Module 8: Deep Dive into Spark MLlib

Case Study II Solution

Case Study: Email Analytics
Domain: IT Security Firm

IT International (ITI) is leading the development of new software that could revolutionize how computers support decision-makers.
The IT security team of ITI building this web-based tool for Enron to

- gain insights from the emails in case of fraud and
- identify any abnormal behavior in the email communication to prevent the unexpected.

The dataset contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M messages.

Tasks:

As part of the BigData consultant you are expected to implement the following use cases:

1. Load data into Spark DataFrame
2. Display the top 10 high-frequency users based on weekly numbers of emails sent
3. Extract top 20 keywords from the subject text for both
   - for the top 10 high-frequency users and
   - for the non-high frequency users
4. Extract top 10 keywords by identifying removing the common stop words.
5. Extend the stop words dictionary by adding your own stop words such as '__ '
6. Introduce a new column label to identify new, replied, and forwarded messages
7. Get the trend of the over mail activity using the pivot table from spark itself
8. Use k-means clustering to create 4 clusters from the extracted keywords
9. Use LDA to generate 4 topics from the extracted keywords
10. Can you identify top keywords in the spam messages across the organization?

Solution

- Download and Extract Dataset
  https://www.cs.cmu.edu/~./enron/enron_mail_20150507.tar.gz

- This dataset is quite large, so you can get the part of data by pressing ctrl+c while extraction is in progress
  tar -zxvf enron_mail_20150507.tar.gz

- Clean the dataset

- For the current use case, we are considering only sent mails from a limited set of users:
  mkdir mail_dataset

  cp -r ../maildir/a*/sent/

- Remove ^M from the files:
  sed -i -e 's/\r//g' sent/*

- Extracting lines and creating a single output file. Here we have put the separator as #
  head -q -n 5 sent/* | paste - - - - - -d# > output.csv

  - *Head -q -n 5*: head the file and output 5 lines
  - *Paste - - - - - -d#*: join 5 lines with # as delimiter.

- Loading datasets into HDFS
  hdfs dfs -mkdir use_cases/maildir

  hdfs dfs -copyFromLocal output.csv use_cases/maildir

  - - -d# > output.csv

- You should now enter inside the PySpark Shell. ***Type pyspark2*** to enter the python spark shell for running further codes.

- You should now enter inside the PySpark Shell. ***Type*** ***pyspark2*** to enter the python spark shell for running further codes.

Step 1: Load data into Spark DataFrame

```
import pyspark.sql.functions as F

raw =
spark.read.option("delimiter","#").csv("use_cases/maildir/output.csv").toDF("messageid","date","from_","to_","subject")

df =
raw.withColumn("date",F.trim(F.split(raw.date,":")[1])).withColumn("from_",F.trim(F.split(raw.from_,":")[1])).withColumn("to_",F.trim(F.split(raw.to_,":")[1])).withColumn("subject",F.trim(F.split(raw.subject,"Subject:")[1]))
```

Step 2: Display the top 10 high frequency users based on weekly numbers of mails send

```
df1 = df.withColumn("week",F.weekofyear(F.unix_timestamp(df.date,"EEE, dd MMM yyyy HH").cast("timestamp")))

maxweek = df1.agg(F.max(df1.week)).first()[0]

df1.groupBy("from_").count().withColumn("avgcount",F.col("count")/maxweek).sort(F.col("avgcount").desc()).show()
```

- Tokenize the dataset

```
from pyspark.ml.feature import Tokenizer

tokenizer = Tokenizer().setInputCol("subject").setOutputCol("words")

transformed = tokenizer.transform(df1)
```

Step 3: Extract top 20 keywords from the subject text for both 1) for the top 10 high frequency users and 2) for the non-high frequency users

```
top_users = [v.asDict()["from_"] for v in
df1.groupBy("from_").count().sort(F.col("count").desc()).take(10)]
```

```
topuserdata = transformed.filter(transformed.subject != “”).filter(transformed.from_.isin(top_users))

topuserdata.withColumn("keyword",F.explode("words")).groupBy("keyword").count().sort(F.col("count").desc()).show(20)

otheruserdata = transformed.filter(transformed.subject != "").filter(transformed.from_.isin(top_users) == False)

otheruserdata.withColumn("keyword",F.explode("words")).groupBy("keyword").count().sort(F.col("count").desc()).show(20)
```

Step 4: Extract top 10 keywords from subject by identifying removing the common stop words

```
from pyspark.ml.feature import StopWordsRemover

remover = StopWordsRemover().setInputCol("words").setOutputCol("filtered")

cleaned = remover.transform(transformed)

cleaned.filter(cleaned.subject != "").withColumn("keyword",F.explode(cleaned.words)).groupBy("keyword").count().sort(F.col("count").desc()).show()
```


Step 5: Extend the stop words dictionary by adding your own stop words such as ‘—‘

```
stopwords = StopWordsRemover().getStopWords()  + ["-","re:","fw:"]

remover = StopWordsRemover().setStopWords(stopwords).setInputCol("words").setOutputCol("filtered")

cleaned = remover.transform(transformed)

cleaned.filter(cleaned.subject != "").withColumn("keyword",F.explode(cleaned.words)).groupBy("filtered").count().sort(F.col("count").desc()).show()
```

Step 6: Introduce a new column label to identify new, replied, and forwarded messages

```
df2 = cleaned.withColumn("msgtype",when(cleaned.subject.startswith("Re:"),1).otherwise(when(cleaned.subject.startswith("Fw:"),2).otherwise(0)))
```

Step 7: Get the trend of the over mail activity using the pivot table from spark itself

```
df2.groupBy("week").pivot("msgtype").count().show()
```

- Convert keywords to feature vector

```
from pyspark.ml.feature import CountVectorizer, CountVectorizerModel
df4 = df2.filter(df2.subject !== "")
cvmodel = CountVectorizer().setInputCol("filtered").setOutputCol("features").fit(df4)
featured = cvmodel.transform(df4)
```

Step 8: Use kmeans clustering to create 4 clusters from the extracted keywords

```
from org.apache.spark.ml.clustering import KMeans
kmeans = KMeans().setK(4).setSeed(1L)
model = kmeans.fit(featured)
predictions = model.transform(featured)
```

Step 9: Use LDA to generate 4 topics from the extracted keywords

```
from pyspark.ml.clustering import LDA
lda = LDA().setK(4).setMaxIter(10)
model = lda.fit(featured)
topics = model.describeTopics(4)
topic_indices = topics.select("termIndices").rdd.map(lambda x:x[0][0]).collect()
[cvmodel.vocabulary[v] for v in topic_indices]
```

Step 10: Can you identify top keywords in the spam messages across organization