reddy Komidi

50315873

50314294

50310287

saishant

tejaakas

vamshidh

Java Version changes: We have used java version 8 (by changing it from java 11) in VM

Introduction: The goals of this assignment are to use Spark Libraries to implement an end to end Predictive Analytics Pipeline. Precisely, the objective of the assignment is to implement a movie genre prediction model using Apache Spark. In given data, we are given 31108 rows that correspond to the movie and each movie can have multiple genres which makes it a multi-label classification project.

Task 1 – Term Document Matrix:

We create a machine learning model by using logistic regression in spark to use the information provided in the train set to predict the genres associated with a movie. We create a term-document matrix from the plots and use these as feature vectors for the machine learning model.

A term document matrix is a way of representing the words in the text as a table (or matrix) of numbers. The rows of the matrix represent the text responses to be analyzed, and the columns of the matrix represent the words from the text that are to be used in the analysis. The most basic version is binary. A 1 represents the presence of a word and 0 its absence.

Spark ML standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow.

- **DataFrame:** Spark ML uses DataFrame from Spark SQL as an ML dataset, which can hold a variety of data types. E.g., a DataFrame could have different columns storing text, feature vectors, true labels, and predictions.
- **Transformer:** A Transformer is an algorithm which can transform one DataFrame into another DataFrame. E.g., an ML model is a Transformer which transforms DataFrame with features into a DataFrame with predictions.
- **Estimator:** An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer. E.g., a learning algorithm is an Estimator which trains on a DataFrame and produces a model.
- **Pipeline:** A Pipeline chains multiple Transformers and Estimators together to specify an ML workflow.
- **Parameter:** All Transformers and Estimators now share a common API for specifying parameters.

Pipeline Stages: In this task, we have 3 stages in the pipeline namely Regex Tokenizer, Stopwords, hashingTF.

We use the regexTokenizer, a regex based tokenizer that extracts tokens either by using the provided regex pattern to split the text (default) or repeatedly matching the regex (if gaps is false). Optional parameters also allow filtering tokens using a minimal length. It returns an array of strings that can be empty. This outputs a set of words for each row. Next, we remove the stopwords by using the stopwordsRemover function given in spark.ml library. This acts upon the words column and outputs the filtered column. Next, we used the hashingTF function present in the spark.ml library.

HashingTF is used here to map sequence of terms to their term frequencies. We use the result of hashingTF to transform the document into tf. Basically, convert documents into a numerical representation which can be fed directly or with further processing. 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.

We then use logistic Regression model from spark which fits the model. We use a for loop to iterate through all the labels to train 20 different models and keep adding the predictions to the dataframe. We generate predictions for the test set and convert predictions into a csv file.

F1 Score -0.97380

Task 2 – TF-IDF

Term frequency-inverse document frequency (TF-IDF) is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus. TF and IDF are implemented in HashingTF and IDF. Each record could be an iterable of strings or other types.

Pipeline Stages: In this task, we have 4 stages in the pipeline namely Regex Tokenizer, Stopwords, hashingTF, IDF. We use the regexTokenizer, a regex based tokenizer that extracts tokens either by using the provided regex pattern to split the text (default) or repeatedly matching the regex (if gaps is false). This outputs a set of words for each row. Next, we remove the stopwords by using the stopwordsRemover function given in spark.ml library. This acts upon the words column and outputs the filtered column. Next, we used the hashingTF function present in the spark.ml library. HashingTF is used here to map sequence of terms to their term frequencies. We use the result of hashingTF to transform the document into tf. Next, we use IDF model.

IDF is an Estimator which is fit on a dataset and produces an IDFModel. The IDFModel takes feature vectors created from HashingTF and scales each feature. Intuitively, it down-weights features which appear frequently in a corpus.

We then use logistic Regression model from spark which fits the model. We use a for loop to iterate through all the labels to train 20 different models and keep adding the predictions to the dataframe. We generate predictions for the test set and convert predictions into a csv file.

F1 Score -0.98161


Task 3 – Word2Vec

Pipeline Stages: In this task, we have 3 stages in the pipeline namely Regex Tokenizer, Stopwords, Word2Vec. We use the regexTokenizer, a regex based tokenizer that extracts tokens either by using the provided regex pattern to split the text (default) or repeatedly matching the regex (if gaps is false). This outputs a set of words for each row. Next, we remove the stopwords by using the stopwordsRemover function given in spark.ml library. This acts upon the words column and outputs the filtered column. Next, we use Word2Vec model.

Word2Vec first constructs a vocabulary from the corpus and then learns vector representation of words in the vocabulary. Word2Vec computes distributed vector representation of words. The main advantage of the distributed representations is that similar words are close in the vector space, which makes generalization to novel patterns easier and model estimation more robust.

We then use logistic Regression model from spark which fits the model. We use a for loop to iterate through all the labels to train 20 different models and keep adding the predictions to the dataframe. We generate predictions for the test set and convert predictions into a csv file.

F1 Score -0.99977

| | | | |
|---|---|-------------------------------------|---|
|  | | <input type="text" value="Search"/> | |
| <div><div>Home</div><div>Compete</div><div>Data</div><div>Notebooks</div><div>Discuss</div><div>Courses</div><div>More</div><div>Recently Viewed</div><div>Movie Genre Prediction</div></div> | Overview Data Notebooks Discussion Leaderboard Rules Team | | My Submissions |
| | 5 submissions for Tacklers | | Sort by Most recent |
| | All Successful Selected | | |
| | Submission and Description | | Public Score Use for Final Score |
| | predictions_part3_trial.csv 5 hours ago by Shanthan Venreddy | | 0.99587 <input type="checkbox"/> |
| | trial p3 | | |
| | part3.csv 6 hours ago by Shanthan Venreddy | | 0.99977 <input checked="" type="checkbox"/> |
| | part 3 | | |
| | predictions_part2.csv 3 days ago by Shanthan Venreddy | | 0.98161 <input checked="" type="checkbox"/> |
| | part2 | | |
| | p1_1.csv 3 days ago by Shanthan Venreddy | | 0.97380 <input type="checkbox"/> |
| | part 1 | | |
| | sample.csv 3 days ago by Shanthan Venreddy | | 0.50000 <input type="checkbox"/> |
| | add submission details | | |