ID2221 PROJECT
SENTIMENT ANALYSIS OF TWITTER DATA USING SPARKML

Authors: Rithika Harish Kumar, Catherine Nawire Rakama

ABSTRACT

The main scope of the project is to explore the SparkML library and analyse the sentiment of the tweets. The tweets are trained by various machine learning approaches on Spark. The result from the various approaches are compared in terms of accuracy.

INTRODUCTION

Problem Description

Many organizations today are focusing on utilizing data that is being generated either by their business processes or from their clients through various platforms such social media to get public opinions on their services and products and use the data to predict outcome of certain demand, make strategic decisions or improve their products.

The project will investigate how to utilize data to make predictions that can be useful in making strategic decisions or improve company's products. There are various methods and algorithms that can be used to achieve this and this make it difficult for organizations to make a decision on which approach to use. To address this problem, this project will analyze the sentiment of the tweets to either positive or negative by exploring different machine learning algorithms that runs on Apache PySpark. The outcome will be an evaluation of their performance and prediction accuracy.

The goal here is *not* to apply analysis to any specific application but to test various machine learning[2] models on Pyspark in order to evaluate the accuracy of the predicted sentiments. The choice to twitter sentiment analysis as a use case is because it has many areas of application. The project is an extension of an article[1].

METHODOLOGY

Procedure

Prepare environment by installing Java, Anaconda for Jupyter Notebook and Apache Spark.
Set Environmental Variables for Java and PySpark.
Pip install findspark (only for Windows).
Set Pyspark driver for python.
Run jupyter notebook.

Data Collection and Analysis

Dataset used is annotated tweets from "http://help.sentiment140.com/for-students/" provided by Stanford. The dataset has information about the sentiment, id, date, query, user and text of the tweet. We extract the sentiment and the text for our analysis. The value is 0 for negative tweets and 4 for positive tweets.

Methods

Data preprocessing was done by using BeautifulSoup library. Decoding HTML to general text, removing UTF-8 byte order mark, removing 'http' and 'www', converting to lowercase, removing numbers and special characters, tokenizing and joining are the steps carried out to clean the data. After which the tweet data and its sentiment were saved in csv file.

A *SparkContext* was created to load the cleaned tweets to a DataFrame. The data was split into training(98%) and testing(2%) data. Various classification algorithms were used to predict the sentiment of the tweets i.e. either positive(1) or negative(0). Neutral is not taken into account for this project.

For all the models, *Pipeline* was used to streamline the feature extraction process. Feature extraction is done using methods like *TF-IDF[1]*, *CountVectorizer[1]*, *Word2Vec* and *NGram[1]*. Then they were trained with *Logistic Regression*. A feature technique that gives the best accuracy is used for comparison with classification algorithms like *Random Forest*, *Naive Bayes* to compare and evaluate.

Algorithms

The following models were tested:

- TF-IDF and Logistic Regression.
- CountVectorizer, IDF and Logistic Regression.
- Word2Vec and Logistic Regression.
- NGram with Logistic Regression.
- Ngram with Random Forest Classifier.
- Ngram with Naive Bayes.

TF-IDF and Logistic Regression:

Feature extraction is done by converting the tweets to tokens and then calculating the term frequency and inverse document frequency. The top features are filtered by dimensionality reduction with possible collisions. After extracting the features, the model is trained with logistic regression.

CountVectorizer, IDF and Logistic Regression:

Feature extraction is done by converting the tweets to tokens and then calculating the count vectoriser and inverse document frequency.The top features are filtered by discarding infrequent tokens. After extracting the features, the model is trained with logistic regression.

Word2Vec and Logistic Regression:

Tweets are converted to tokens and word2vec is used for feature extraction. After extracting the features, the model is trained with logistic regression.

NGram and Logistic Regression:

5,460 features each from unigram, bigram, trigram were extracted, to have around 16,000 features totally. Vector assembler was used in the pipeline to combine the features. After extracting the features, the model is trained with logistic regression.

Among the above feature extractors, NGram performed fairly better, therefore these features were used to compare various classifiers.

NGram and Random Forest Classifier:

Random forests train a set of decision trees separately, so the training can be done in parallel. Each tree's prediction is counted as a vote for one class. The label is predicted to be the class which receives the most votes.

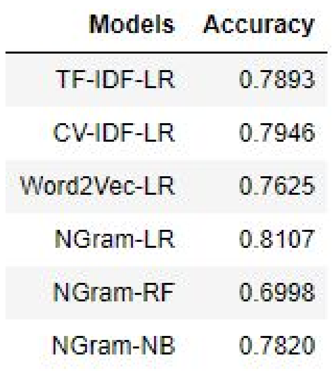
NGram and Naive Bayes:

It computes the conditional probability distribution of each feature given label, and then it applies Bayes' theorem to compute the conditional probability distribution of label given an observation and use it for prediction. Multinomial naive bayes with smoothing value of 1 was used to train and classify the tweets. Each feature represents a term whose value is the frequency of the term.

RESULTS

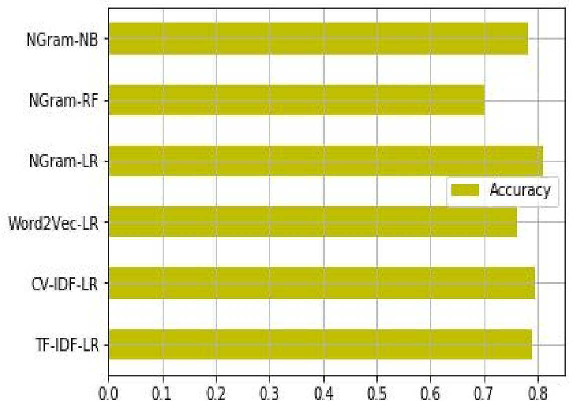
The evaluation is done based on the accuracy obtained for each of the models. The accuracy is calculated by counting the number of predictions matching the label and dividing it by the total entries.
The table shows the respective accuracy scores for the models.

Tabulation of Scores



Models	Accuracy
TF-IDF-LR	0.7893
CV-IDF-LR	0.7946
Word2Vec-LR	0.7625
NGram-LR	0.8107
NGram-RF	0.6998
NGram-NB	0.7820

Bar Graph showing accuracy comparisons of various models



CONCLUSION

Thus, among the various methods NGram with Logistic Regression provides a better accuracy score of 81%(0.8107) comparatively.

REFERENCES

[1]https://towardsdatascience.com/sentiment-analysis-with-pyspark-bc8e83f80c35
[2]https://spark.apache.org/docs/latest/ml-guide.html