

BIA 678 Final Project
Twitter Sentiment Analysis Using Spark

Shikuan Gao
Rhutvij Savant

Abstract

The topic of our group final paper is twitter sentiment analysis using Spark. Nowadays, twitter has a wealth of data for those trying to understand how people feel about brands, topics, and more. Mining twitter data for insights is one of the most common natural language processing tasks. After twitter sentiment analysis, we believe the analysis of tweets will make it simple to understand when people think positively or negatively about a certain keyword. These analyzed tweets are especially valuable when there are many tweets around a subject. The project aims to implement our work in the social network media and finance companies as well as interfaces of our systems, while making our lives better and our experience richer and efficient.

Introduction

Twitter sentiment analysis is using Python API to stream tweets about one or more hashtag and classify the tweets as positively or negatively inclined to give an idea on what people think about the topic without having to read hundreds of tweets. As we all know, Spark is a fast and general cluster computing system for Big Data. It provides high-level APIs in Scala, Java, Python, and R, and an optimized engine that supports general computation graphs for data analysis. It also supports a rich set of higher-level tools including Spark SQL for SQL and Data Frames, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for stream processing. There are multiple ways about Spark performance in different programming languages. The main language we learned here is Python, so in our project we will use Pyspark as one of the python API for spark. One of the most important contents in Pyspark package is called spark

context, it is the main entry point for Spark functionality. We will mention Pyspark and its package more in the later section of our project paper.

Objective

In our project, we are first trying to build a sentiment analysis model with the dataset we have, in this case the model can be applied to more future uses. Once the model is built, tweets from different sources will be analyze and sentiment analysis will be performed on future gathered data to analyses positivity/negativity in different sources and improve so much efficiency in this area.

Second, the goal of our project is to predict sentiment for the given twitter post using Spark. Also, the objective of this project is to show how sentiment analysis can help improve the user experience over a social network or system interface. In our understanding of programming skills, sentiment analysis can predict many different emotions attached to the text, but in this paper only 2 majors we are considering which are positive and negative tweets.

Data

The dataset we use for this project is annotated tweets from Sentiment 140. This dataset is originated from a Stanford research project. Dataset has 1.6million observations, with no null entries, and importantly for the "sentiment" column, even though the dataset description mentioned neutral class, the training set has no neutral class. 50% of the data is with negative label, and another 50% with positive label. From this we can see there's no skewness on the class division.

	sentiment	id	date	query_string	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zi - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElieCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all...

Figure.1

From the figure shown above, the data file format has 6 fields, there is "sentiment" column which is the polarity of the tweet (0 = negative, 4 = positive), "id" column is unique ID for each tweet (1467810369), "date" column is the date of the tweet Mon APR 06 22:19:45 PDT 2009), "query " column indicates whether the tweet has been collected with any particular query keyword, but for this column, 100% of the entries are with value "NO_QUERY", "user" column is the twitter handle name for the user who tweeted (mattycus), and finally the text of the tweet.

Exploratory Data Analysis

Now that we have described the features in the dataset, we can carry out exploratory data analysis for better understanding of the data. It is important to perform data analysis to ensure that there is no sample bias in the dataset so that it will not have any effect on the implementation results. In the real world the tweets posted on various topics may be positive as well as negative and if the training data has a high percentage of either negative or positive sentiment tweets then that could affect the implementation results when we run our prediction on real world twitter data. From our exploratory data analysis, it is evident that half the tweets have positive sentiment and the remaining half has negative sentiment tweets. The dataset has a collection of 1.6 million tweets collected over a period of three months of April, May and June of the year 2009.

Given below is a chart that shows the analysis of total number of tweets posted on any given day, the chart is dynamic and can also show the monthly or weekly tweets.

Total Number of Tweets

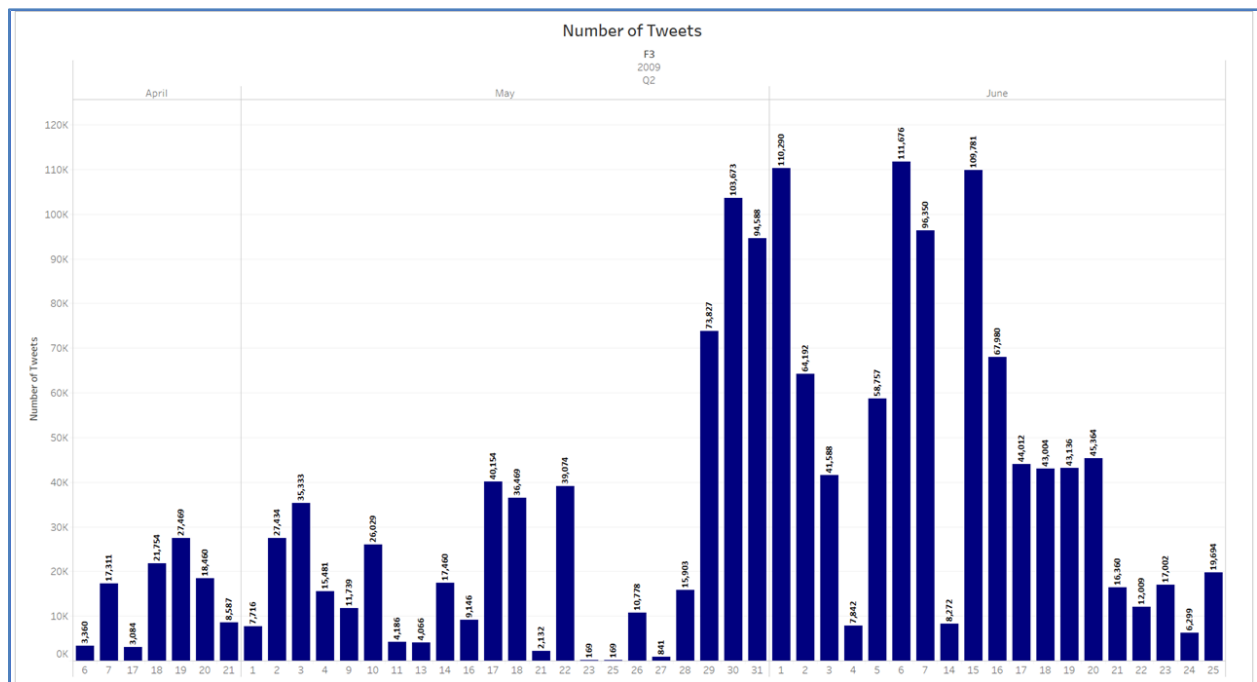


Figure.2

The chart below is an analysis of average number of words for each of the tweets. As we are computing a large number of tweets and in such projects the social media data is always increasing, this analysis helps us to understand the characteristics of the tweets posted and the computational power that may be necessary in the future. On average the users type 14 to 15 words per tweet

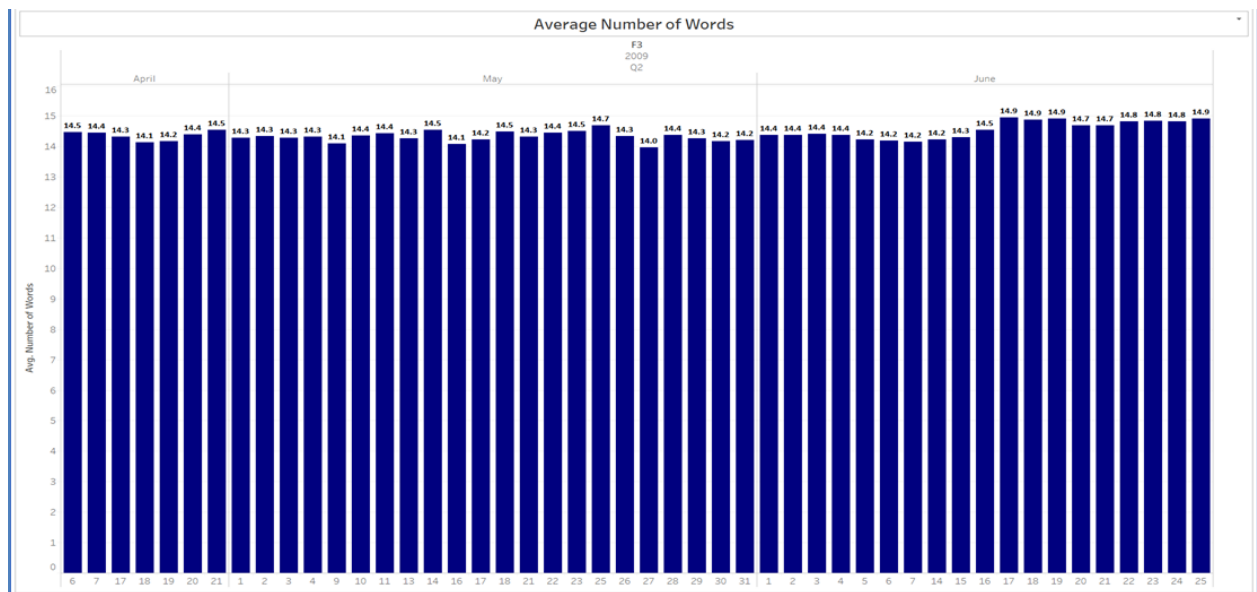


Figure.3

The chart below is an analysis of the average number of characters used in a tweet or length of the tweet. On average there are 70 to 75 characters used in each tweet. This analysis is similar to the data analysis for average number of words given above and is important to explain the characteristics of the tweets.

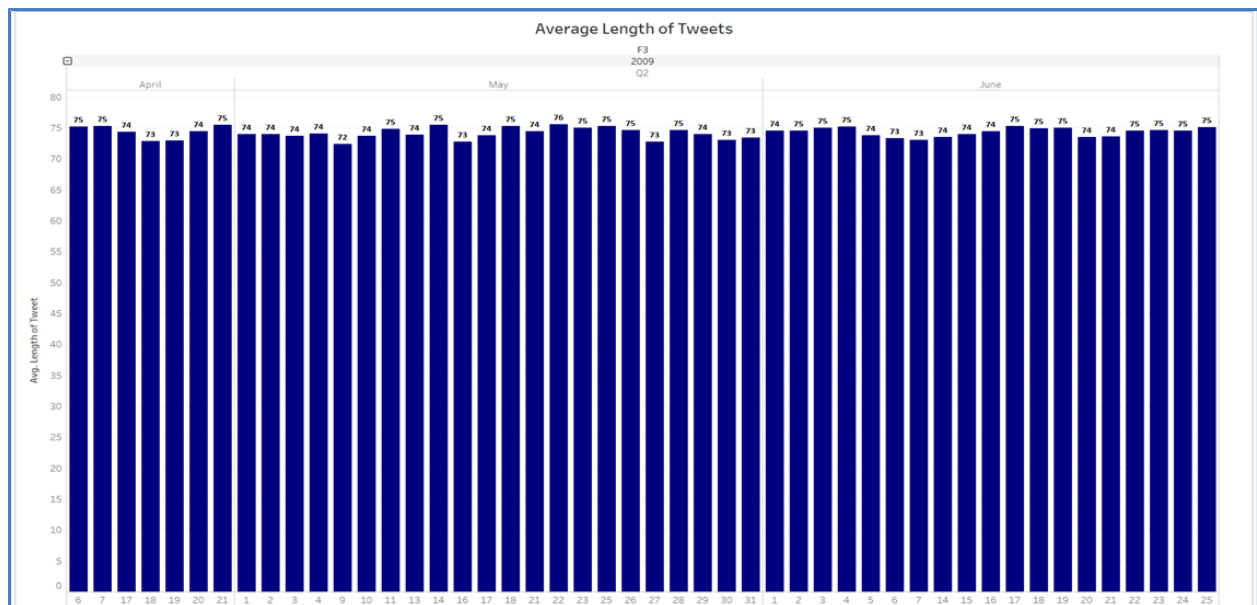


Figure.4

The analysis given below represents the percentage of positive and negative tweet sentiment analyzed over a period of time. It describes the percent of tweets that are positive and negative in a month and can be expanded to a daily analysis.

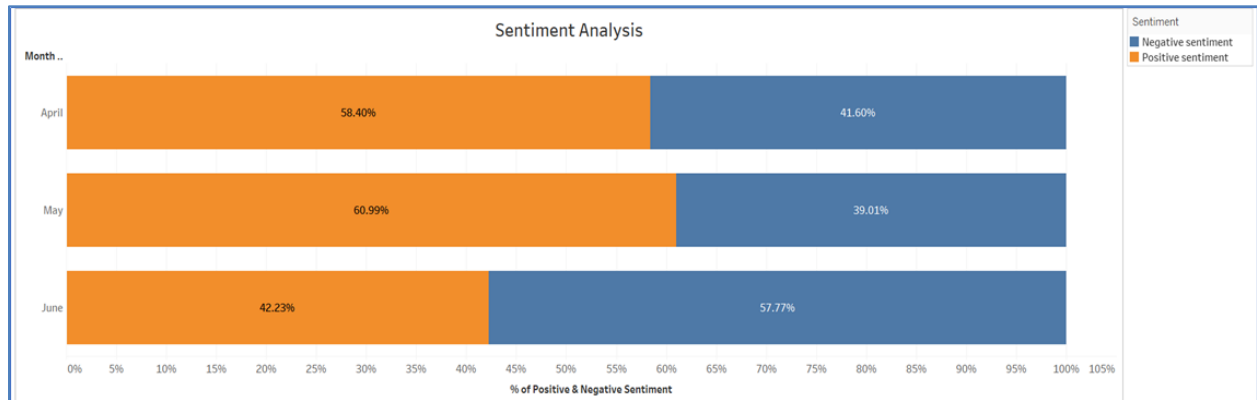


Figure.5

There is a total of about 660,000 users in the dataset and the chart below shows the analysis of the average number of tweets made by the users daily and can also be dynamically analyzed for a weekly or monthly basis. On average the users have posted between one to two tweets on any given day.

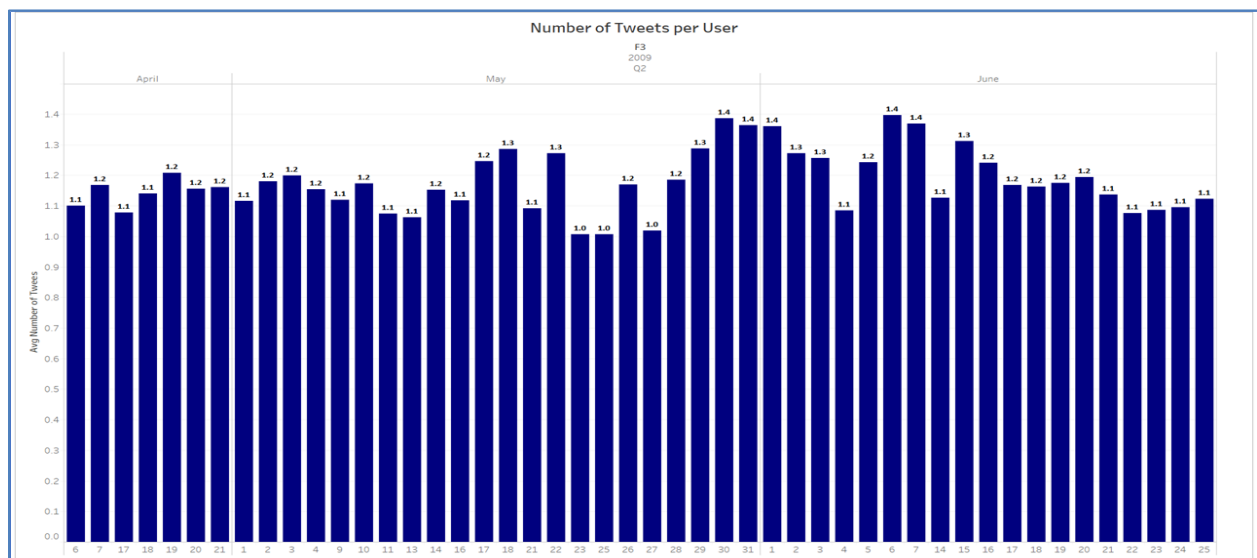


Figure.6

Data Preparation

Data preparation is the process of gathering, combining, structuring and organizing data so it can be analyzed as part of data visualization, analytics and machine learning applications. In our project, there are five data preparations we are thinking to deal with. In our project, we are going to discuss these in the following five subsections.

Data Preparation 1: HTML decoding

In the dataset we have, the HTML encoding has not been converted to text, and ended up in text field as '&','"', etc. In our first step of data preparations, we will use BeautifulSoup the Python package to decoding HTML to general text.

Data Preparation 2: @mention

The second part of the preparation is to deal with @mention. Even the @mention brings a certain information of another user in the whole tweets, but this information is not valuable for our project to build sentiment analysis model.

Data Preparation 3: URL links

The third part of the preparation is to deal with URL links, URL links brings the same with @mention in the whole tweets, this information is not valuable for our project to build sentiment analysis model.

Data Preparation 4: UTF-8 BOM (Byte Order Mark)

The fourth part of the preparation is to deal with UTF-8 BOM (Byte Order Mark). The UTF-8 BOM is a sequence of bytes (EF BB BF) that allows the reader to identify a file as being encoded in UTF-8. Normally, the BOM is used to signal the endianness of an encoding, but since endianness is irrelevant to UTF-8, the BOM is unnecessary. In the dataset we have, the strange patterns of characters "\xef\xbf\xbd" which are UTF-8 BOM. After using python code to decode text with 'utf-8-sig', this BOM will be replaced with Unicode unrecognizable special characters.

Data Preparation 5: hashtag / numbers

The last part of the preparation is to deal with hashtag and numbers. Because the text used with a hashtag cannot provide useful information about the tweet. We decided to leave the text intact and just remove the '#'.

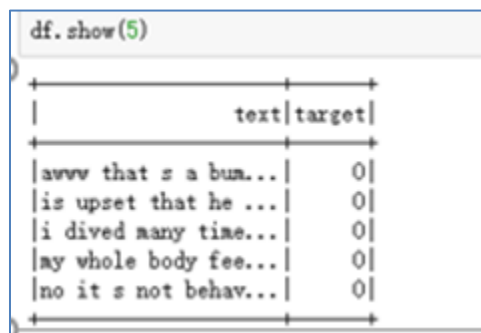
A	B
text	target
awww that s a bummer you shoulda got david carr of third day to do it d	0
is upset that he can t update his facebook by texting it and might cry as a result school today also blah	0
i dived many times for the ball managed to save the rest go out of bounds	0
my whole body feels itchy and like its on fire	0
no it s not behaving at all i m mad why am i here because i can t see you all over there	0
not the whole crew	0
need a hug	0
hey long time no see yes rains a bit only a bit lol i m fine thanks how s you	0
k nope they didn t have it	0
que me muera	0
spring break in plain city it s snowing	0
i just re pierced my ears	0
i couldn t bear to watch it and i thought the va loss was embarrassing	0
it it counts idk why i did either you never talk to me anymore	0
i would ve been the first but i didn t have a gun not really though zac snyder s just a doucheclown	0
i wish i got to watch it with you i miss you and how was the premiere	0
holis death scene will hurt me severely to watch on film wry is directors cut not out now	0
about to file taxes	0
ahh ive always wanted to see rent love the soundtrack	0
oh dear were you drinking out of the forgotten table drinks	0
i was out most of the day so didn t get much done	0
one of my friend called me and asked to meet with her at mid valley today but i ve no time sigh	0
barista i baked you a cake but i ate it	0
this week is not going as i had hoped	0
blagh class at tomorrow	0
i hate when i have to call and wake people up	0
just going to cry myself to sleep after watching marley and me	0
im sad now miss lily	0
ooooh lol that leslie and ok i won t do it again so leslie won t get mad again	0
meh almost lover is the exception this track gets me depressed every time	0
some hacked my account on aim now i have to make a new one	0
i want to go to promote gear and groove but unfortunately no ride there i may b going to the one in anaheim in may though	0
thought sleeping in was an option tomorrow but realizing that it now is not evaluations in the morning and work in the afternoon	0
aww i love you too am here i miss you	0
i m sorry again ever to sleep at night	0

Figure.7

As the figure shows above, this is our project final clean tweets data save as csv. file.

Initial Spark

As we mentioned in above section, one of the most important contents in Pyspark package is called spark context, it is the main entry point for Spark functionality. In order to use PySpark in Python, we use a package called Findspark to make a spark context available in Python Jupyter Notebook. Then, we can load the Dataframe in Spark context, the following figure shows the below, it is the first five rows of data shows as Sparkcontext.



```
df.show(5)
```

text	target
awww that s a bum...	0
is upset that he ...	0
i dived many time...	0
my whole body fee...	0
no it s not behav...	0

Figure.8

Tokenization and Term Frequency

The model we used in our project is tokenization and term frequency which are two commons model in natural language processing tasks. Natural language processing (NLP) is a subfield of computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. Tokenizer splits text into tokens. These tokens could be paragraphs, sentences or individual words. Here, we are tokenizing into words. Term frequency is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus.

Results

In our first results using binary classification evaluator, we have our raw prediction as about 86% shows as below figure.

```
: from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")
evaluator.evaluate(predictions)
: 0.8608587198369572
```

Figure.9

Second, there is another that we can get term frequent results calculation. It is CountVectorizer in Spark ML. The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. We have our results using CountVectorizer combine with logistic regression accuracy is about 80% with ROC- AUC 0.8675 shows as below figure.

```
Accuracy Score: 0.7979
ROC-AUC: 0.8675
CPU times: user 82 ms, sys: 18 ms, total: 100 ms
Wall time: 4min 48s
```

Figure.10

Conclusion

Sentiment analysis can be used on social media big data to gain valuable business insights for various consumer, media and finance companies. From our implementation results we have seen that it is possible to predict the positive and negative emotions of users in tweets with more than 80% accuracy. There are millions of tweets each day and a large amount of data is generated. We have used Spark and natural language processing for the implementation of sentiment analysis and these projects can be replicated for various real world companies to make important business decisions after they analyze their customer sentiment from tweets. As the amount of twitter data is ever increasing this will help improve the accuracy of the models and performance of the projects. The analysis of such user sentiment will be beneficial for making key business decisions and future strategies.

Appendix

Codes:

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.2.1-bin-hadoop2.7"
```

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
sc = spark.sparkContext
sc
```

```
import findspark
findspark.init()
import pyspark as ps
import warnings
from pyspark.sql import SQLContext

try:
    sqlContext = SQLContext(sc)
    print("Just created a SparkContext")
except ValueError:
    warnings.warn("SparkContext already exists in this scope")
```

```
df = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true').load('../datalab/clean_tweet.csv')
```

```
type(df)
```

```
pyspark.sql.dataframe.DataFrame
```

```
df.show(5)
```

text	target
i saw that s a bum...	0
is upset that he ...	0
i dived many time...	0
my whole body fee...	0
no it s not behav...	0

```

tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashtf = HashingTF(numFeatures=2**16, inputCol="words", outputCol='tf')
idf = IDF(inputCol='tf', outputCol="features", minDocFreq=5) #minDocFreq: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label")
pipeline = Pipeline(stages=[tokenizer, hashtf, idf, label_stringIdx])

pipelineFit = pipeline.fit(train_set)
train_df = pipelineFit.transform(train_set)
val_df = pipelineFit.transform(val_set)
train_df.show(5)

```

text	target	words	tf	features	label
a	0	[a]	(65536, [30802], [1...]	(65536, [30802], [1...]	0.0
a actually don t ...	0	[a, actually, don...	(65536, [1903, 1588...	(65536, [1903, 1588...	0.0
a actually due to...	0	[a, actually, due...	(65536, [338, 1903...	(65536, [338, 1903...	0.0
a ah kan fb nih n...	0	[a, ah, kan, fb, ...]	(65536, [546, 6387...	(65536, [546, 6387...	0.0
a airfranceflight	0	[a, airfranceflight]	(65536, [30802, 527...	(65536, [30802, 527...	0.0

only showing top 5 rows

```

: from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")
evaluator.evaluate(predictions)

```

```

: 0.8608587198369572

```

```

: accuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(val_set.count())
accuracy

```

```

: 0.7912807621063773

```

```

from pyspark.ml.feature import CountVectorizer

```

```

tokenizer = Tokenizer(inputCol="text", outputCol="words")
cv = CountVectorizer(vocabSize=2**16, inputCol="words", outputCol='cv')
idf = IDF(inputCol='cv', outputCol="features", minDocFreq=5) #minDocFreq: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label")
lr = LogisticRegression(maxIter=100)
pipeline = Pipeline(stages=[tokenizer, cv, idf, label_stringIdx, lr])

pipelineFit = pipeline.fit(train_set)
predictions = pipelineFit.transform(val_set)
accuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(val_set.count())
roc_auc = evaluator.evaluate(predictions)

```

```

print ("Accuracy Score: {0:.4f}".format(accuracy))
print ("ROC-AUC: {0:.4f}".format(roc_auc))

```

```

Accuracy Score: 0.7979

```

```

ROC-AUC: 0.8675

```

```

CPU times: user 82 ms, sys: 18 ms, total: 100 ms

```

```

Wall time: 4min 48s

```