CSK180002

PROJECT PART 2

CHINMAY SUNIL KARANDIKAR

Table of Contents

1. Explore the dataset and provide analysis by product-category and year: 2
    1. Number of reviews 2
    2. Number of distinct users 2
    3. Average and Median review stars 3
    4. Percentiles of length of the review. Use the following percentiles: [0.1, 0.25, 0.5, 0.75, 0.9, 0.95] 4
    5. Percentiles for number of reviews per product. For example, 10% of books got 5 or less reviews. Use the following percentiles: [0.1, 0.25, 0.5, 0.75, 0.9, 0.95] 4
    6. Identify week number (each year has 52 weeks) for each year and product category## with most positive reviews (4 and 5 star) 5
2. Provide detailed analysis of "Digital eBook Purchase" versus Books. 6
    1. Using Spark Pivot functionality, produce DataFrame with following columns: 6
        1.Year 6
        2.Month 6
        3. Total number of reviews for "Digital eBook Purchase" category 6
        4. Total number of reviews for "Books" category 6
        5. Average stars for reviews for "Digital eBook Purchase" category 6
        6. Average stars for reviews for "Books" category 6
    2. Produce two graphs to demonstrate aggregations from #1: 7
        1. Number of reviews 7
        2. Average stars 7
    3. Identify similar products (books) in both categories. Use "product_title" to match products. To account for potential differences in naming of products, compare titles after stripping spaces and converting to lower case. 8
        1. Is there a difference in average rating for the similar books in digital and printed form? 8
        2. To answer #1, you may calculate number of items with high stars in digital form versus printed form, and vice versa. Alternatively, you can make the conclusion by using appropriate pairwise statistic. 9
    4. Using provided LDA starter notebook, perform LDA topic modeling for the reviews in Digital_Ebook_Purchase and Books categories. Consider reviews for the January of 2015 only 9
        1. Perform LDA separately for reviews with 1/2 stars and reviews with 4/5 stars. 9
        2. Add stop words to the standard list as needed. In the example notebook, you can see some words like 34, br, p appear in the topics. 18
        3. Identify 5 top topics for each case (1/2 versus 4/5) 20
        4. Does topic modeling provides good approximation to number of stars given in the review? 21

1. Explore the dataset and provide analysis by product-category and year:

1. Number of reviews

Command

```
gauzed.groupby("year","product_category").agg(F.countDistinct("review_id")
.alias('NoOfReview')).show(5)
```

Output

```
+----+--------------------+----------+
|year|    product_category|NoOfReview|
+----+--------------------+----------+
|2014|               Books|   3540834|
|2010|Digital_Ebook_Pur...|    102514|
|2015|               Books|   2860737|
|2013|            Wireless|   1767127|
|2014|         Mobile_Apps|   1728284|
+----+--------------------+----------+
only showing top 5 rows
```

2. Number of distinct users

Command

```
gauzed.groupby("year","product_category").agg(F.countDistinct("customer_id
").alias('NoOfDistinctUsers')) \
.sort("year",ascending=True).show(10)
```

Output

```
+----+--------------------+-----------------+
|year|    product_category|NoOfDistinctUsers|
+----+--------------------+-----------------+
|2005|               Books|           290585|
|2005|            Wireless|            10584|
|2005|Digital_Ebook_Pur...|               17|
|2005|Digital_Video_Dow...|                6|
|2005|           Video_DVD|            95196|
|2005|                  PC|            15781|
|2006|Digital_Video_Dow...|              154|
|2006|           Video_DVD|           105660|
|2006|                  PC|            23176|
|2006|            Wireless|            17984|
+----+--------------------+-----------------+
only showing top 10 rows
```

3. Average and Median review stars

Command

```
gauzed.groupby("year","product_category").agg(round(F.avg("star_rating"),2
).alias('AvgRating'),
                                                F.expr('percentile_approx(star_ra
ting,0.5)').alias('Median')) \
.sort("year","product_category",ascending=True).show()
```

Output

```
+----+--------------------+---------+------+
|year|    product_category|AvgRating|Median|
+----+--------------------+---------+------+
|2005|               Books|     4.15|     5|
|2005|Digital_Ebook_Pur...|     3.58|     4|
|2005|Digital_Video_Dow...|     3.75|     4|
|2005|                  PC|     3.62|     4|
|2005|           Video_DVD|      4.0|     5|
|2005|            Wireless|     3.41|     4|
|2006|               Books|      4.2|     5|
|2006|Digital_Ebook_Pur...|     4.03|     5|
|2006|Digital_Video_Dow...|     3.63|     4|
|2006|                  PC|     3.72|     4|
|2006|           Video_DVD|     4.08|     5|
|2006|            Wireless|     3.51|     4|
|2007|               Books|     4.26|     5|
|2007|Digital_Ebook_Pur...|     3.94|     5|
|2007|Digital_Video_Dow...|      3.6|     4|
|2007|                  PC|     3.94|     5|
|2007|           Video_DVD|     4.16|     5|
|2007|            Wireless|     3.76|     4|
|2008|               Books|     4.23|     5|
|2008|Digital_Ebook_Pur...|     3.95|     5|
+----+--------------------+---------+------+
only showing top 20 rows
```

4. Percentiles of length of the review. Use the following percentiles: [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]

Command

```
from pyspark.sql.functions import stddev_pop,min,max,length,count,mean
Length1=gauzed.withColumn('length',length(df.review_body))
Length2=Length1.groupby("year","product_category").agg(round(F.avg("length
"),2).alias('AvgOfReviews'))
columnName = "AvgOfReviews"
quantiles = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
Deflection = 0.01
Length2.stat.approxQuantile("AvgOfReviews",quantiles,Deflection)
```

Output

```
[188.95, 349.16, 586.57, 845.33, 961.96, 1170.03]
```

5. Percentiles for number of reviews per product. For example, 10% of books got 5 or less reviews. Use the following percentiles: [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]

Command

```
from pyspark.sql.functions import stddev_pop, min, max,length,count, mean
dataframe=gauzed.groupby("year","product_id","product_category").agg(F.cou
ntDistinct("review_id").alias('NoOfReviews'))
quantiles = [0.1, 0.25, 0.5, 0.75, 0.9, 0.95]
Deflection = 0.01
dataframe.stat.approxQuantile("NoOfReviews",quantiles,Deflection)
```

Output

```
[1.0, 1.0, 1.0, 3.0, 9.0, 18.0]
```

6. Identify week number (each year has 52 weeks) for each year and product category## with most positive reviews (4 and 5 star)

Command

```
from pyspark.sql.functions import *
star4rating = gauzed.star_rating.isin(4)
star5rating = gauzed.star_rating.isin(5)
dfQ6=gauzed.select("product_category","year","review_date") \
.withColumn("week_number",weekofyear("review_date")).where(star4rating | s
tar5rating)
df2Q6 = dfQ6.groupby("product_category","year","week_number").agg(F.countD
istinct("week_number").alias("TotCount"))
df2Q6.drop('TotCount').show()
```

Output

```
+--------------------+----+-----------+
|    product_category|year|week_number|
+--------------------+----+-----------+
|           Video_DVD|2015|         12|
|               Books|2011|         36|
|Digital_Ebook_Pur...|2015|         16|
|           Video_DVD|2011|         37|
|Digital_Ebook_Pur...|2014|         11|
|               Books|2008|         48|
|               Books|2007|         37|
|Digital_Ebook_Pur...|2015|         11|
|         Mobile_Apps|2013|          2|
|Digital_Ebook_Pur...|2013|         49|
|Digital_Ebook_Pur...|2013|         19|
|               Books|2014|          6|
|               Books|2009|         36|
|                  PC|2010|         20|
|               Books|2010|          3|
|           Video_DVD|2009|          9|
|               Books|2010|         12|
|               Books|2006|         12|
|                  PC|2012|         24|
|         Mobile_Apps|2011|         32|
+--------------------+----+-----------+
only showing top 20 rows
```

2. Provide detailed analysis of "Digital eBook Purchase" versus Books.

1. Using Spark Pivot functionality, produce DataFrame with following columns:

1.Year

2.Month

3. Total number of reviews for "Digital eBook Purchase" category

4. Total number of reviews for "Books" category

5. Average stars for reviews for "Digital eBook Purchase" category

6. Average stars for reviews for "Books" category

Command

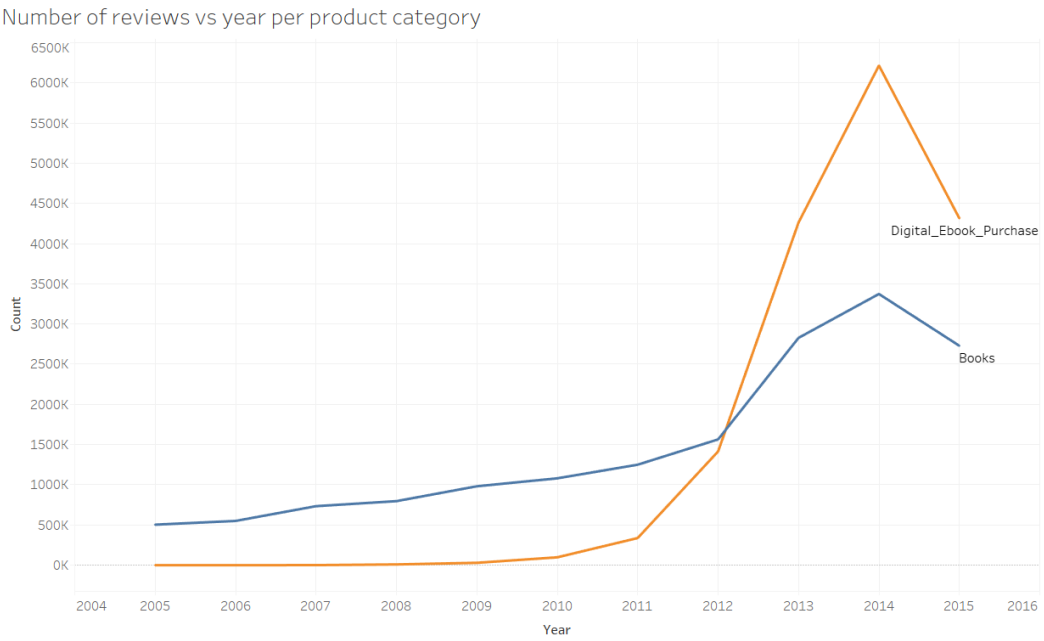
```
tobepivoted=['Digital_Ebook_Purchase','Books']
postpivoting=gauzed.groupBy("year",F.month(F.col("review_date"))).pivot("product_category",tobepivoted)\
 .agg((F.count("review_id")).alias("CountOfReviews"),
 F.round(F.mean("star_rating"),3).alias("AvgRating")).sort("year","month(review_date)",ascending=True).show()
```

Output

```
+----+-----------------+-----------------------------------+-------------------------------+-------------------+----------
-----+
|year|month(review_date)|Digital_Ebook_Purchase_CountOfReviews|Digital_Ebook_Purchase_AvgRating|Books_CountOfReviews|Books_AvgR
ating|
+----+-----------------+-----------------------------------+-------------------------------+-------------------+----------
-----+
|2005|                1|                                  1|                            5.0|              40426|
4.121|
|2005|                2|                               null|                           null|              33726|
4.125|
|2005|                3|                                  2|                            4.5|              38882|
4.122|
|2005|                4|                                  1|                            5.0|              36887|
4.132|
|2005|                5|                                  1|                            1.0|              36873|
4.132|
|2005|                6|                               null|                           null|              36608|
4.115|
|2005|                7|                                  3|                            2.0|              45945|
4.128|
|2005|                8|                                  3|                          2.667|              58926|
4.186|
|2005|                9|                                  2|                            4.0|              58127|
4.203|
|2005|               10|                                  4|                            4.0|              51214|
4.18|
|2005|               11|                                  1|                            5.0|              40885|
4.151|
|2005|               12|                                  1|                            5.0|              42522|
4.126|
|2006|                1|                                  8|                          3.375|              51994|
4.135|
|2006|                2|                                  5|                            4.6|              54413|
4.203|
|2006|                3|                               null|                           null|              66895|
4.233|
|2006|                4|                               null|                           null|              27675|
4.132|
|2006|                5|                                  1|                            5.0|              45007|
4.18|
|2006|                6|                                  5|                            4.2|              48050|
4.184|
|2006|                7|                                  1|                            4.0|              55793|
4.2|
|2006|                8|                                  9|                          4.444|              54418|
4.213|
+----+-----------------+-----------------------------------+-------------------------------+-------------------+----------
-----+
only showing top 20 rows
```

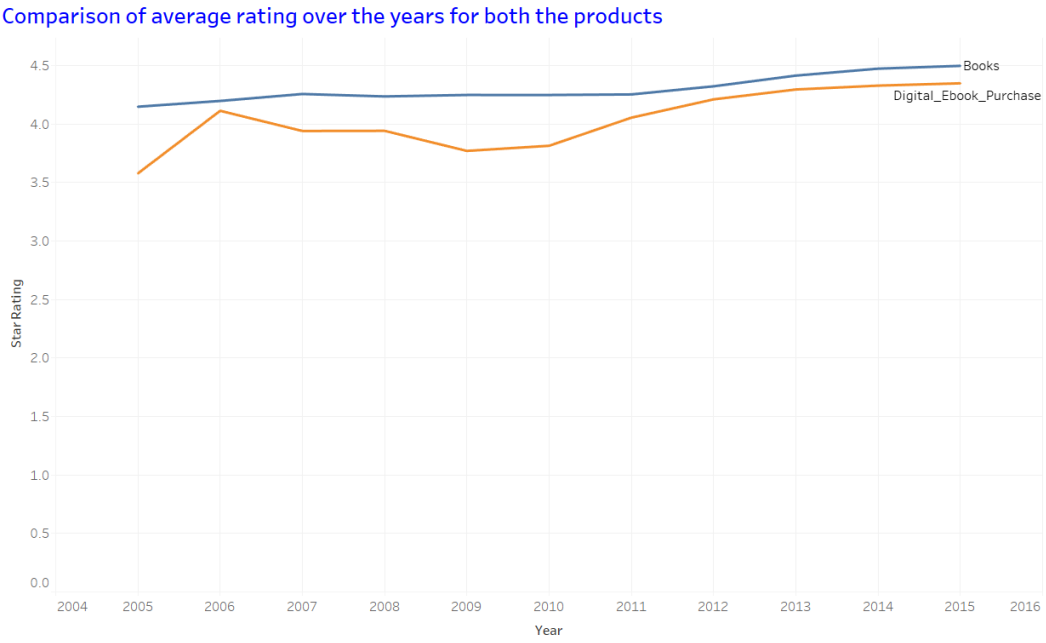
2. Produce two graphs to demonstrate aggregations from #1:

1. Number of reviews



The number of reviews for ebooks has decreased considerably, this explains that people are moving away to paper books

2. Average stars



People clearly tend to prefer paper books as opposed to ebooks.

3. Identify similar products (books) in both categories. Use "product_title" to match products. To account for potential differences in naming of products, compare titles after stripping spaces and converting to lower case.

1. Is there a difference in average rating for the similar books in digital and printed form?

Command

```
product=['Digital_Ebook_Purchase']
DigEbook=gauzed.groupBy("product_title","product_category")\
 .agg((F.count("review_id")).alias("CountOfReviews"),
 F.round(F.mean("star_rating"),3).alias("AvgRating")).filter(F.col("produc
t_category").isin(product))
trimDigEbook=DigEbook.select(F.lower(F.trim(F.col("product_title"))).alias
("Title"),F.col("CountOfReviews") \
                        ,F.col("AvgRating"))
book1=['Books']
book2=gauzed.groupBy("product_title","product_category")\
 .agg((F.count("review_id")).alias("CntReviewsBook"),
 F.round(F.mean("star_rating"),3).alias("AvgRatingBook")).filter(F.col("pr
oduct_category").isin(book1))
trimEbook=book2.select(F.lower(F.trim(F.col("product_title"))).alias("Book
Title"),F.col("CntReviewsBook") \
                        ,F.col("AvgRatingBook"))
joinExpression = trimEbook["BookTitle"] == trimDigEbook["Title"]
joinType = "inner"
final=trimEbook.join(trimDigEbook, joinExpression, joinType)
final.show()
```

Output

```
+--------------------+-------------+-------------+--------------------+-------------+---------+
|           BookTitle|CntReviewsBook|AvgRatingBook|               Title|CountOfReviews|AvgRating|
+--------------------+-------------+-------------+--------------------+-------------+---------+
|"rays of light": ...|            2|          5.0|"rays of light": ...|            1|      5.0|
|"the siege of khe...|           19|        4.316|"the siege of khe...|          156|    3.327|
|          'dem bon'z|            4|          5.0|          'dem bon'z|            2|      5.0|
|   0400 roswell time|            1|          5.0|   0400 roswell time|            6|    3.667|
|10 smart things g...|            1|          5.0|10 smart things g...|            6|    4.833|
|10 smart things g...|           19|        4.789|10 smart things g...|            6|    4.833|
|100 prayers for y...|           11|          5.0|100 prayers for y...|            7|      5.0|
|13 cent killers: ...|           37|        2.811|13 cent killers: ...|           15|    3.933|
|25 essentials: te...|           41|        4.439|25 essentials: te...|            1|      5.0|
|30 before 30: tra...|            2|          3.5|30 before 30: tra...|           33|     4.97|
|300 hard word sea...|            2|          4.5|300 hard word sea...|            7|      1.0|
|42 rules to incre...|            1|          5.0|42 rules to incre...|            2|      5.0|
|50 american heroe...|            2|          5.0|50 american heroe...|            3|      4.0|
|50 successful har...|           49|        4.347|50 successful har...|            3|    4.667|
|52 prepper projec...|           30|          3.9|52 prepper projec...|            2|      4.5|
|73 north: the bat...|            6|          5.0|73 north: the bat...|            1|      5.0|
|<i>change</i> the...|            8|         4.75|<i>change</i> the...|            2|      4.5|
|      a changed life|            5|          4.2|      a changed life|           36|    4.222|
|a chip off the ol...|            1|          5.0|a chip off the ol...|            1|      5.0|
|a closer look at ...|            1|          5.0|a closer look at ...|            1|      1.0|
+--------------------+-------------+-------------+--------------------+-------------+---------+
only showing top 20 rows
```

2. To answer #1, you may calculate number of items with high stars in digital form versus printed form, and vice versa. Alternatively, you can make the conclusion by using appropriate pairwise statistic.

Command

```
ratingforprintedbook=F.col("AvgRatingBook")>4
final.where(ratingforprintedbook).count()
```

Output

```
276590
```

Command

```
ratingforebook=F.col("AvgRating")>4
final.where(ratingforebook).count()
```

Output

```
245526
```

We can see that printed book has got more number of higher rating i.e count of more than 4 star ratings is higher for printed books as compared to digital book star ratings.

4. Using provided LDA starter notebook, perform LDA topic modeling for the reviews in Digital_Ebook_Purchase and Books categories. Consider reviews for the January of 2015 only

1. Perform LDA separately for reviews with 1/2 stars and reviews with 4/5 stars.

Command for topic modelling for 4/5 stars

```
from pyspark.mllib.clustering import LDA, LDAModel
from pyspark.mllib.linalg import Vectors
from pyspark.ml.feature import CountVectorizer, IDF,RegexTokenizer, Tokenizer
from pyspark.sql.types import ArrayType
from pyspark.sql.types import StringType
from pyspark.sql.types import *
from pyspark.sql.functions import udf
from pyspark.sql.functions import struct
import re
from pyspark.ml.feature import StopWordsRemover
from pyspark.ml.clustering import LDA
from pyspark.ml.feature import CountVectorizer
```

In [39]:

```
df_ml = gauzed.filter((F.col("product_category")=="Digital_Ebook_Purchase") |
(F.col("product_category")=="Books") \
                    & (F.col("year")==2015) \
                    & (F.col("review_date")<'2015-02-01')
                    & (F.col("star_rating")>3))
```

In [40]:

```
df1 = df_ml.withColumn('review_text',
                       F.concat(F.col('review_headline'),F.lit(' '),
F.col('review_body')))
corpus =df1.select('review_text')

# This will return a new DF with all the columns + id
corpus_df = corpus.withColumn("id", F.monotonically_increasing_id())
# Remove records with no review text
corpus_df = corpus_df.dropna()

corpus_df.persist()
print('Corpus size:', corpus_df.count())
corpus_df.show(5)

corpus_df.printSchema()
```

Output

```
Corpus size: 18287530
+--------------------+---+
|         review_text| id|
+--------------------+---+
|Nice Story but ve...|  0|
|Beautiful and hea...|  1|
|Worth The Wait. T...|  2|
|written before. I...|  3|
|Entertaining Rev....|  4|
+--------------------+---+
only showing top 5 rows

root
 |-- review_text: string (nullable = true)
 |-- id: long (nullable = false)
```

Code for tokenizing

```
tokenizer = Tokenizer(inputCol="review_text", outputCol="words")
countTokens = udf(lambda words: len(words), IntegerType())
'''
tokenized_df = tokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words").withColumn("tokens", countTokens(col("words"))).show()
'''
regexTokenizer = RegexTokenizer(inputCol="review_text",
                                outputCol="words",pattern="\\w+", gaps=False)
# alternatively, pattern="\\w+", gaps(False) pattern="\\W"

tokenized_df = regexTokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words") \
    .withColumn("tokens", countTokens(F.col("words"))).show()
```

Output

```
+--------------------+--------------------+------+
|         review_text|               words|tokens|
+--------------------+--------------------+------+
|Nice Story but ve...|[nice, story, but...|    40|
|Beautiful and hea...|[beautiful, and, ...|    76|
|Worth The Wait. T...|[worth, the, wait...|    77|
|written before. I...|[written, before,...|   327|
|Entertaining Rev....|[entertaining, re...|    51|
|Fastest 600 page ...|[fastest, 600, pa...|    45|
|Amazing It is a c...|[amazing, it, is,...|    27|
|Huge impact Profo...|[huge, impact, pr...|    27|
|LOVED LOVED LOVED...|[loved, loved, lo...|    25|
|Five Stars very h...|[five, stars, ver...|     4|
|This is an awesom...|[this, is, an, aw...|    26|
|Kept me intereste...|[kept, me, intere...|    29|
|So many of these ...|[so, many, of, th...|    50|
|she is an incredi...|[she, is, an, inc...|    43|
|Thoroughly enjoye...|[thoroughly, enjo...|    42|
|This book has mad...|[this, book, has,...|    39|
|Not as good as th...|[not, as, good, a...|    33|
|Writer's Block Wo...|[writer, s, block...|    74|
|One of my favorit...|[one, of, my, fav...|    43|
|Wow This book was...|[wow, this, book,...|    74|
+--------------------+--------------------+------+
only showing top 20 rows
```

Code for LDA for 4/5 stars

```
#k=10 means 10 words per topic
lda = LDA(k=10, maxIter=10)
model = lda.fit(countVectors)
```

In [47]:

```
topics = model.describeTopics(5)
topics_rdd = topics.rdd

topics_words = topics_rdd\
       .map(lambda row: row['termIndices'])\
       .map(lambda idx_list: [vocab[idx] for idx in idx_list])\
       .collect()

for idx, topic in enumerate(topics_words):
    print ("topic: ", idx)
    print ("----------")
    for word in topic:
       print (word)
    print ("----------")
```

Output

```
topic:  0
----------
story
characters
love
read
good
----------
topic:  1
----------
good
read
story
great
really
----------
topic:  2
----------
read
series
books
great
love
----------
topic:  3
----------
story
life
read
love
world
----------
topic:  4
----------
read
story
good
mystery
great
----------
topic:  5
----------
read
like
great
time
interesting
----------
topic:  6
----------
read
characters
reading
story
great
----------
topic:  7
----------
great
read
life
good
information
----------
topic:  8
----------
love
story
like
really
read
----------
topic:  9
----------
read
author
good
books
enjoyed
----------
```

Topic modelling for 1/2 stars

Command

```
df_ml1 = gauzed.filter((F.col("product_category")=="Digital_Ebook_Purchase")
| (F.col("product_category")=="Books") \
                    & (F.col("year")==2015) \
                    & (F.col("review_date")<'2015-02-01')
                    & (F.col("star_rating")<3))
```

In [50]:

```
df1 = df_ml1.withColumn('review_text',
                        F.concat(F.col('review_headline'),F.lit(' '), F.col('r
eview_body')))
corpus =df1.select('review_text')

# This will return a new DF with all the columns + id
corpus_df = corpus.withColumn("id", F.monotonically_increasing_id())
# Remove records with no review text
corpus_df = corpus_df.dropna()

corpus_df.persist()
print('Corpus size:', corpus_df.count())
corpus_df.show(5)

corpus_df.printSchema()
```

Output

```
Corpus size: 17950045
+--------------------+---+
|         review_text| id|
+--------------------+---+
|Nice Story but ve...|  0|
|Beautiful and hea...|  1|
|Worth The Wait. T...|  2|
|written before. I...|  3|
|Entertaining Rev....|  4|
+--------------------+---+
only showing top 5 rows

root
 |-- review_text: string (nullable = true)
 |-- id: long (nullable = false)
```

Tokenizing for 1/2 stars

Command

```
tokenizer = Tokenizer(inputCol="review_text", outputCol="words")
countTokens = udf(lambda words: len(words), IntegerType())
'''
tokenized_df = tokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words").withColumn("tokens", countTokens(col("words"))).show()
'''
regexTokenizer = RegexTokenizer(inputCol="review_text",
                                outputCol="words",pattern="\\w+", gaps=False)
# alternatively, pattern="\\w+", gaps(False) pattern="\\W"

tokenized_df = regexTokenizer.transform(corpus_df)
tokenized_df.select("review_text", "words") \
    .withColumn("tokens", countTokens(F.col("words"))).show()
```

Output

```
+--------------------+--------------------+------+
|         review_text|               words|tokens|
+--------------------+--------------------+------+
|Nice Story but ve...|[nice, story, but...|    40|
|Beautiful and hea...|[beautiful, and, ...|    76|
|Worth The Wait. T...|[worth, the, wait...|    77|
|written before. I...|[written, before,...|   327|
|Entertaining Rev....|[entertaining, re...|    51|
|Fastest 600 page ...|[fastest, 600, pa...|    45|
|Amazing It is a c...|[amazing, it, is,...|    27|
|Huge impact Profo...|[huge, impact, pr...|    27|
|LOVED LOVED LOVED...|[loved, loved, lo...|    25|
|Five Stars very h...|[five, stars, ver...|     4|
|This is an awesom...|[this, is, an, aw...|    26|
|Kept me intereste...|[kept, me, intere...|    29|
|So many of these ...|[so, many, of, th...|    50|
|she is an incredi...|[she, is, an, inc...|    43|
|Thoroughly enjoye...|[thoroughly, enjo...|    42|
|Not as good as th...|[not, as, good, a...|    33|
|Writer's Block Wo...|[writer, s, block...|    74|
|One of my favorit...|[one, of, my, fav...|    43|
|Wow This book was...|[wow, this, book,...|    74|
|THE BEST OF THE B...|[the, best, of, t...|    86|
+--------------------+--------------------+------+
only showing top 20 rows
```

Command for removing stop words

```
remover = StopWordsRemover(inputCol="words", outputCol="filtered")
tokenized_df1 = remover.transform(tokenized_df)
tokenized_df1.show(5)

stopwordList = stop_words

remover=StopWordsRemover(inputCol="filtered", outputCol="filtered_more" ,s
topWords=stopwordList)
tokenized_df2 = remover.transform(tokenized_df1)
tokenized_df2.show(5)
```

Output

```
+--------------------+---+--------------------+--------------------+
|         review_text| id|               words|            filtered|
+--------------------+---+--------------------+--------------------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|
+--------------------+---+--------------------+--------------------+
only showing top 5 rows

+--------------------+---+--------------------+--------------------+--------------------+
|         review_text| id|               words|            filtered|       filtered_more|
+--------------------+---+--------------------+--------------------+--------------------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|[nice, story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|[beautiful, heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|[worth, wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|[written, really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|[entertaining, re...|
+--------------------+---+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Command for counting total records in the DF

```
cv = CountVectorizer(inputCol="filtered_more", outputCol="features", vocab
Size = 10000)
cvmodel = cv.fit(tokenized_df2)
featurized_df = cvmodel.transform(tokenized_df2)
vocab = cvmodel.vocabulary
featurized_df.select('filtered_more','features','id').show(5)

countVectors = featurized_df.select('features','id')
countVectors.persist()
print('Records in the DF:', countVectors.count())
```

```
countVectors = featurized_df.select('features','id')
countVectors.persist()
print('Records in the DF:', countVectors.count())
```

Output

```
+--------------------+--------------------+---+
|       filtered_more|            features| id|
+--------------------+--------------------+---+
|[nice, story, rus...|(10000,[0,1,4,5,7...|  0|
|[beautiful, heart...|(10000,[1,5,9,12,...|  1|
|[worth, wait, sto...|(10000,[1,27,56,8...|  2|
|[written, really,...|(10000,[0,4,5,9,1...|  3|
|[entertaining, re...|(10000,[0,22,37,4...|  4|
+--------------------+--------------------+---+
only showing top 5 rows

Records in the DF: 17950045
```

Performing LDA for 1/2 stars

Command

```
lda = LDA(k=10, maxIter=5)
model = lda.fit(countVectors)
```

In [56]:

```
topics = model.describeTopics()
topics_rdd = topics.rdd

topics_words = topics_rdd\
       .map(lambda row: row['termIndices'])\
       .map(lambda idx_list: [vocab[idx] for idx in idx_list])\
       .collect()

for idx, topic in enumerate(topics_words):
    print ("topic: ", idx)
    print ("----------")
    for word in topic:
       print (word)
    print ("----------")
```

Output

```
topic:  0
----------
story
good
love
read
characters
series
author
time
world
great
----------
topic:  1
----------
good
read
story
great
stars
really
like
love
series
characters
----------
topic:  2
----------
read
series
books
like
great
love
reading
story
loved
wait
----------
topic:  3
----------
story
read
love
characters
written
like
great
novel
way
author
----------
topic:  4
----------
read
good
like
great
books
reading
easy
new
story
author
----------
topic:  5
----------
read
great
time
like
life
history
reading
good
people
know
----------
topic:  6
----------
read
love
loved
series
characters
story
reading
great
wait
books
----------
topic:  7
----------
great
read
reading
story
series
stars
recommend
forward
life
good
----------
topic:  8
----------
story
love
really
life
like
read
characters
loved
stars
know
----------
topic:  9
----------
read
good
characters
author
story
enjoyed
books
like
great
reading
----------
```

2. Add stop words to the standard list as needed. In the example notebook, you can see some words like 34, br, p appear in the topics.

Command

```
stop_words = ['a', 'about', 'above', 'across', 'after', 'afterwards', 'aga
in', 'against', 'all', 'almost', 'alone', 'along', 'already', 'also', 'alt
hough', 'always', 'am', 'among', 'amongst', 'amoungst', 'amount', 'an', 'a
nd', 'another', 'any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere
', 'are', 'around', 'as', 'at', 'back', 'be', 'became', 'because', 'become
', 'becomes', 'becoming', 'been', 'before', 'beforehand', 'behind', 'being
', 'below', 'beside', 'besides', 'between', 'beyond', 'bill', 'both', 'bot
tom', 'but', 'by', 'call', 'can', 'cannot', 'cant', 'co', 'computer', 'con
', 'could', 'couldnt', 'cry', 'de', 'describe', 'detail', 'do', 'done', 'd
own', 'due', 'during', 'each', 'eg', 'eight', 'either', 'eleven', 'else',
'elsewhere', 'empty', 'enough', 'etc', 'even', 'ever', 'every', 'everyone'
, 'everything', 'everywhere', 'except', 'few', 'fifteen', 'fify', 'fill',
'find', 'fire', 'first', 'five', 'for', 'former', 'formerly', 'forty', 'fo
und', 'four', 'from', 'front', 'full', 'further', 'get', 'give', 'go', 'ha
d', 'has', 'hasnt', 'have', 'he', 'hence', 'her', 'here', 'hereafter', 'he
reby', 'herein', 'hereupon', 'hers', 'herself', 'him', 'himself', 'his', '
how', 'however', 'hundred', 'i', 'ie', 'if', 'in', 'inc', 'indeed', 'inter
est', 'into', 'is', 'it', 'its', 'itself', 'keep', 'last', 'latter', 'latt
erly', 'least', 'less', 'ltd', 'made', 'many', 'may', 'me', 'meanwhile', '
might', 'mill', 'mine', 'more', 'moreover', 'most', 'mostly', 'move', 'muc
h', 'must', 'my', 'myself', 'name', 'namely', 'neither', 'never', 'neverth
eless', 'next', 'nine', 'no', 'nobody', 'none', 'noone', 'nor', 'not', 'no
thing', 'now', 'nowhere', 'of', 'off', 'often', 'on', 'once', 'one', 'only
', 'onto', 'or', 'other', 'others', 'otherwise', 'our', 'ours', 'ourselves
', 'out', 'over', 'own', 'part', 'per', 'perhaps', 'please', 'put', 'rathe
r', 're', 'same', 'see', 'seem', 'seemed', 'seeming', 'seems', 'serious',
'several', 'she', 'should', 'show', 'side', 'since', 'sincere', 'six', 'si
xty', 'so', 'some', 'somehow', 'someone', 'something', 'sometime', 'someti
mes', 'somewhere', 'still', 'such', 'system', 'take', 'ten', 'than', 'that
', 'the', 'their', 'them', 'themselves', 'then', 'thence', 'there', 'there
after', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they', '
thick', 'thin', 'third', 'this', 'those', 'though', 'three', 'through', 't
hroughout', 'thru', 'thus', 'to', 'together', 'too', 'top', 'toward', 'tow
ards', 'twelve', 'twenty', 'two', 'un', 'under', 'until', 'up', 'upon', 'u
s', 'very', 'via', 'was', 'we', 'well', 'were', 'what', 'whatever', 'when'
, 'whence', 'whenever', 'where', 'whereafter', 'whereas', 'whereby', 'wher
ein', 'whereupon', 'wherever', 'whether', 'which', 'while', 'whither', 'wh
o', 'whoever', 'whole', 'whom', 'whose', 'why', 'will', 'with', 'within',
'without', 'would', 'yet', 'you', 'your', 'yours', 'yourself', 'yourselves
', '']
stop_words = stop_words + ['br','book','34','y','m','ich','zu']
```

```
remover = StopWordsRemover(inputCol="words", outputCol="filtered")
tokenized_df1 = remover.transform(tokenized_df)
tokenized_df1.show(5)

stopwordList = stop_words
```

```
remover=StopWordsRemover(inputCol="filtered", outputCol="filtered_more" ,s
topWords=stopwordList)
tokenized_df2 = remover.transform(tokenized_df1)
tokenized_df2.show(5)
```

Output

```
+--------------------+---+--------------------+--------------------+
|         review_text| id|               words|            filtered|
+--------------------+---+--------------------+--------------------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|
+--------------------+---+--------------------+--------------------+
only showing top 5 rows

+--------------------+---+--------------------+--------------------+--------------------+
|         review_text| id|               words|            filtered|       filtered_more|
+--------------------+---+--------------------+--------------------+--------------------+
|Nice Story but ve...|  0|[nice, story, but...|[nice, story, rus...|[nice, story, rus...|
|Beautiful and hea...|  1|[beautiful, and, ...|[beautiful, heart...|[beautiful, heart...|
|Worth The Wait. T...|  2|[worth, the, wait...|[worth, wait, sto...|[worth, wait, sto...|
|written before. I...|  3|[written, before,...|[written, really,...|[written, really,...|
|Entertaining Rev....|  4|[entertaining, re...|[entertaining, re...|[entertaining, re...|
+--------------------+---+--------------------+--------------------+--------------------+
only showing top 5 rows
```

```
cv = CountVectorizer(inputCol="filtered_more", outputCol="features", vocab
Size = 10000)
cvmodel = cv.fit(tokenized_df2)
featurized_df = cvmodel.transform(tokenized_df2)
vocab = cvmodel.vocabulary
featurized_df.select('filtered_more','features','id').show(5)

countVectors = featurized_df.select('features','id')
countVectors.persist()
print('Records in the DF:', countVectors.count())
```

Output

```
+--------------------+--------------------+---+
|       filtered_more|            features| id|
+--------------------+--------------------+---+
|[nice, story, rus...|(10000,[0,1,4,5,7...|  0|
|[beautiful, heart...|(10000,[1,5,9,12,...|  1|
|[worth, wait, sto...|(10000,[1,28,57,8...|  2|
|[written, really,...|(10000,[0,4,5,9,1...|  3|
|[entertaining, re...|(10000,[0,22,37,4...|  4|
+--------------------+--------------------+---+
only showing top 5 rows

Records in the DF: 18287530
```

3. Identify 5 top topics for each case (1/2 versus 4/5)

Topics for case with 1/2 stars

topic: 0	topic: 3	topic: 6	
story	story	read	
good	read	love	
love	love	loved	
read	characters	series	
characters	written	characters	
series	like	story	
author	great	reading	
time	novel	great	
world	way	wait	
great	author	books	
topic: 1	**topic: 4**	**topic: 7**	
good	read	great	
read	good	read	
story	like	reading	
great	great	story	
stars	books	series	
really	reading	stars	
like	easy	recommend	
love	new	forward	
series	story	life	
characters	author	good	
topic: 2	**topic: 5**	**topic: 8**	**topic: 9**
read	read	story	read
series	great	love	good
books	time	really	characters
like	like	life	author
great	life	like	story
love	history	read	enjoyed
reading	reading	characters	books
story	good	loved	like
loved	people	stars	great
wait	know	know	reading

Topics for case with 4/5 stars

```
topic:  0
----------
story
characters
love
read
good
----------
topic:  1
----------
good
read
story
great
really
----------
topic:  2
----------
read
series
books
great
love
----------
topic:  3
----------
story
life
read
love
world
----------
topic:  4
----------
read
story
good
mystery
great
----------
topic:  5
----------
read
like
great
time
interesting
----------
topic:  6
----------
read
characters
reading
story
great
----------
topic:  7
----------
great
read
life
good
information
----------
topic:  8
----------
love
story
like
really
read
----------
topic:  9
----------
read
author
good
books
enjoyed
----------
```

4. Does topic modeling provides good approximation to number of stars given in the review?

Answer

We can clearly make out after performing LDA that there are some positive words in reviews which have lower than 3 rating, as a result in this case topic modelling may not provide a good approximation to the number of stars given in a review.