

Problem Statement

Go through below blog and reiterate the same at your end.

https://docs.google.com/document/d/14YUd_wi-KJTBEqqtYoNtPMFeORjFU-yld4hJ11q90o/

(Sentiment_analysis_on_demonetization.pdf)

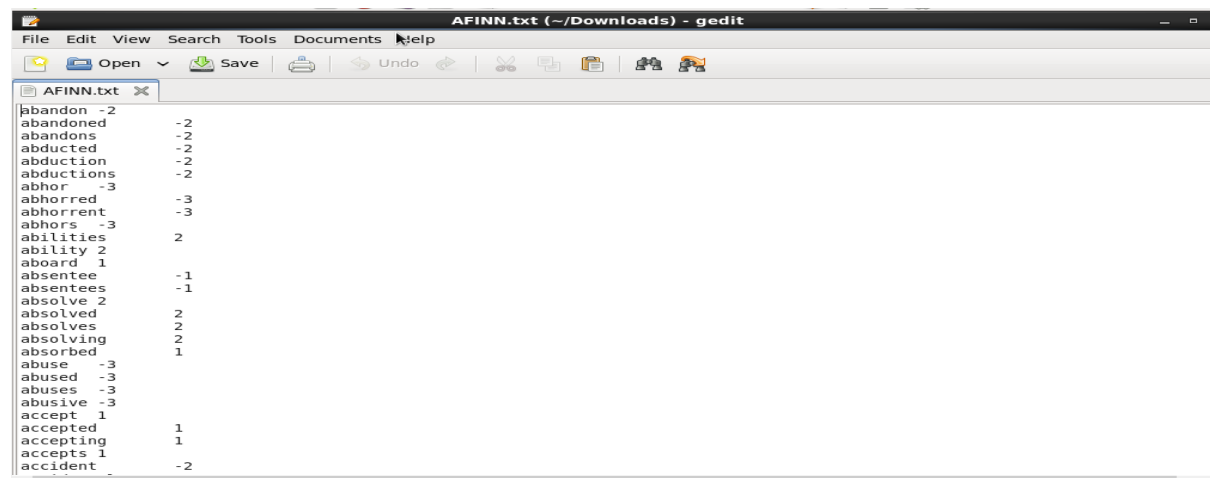
Let us find out the views of different people on the demonetization by analysing the tweets from twitter. Here is the dataset where twitter tweets are gathered in CSV format. You can download the dataset from the below link

<https://drive.google.com/open?id=0ByJLBTmJojzNkRsZWJiY1VGc28>

DataSet

AFINN.txt

Below is a sample of the data. It has 2 columns: **word**, **rating**

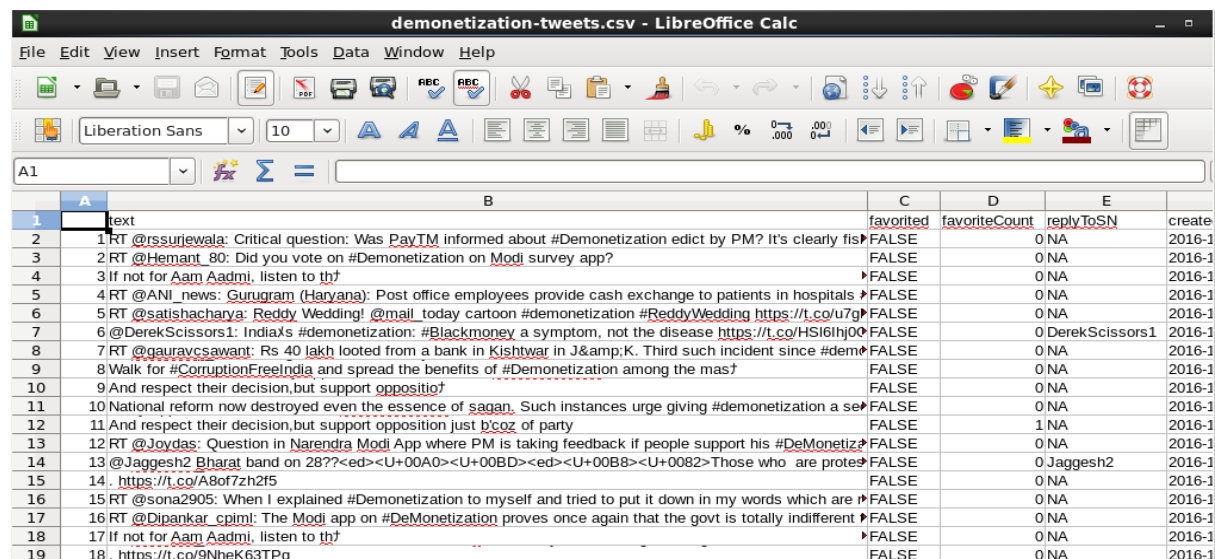


word	rating
abandon	-2
abandoned	-2
abandons	-2
abducted	-2
abduction	-2
abductions	-2
abhor	-3
abhorred	-3
abhorrent	-3
abhors	-3
abilities	2
ability	2
aboard	1
absentee	-1
absentees	-1
absolve	2
absolved	2
absolves	2
absolving	2
absorbed	1
abuse	-3
abused	-3
abuses	-3
abusive	-3
accept	1
accepted	1
accepting	1
accepts	1
accident	-2

demonetization-tweets.csv

Below is a sample of the data. It has 15 columns:

#, **text**, **favorited**, **favoriteCount**, **replyToSN**, **created**, **truncated**, **replyToSID**, **id**, **replyToUID**, **statusSource**, **screenName**, **retweetCount**, **isRetweet**, **retweeted**



#	text	favorited	favoriteCount	replyToSN	created
1	RT @rsshurjewala: Critical question: Was PayTM informed about #Demonetization edict by PM? It's clearly f...	FALSE	0	NA	2016-1
2	RT @Hemant_80: Did you vote on #Demonetization on Modi survey app?	FALSE	0	NA	2016-1
3	If not for Aam Aadmi, listen to th...	FALSE	0	NA	2016-1
4	RT @ANI_news: Gurugram (Haryana): Post office employees provide cash exchange to patients in hospitals	FALSE	0	NA	2016-1
5	RT @satishacharya: Reddy Wedding! @mail_today cartoon #demonetization #ReddyWedding https://t.co/u7g...	FALSE	0	NA	2016-1
6	@DerekScissors1: India's #demonetization: #Blackmoney a symptom, not the disease https://t.co/HSI6lhj00...	FALSE	0	DerekScissors1	2016-1
7	RT @gauravcsawant: Rs 40 lakh looted from a bank in Kishtwar in J&K. Third such incident since #dem...	FALSE	0	NA	2016-1
8	Walk for #CorruptionFreeIndia and spread the benefits of #Demonetization among the mas...	FALSE	0	NA	2016-1
9	And respect their decision, but support oppositio...	FALSE	0	NA	2016-1
10	National reform now destroyed even the essence of sagan. Such instances urge giving #demonetization a se...	FALSE	0	NA	2016-1
11	And respect their decision, but support opposition just b'coz of party	FALSE	1	NA	2016-1
12	RT @Joydas: Question in Narendra Modi App where PM is taking feedback if people support his #DeMonetiz...	FALSE	0	NA	2016-1
13	@Jaggesh2 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protes...	FALSE	0	Jaggesh2	2016-1
14	https://t.co/A8of7zh2f5	FALSE	0	NA	2016-1
15	RT @sona2905: When I explained #Demonetization to myself and tried to put it down in my words which are	FALSE	0	NA	2016-1
16	RT @Dipankar_cpimi: The Modi app on #DeMonetization proves once again that the govt is totally indifferent	FALSE	0	NA	2016-1
17	If not for Aam Aadmi, listen to th...	FALSE	0	NA	2016-1
18	https://t.co/9NheK63TPq	FALSE	0	NA	2016-1

Solution:

- First Import the spark and sql packages to be used for creating dataframes.
- Begin by reading the data file **demonetization-tweets.csv** as a text file from the local FS using the spark context object **sc**.
- The data is then split by comma, as it is from a comma separated value (csv) file and filtered such that the all rows with length greater than 2.
- Next, the data is mapped to only the 1st and 2nd columns where all the '\ ' are removed and the <space> in the 2nd row is also removed.
- Finally, the result from above is transformed into a DataFrame with 2 columns named: id, words. **rdd5df** .
- The data from the DataFrame **rdd5df** is used to create a temporary table **tweets**. Query the temp table **tweets** so as to get the **id** and all the **words** in the 2nd column.
- Using the explode function, every element is separated and a new row created for each. Ex: (1, Hi Hello How) becomes (1, Hi), (1, Hello), (1, How)
- Then, the result is used to create another temporary table **tweet_word**. This is val **explode**.
- We then read the data file **AFINN.txt** as a text file from the local FS using the spark context object **sc**.
- This data is then mapped to its two rows by splitting them by <tab> and transforming it into a DataFrame with columns: **word, rating**.
- Then, the data from the DataFrame is used to create a temporary table **afinn**.
- Join and query the tables **tweet_word** (as **t**) and **afinn** (as **a**).
- Group the data by id. This will give us a group of data corresponding to every id .
- From every group where every word in **t** that matches the word in **a**, we can now select the **id** (from t) and average of all the ratings for that id (from **a**).
- The data is lastly ordered by highest rating first (DESC).
- Display the output of Join in SPARK SQL operation by using **join.show()**.
- Display the entire output of join operation by using **join.foreach(println)**
- Saving the output to csv file at local file system where columns are separated by underscore

Code:

```
scala> import org.apache.spark.sql._
import org.apache.spark.sql._

scala> import sqlContext.implicits._
import sqlContext.implicits._

scala> val tweets = sc.textFile("file:///home/acadgild/Downloads/demonetization-tweets.csv")
tweets: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[116] at textFile at <console>:48

scala> val rdd1 = tweets.map(x => x.split(","))
rdd1: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[117] at map at <console>:50

scala> val rdd2 = rdd1.filter(x=>(x.length>=2))
rdd2: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[118] at filter at <console>:52

scala> val rdd3 = rdd2.map(x => (x(0).replaceAll("\\", ""),x(1).replaceAll("\\", "").toLowerCase))
rdd3: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[119] at map at <console>:54

scala> val rdd4 = rdd3.map(x => (x._1,x._2.split(" ")))
rdd4: org.apache.spark.rdd.RDD[(String, Array[String])] = MapPartitionsRDD[120] at map at <console>:56

scala> val rdd5df =rdd4.toDF("id","words")
rdd5df: org.apache.spark.sql.DataFrame = [id: string, words: array<string>]

scala> rdd5df.registerTempTable("tweets")

scala> val rdd6 = sqlContext.sql("select id as id,explode(words) as word from tweets")
rdd6: org.apache.spark.sql.DataFrame = [id: string, word: string]

scala> val explode = rdd6.registerTempTable("tweet_word")
explode: Unit = ()

scala> val afinn = sc.textFile("file:///home/acadgild/Downloads/AFINN.txt")
afinn: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[123] at textFile at <console>:48
```

```
scala> val rddx = afinn.map(x => x.split("\t"))
rddx: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[124] at map at <console>:50

scala> val rddy = rddx.map(x => (x(0),x(1)))
rddy: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[125] at map at <console>:52

scala> val rddzdf= rddy.toDF("word","rating")
rddzdf: org.apache.spark.sql.DataFrame = [word: string, rating: string]

scala> val rdda = rddzdf.registerTempTable("afinn")
rdda: Unit = ()

scala> val join = sqlContext.sql("SELECT t.id,AVG(a.rating) AS rating FROM tweet_word t JOIN afinn a WHERE t.word = a.word GROUP BY t.id ORDER BY rating DESC")
join: org.apache.spark.sql.DataFrame = [id: string, rating: double]
```

Output:

```
scala> join.show()
+-----+-----+
|   id|rating|
+-----+-----+
|6610|   4.0|
|7025|   4.0|
|7281|   4.0|
|6546|   4.0|
|3822|   4.0|
|4185|   4.0|
|5733|   4.0|
|7994|   4.0|
| 308|   3.5|
|5702|   3.0|
|7772|   3.0|
|5551|   3.0|
|5829|   3.0|
|7393|   3.0|
|2377|   3.0|
|1811|   3.0|
|7825|   3.0|
|6164|   3.0|
|3494|   3.0|
|2654|   3.0|
+-----+-----+
only showing top 20 rows

scala> █
```

For Showing entire output

```
scala> join.foreach(println)█
```

For Saving the output as TextFile

```
scala> join.map(_._mkString("_")).saveAsTextFile("file:/home/acadgild/Downloads/Assignment20_2Tweets.txt")
scala> █
```