Frequent words identification in the CNN and The Daily Mail news article using pySpark FPGrowth algorithm

Aleksandra Łepek

1 Introduction

1.1 Problem statement

The purpose of this project is to identify the most frequently appearing words in the text data coming from CNN and Daily Mail articles and articles highlight. This task can pose challenges due to following factors: number of articles present in dataset and number of words per article .
Additionally the performance of the pySpark FPGrowth algorithm was compared to less robust Pandas solution.

1.2 Data description

The dataset used for this project consists of collection of 300,000 news articles divided into article text itself and short highlight. Articles were collected from CNN and The Daily Mail in the period between April 2007 - April 2015 and June 2010 and April 2015 respectively. The data included in the project is publicly available through Kaggle website.

1.3 Data preprocessing

The dataset was loaded into pySpark dataframe that logically corresponds to dataframe present in Pandas.

```
+--------------------+--------------------+--------------------+
|                  id|             article|          highlights|
+--------------------+--------------------+--------------------+
|0001d1afc246a7964...|By . Associated P...|Bishop John Folda...|
|He contracted the...|                null|                null|
|Church members in...| Grand Forks and ...|                null|
|0002095e55fcbd3a2...|"(CNN) -- Ralph M...|"" of using his r...|
|          Ralph Mata| an internal affa...| allegedly helped...|
+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Next all the rows with null value in any of the columns were removed, along with the special characters, hyperlinks, double spaces and numbers.

```
+--------------------+--------------------+--------------------+
|                  id|             article|                text|
+--------------------+--------------------+--------------------+
|0001d1afc246a7964...|By . Associated P...|Bishop John Folda...|
|0002095e55fcbd3a2...|"(CNN) -- Ralph M...|"" of using his r...|
|          Ralph Mata| an internal affa...| allegedly helped...|
|00027e965c8264c35...|A drunk driver wh...|Craig Eccleston T...|
|0002c17436637c4fe...|(CNN) -- With a b...|Nina dos Santos s...|
|0003ad6ef0c37534f...|Fleetwood are the...|Fleetwood top of ...|
|        Peterborough|        Bristol City| Chesterfield and...|
|0004306354494f090...|He's been accused...|Prime Minister an...|
|0005d61497d21ff37...|By . Daily Mail R...|NBA star calls fo...|
|0006021f772fad0aa...|"By . Daily Mail ...| other passengers...|
+--------------------+--------------------+--------------------+
only showing top 10 rows
```

To prepare data into the tokens and remove any words that are not bring meaning into the sentence, StopWordsRemover and Tokenizer feature from pySpark machine learning library were used and applied.

```
+--------------------+--------------------+--------------------+--------------------+
|                  id|                text|              tokens|             cleaned|
+--------------------+--------------------+--------------------+--------------------+
|0001d1afc246a7964...|Bishop John Folda...|[bishop, john, fo...|[bishop, john, fo...|
|0002095e55fcbd3a2...|"" of using his r...|[of, using, his, ...|[using, role, pol...|
|          Ralph Mata| allegedly helped...|[allegedly, helpe...|[allegedly, helpe...|
|00027e965c8264c35...|Craig Eccleston T...|[craig, eccleston...|[craig, eccleston...|
|0002c17436637c4fe...|Nina dos Santos s...|[nina, dos, santo...|[nina, dos, santo...|
+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Obtained array with the tokenized words was again cleared to make sure that there are no duplicates inside each of the article basket.

```
+----------------------------------------------------------------------------------------------------+
|array_distinct(cleaned)                                                                             |
+----------------------------------------------------------------------------------------------------+
|[bishop, john, folda, north, dakota, taking, time, diagnosed]                                       |
|[using, role, police, officer, help, drug, trafficking, organization, exchange, money, gifts]       |
|[allegedly, helped, group, get, guns]                                                               |
|[craig, eccleston, todd, drunk, least, three, pints, driving, car]                                  |
|[nina, dos, santos, says, europe, must, ready, accept, sanctions, hurt, sides]                      |
+----------------------------------------------------------------------------------------------------+
only showing top 5 rows
```

This process was applied for both types of the text data: highlights and article text. The average number of tokens inside of the highlights was equal to almost 8 with standard

deviation of 6. The longest highlight had 222 tokens.
The average number of tokens inside of the article was equal to 149 with standard deviation of 128. The longest highlight had 614 tokens.

2 Methodology

2.1 Classic Apriori Algorithm

Apriori algorithm is a classical algorithm of association rule mining. Two algorithms Apriori and AprioriTid were introduced and implemented by R. Agarwal et al.[1] to discover all significant association rules for transaction datasets. However this classical algorithm is inefficient due to so many scans of database [2]. Later, many improvements were done to make Apriori better, efficient and faster.

2.2 FP Growth

The FP-Growth algorithm is another algorithm that is used to discover frequent itemsets in large databases. It is an alternative to the Apriori algorithm and is more efficient and faster, particularly for larger datasets. The algorithm works by constructing a tree structure, called the FP-Tree. Items that are below given minimum support will not be added to the tree. Then algorithm recursively searches the tree to identify frequent itemsets. The FP-Growth algorithm has been shown to be more efficient than Apriori for large datasets, particularly when the minimum support threshold is low.

3 Results

3.1 FPGrowth: Highlights text with support of 0.05, 0.005 and 0.0005

Different thresholds for support were used to compare the performance of the FP Growth algorithm over the 382,518 different highlights. The performance of the algorithm was satisfactory for min support of 0.05 and 0.005 with 7.62 seconds and 10.89 seconds of execution respectively. However setting up minimum support of 0.0005 required 109.59 seconds of execution reaching almost a runtime limit in Google Colab.

0.05

items	freq
[said]	29765

0.005

items	freq
[today]	2720
[minister]	1978
[news]	2335
[go]	2540
[park]	2158
[car]	3067
[family]	4126
[sunday]	3578
[told]	8415
[west]	2879

only showing top 10 rows

0.0005

items	freq
[announce]	246
[insurance]	290
[singer]	639
[grand]	896
[trade]	331
[wounds]	216
[today]	2720
[today, said]	295
[isn]	383
[defender]	498

only showing top 10 rows

Figure 1: Frequent items identification based on minimum support

3.2 FPGrowth: Article text with support of 0.05 and different volumes: from 10 articles to 10,000 articles

The same thresholds for minimum support of 0.05 was used to compare the performance of the FP Growth algorithm over the different volume of the data: 10, 100, 1000, 5000, 7500 and 10000 article texts.

Table 1: Comparison of the execution times based on different article volume.

Execution time in seconds				
Volume	10	100	1,000	10,000
Time	1.61	0.76	2.43	28.61

The performance of the algorithm was satisfactory for min support of 0.05 and volume up to 10,000 articles. From the time data we can observe that the time to perform the algorithm was growing exponentially soon reaching the timeout error while trying to process 50,000 articles.

100 articles

items	freq
[need]	5
[need, like]	5
[good]	12
[good, next]	7
[good, next, first]	5
[good, next, firs...	5
[good, next, year]	6
[good, next, year...	6
[good, next, made]	5
[good, next, made...	5

only showing top 10 rows

1,000 articles

items	freq
[want]	127
[want, right]	56
[want, right, said]	50
[want, first]	56
[want, know]	50
[want, told]	52
[want, back]	53
[want, year]	82
[want, year, one]	67
[want, year, one,...	60

only showing top 10 rows

10,000 articles

items	freq
[says]	1308
[says, made]	506
[says, new]	644
[says, new, said]	549
[says, people]	647
[says, people, said]	553
[says, year]	847
[says, year, one]	626
[says, year, one,...	545
[says, year, said]	729

only showing top 10 rows

Figure 2: Frequent Items identified in different articles volume

The association rules were also listed, however with increasing volume of the articles the output table was taking more time to compute.

100 articles

antecedent	consequent	confidence	lift	support
[think, people, y...	[four]	1.0	5.88235294117647	0.05
[published, home,...	[said]	1.0	1.923076923076923	0.05
[published, home,...	[time]	1.0	2.7777777777777777	0.05
[day, last, two, ...	[court]	1.0	11.11111111111111	0.05
[day, last, two, ...	[said]	1.0	1.923076923076923	0.05
[told, years, two...	[day]	1.0	3.846153846153846	0.05
[told, years, two...	[time]	1.0	2.7777777777777777	0.05
[told, years, two...	[year]	1.0	2.5	0.05
[told, years, two...	[said]	1.0	1.923076923076923	0.05
[published, updat...	[est]	1.0	7.692307692307692	0.06
[2013, est, publi...	[home]	1.0	4.166666666666667	0.05
[2013, est, publi...	[year]	1.0	2.5	0.05
[2013, est, publi...	[said]	1.0	1.923076923076923	0.05
[2013, est, publi...	[years]	1.0	3.2258064516129035	0.05
[want, 000, take,...	[first]	1.0	3.4482758620689657	0.05
[want, 000, take,...	[many]	1.0	6.25	0.05
[taken, two, also]	[home]	1.0	4.166666666666667	0.05
[taken, two, also]	[day]	1.0	3.846153846153846	0.05
[taken, two, also]	[year]	1.0	2.5	0.05
[taken, two, also]	[000]	1.0	6.25	0.05

only showing top 20 rows

1,000 articles

antecedent	consequent	confidence	lift	support
[est, published, ...	[updated]	1.0	5.780346820809249	0.055
[2013, est, updat...	[published]	1.0	5.46448087431694	0.07
[home, three, tol...	[said]	1.0	1.8083182640144664	0.05
[later, told, old...	[year]	1.0	2.347417840375587	0.053
[2013, est, publi...	[updated]	1.0	5.780346820809249	0.054
[est, published, ...	[updated]	1.0	5.780346820809249	0.052
[right, left, tol...	[said]	1.0	1.8083182640144664	0.054
[work, added]	[said]	1.0	1.8083182640144664	0.052
[est, published, ...	[updated]	1.0	5.780346820809249	0.053
[spokesman, year]	[said]	1.0	1.8083182640144664	0.058
[est, left, year,...	[updated]	1.0	5.780346820809249	0.063
[est, published, ...	[updated]	1.0	5.780346820809249	0.055
[updated, publish...	[est]	1.0	6.211180124223603	0.064
[years, old, two,...	[year]	1.0	2.347417840375587	0.051
[est, updated, fo...	[published]	1.0	5.46448087431694	0.05
[est, home, year]	[updated]	1.0	5.780346820809249	0.054
[updated, publish...	[est]	1.0	6.211180124223603	0.055
[three, old, time...	[year]	1.0	2.347417840375587	0.059
[three, told, old...	[year]	1.0	2.347417840375587	0.053
[est, new]	[updated]	1.0	5.780346820809249	0.064

only showing top 20 rows

10,000 articles

antecedent	consequent	confidence	lift	support
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0599
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0526
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0546
[10, est, published]	[updated]	1.0	5.31632110579479	0.0582
[est, published, ...]	[updated]	1.0	5.31632110579479	0.058
[mr, est, publish...]	[updated]	1.0	5.31632110579479	0.0539
[2013, updated, p...]	[est]	1.0	5.74712643678161	0.0544
[2013, est, last]	[updated]	1.0	5.31632110579479	0.0521
[family, est, pub...]	[updated]	1.0	5.31632110579479	0.0509
[2013, updated, p...]	[est]	1.0	5.74712643678161	0.0543
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0531
[est, published, ...]	[updated]	1.0	5.31632110579479	0.051
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0505
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0696
[2013, updated, p...]	[est]	1.0	5.74712643678161	0.054
[est, published, ...]	[updated]	1.0	5.31632110579479	0.0512
[2013, updated, p...]	[est]	1.0	5.74712643678161	0.0517
[life, est]	[updated]	1.0	5.31632110579479	0.0526
[family, est, pub...]	[updated]	1.0	5.31632110579479	0.0556
[2013, updated, p...]	[est]	1.0	5.74712643678161	0.06

only showing top 20 rows

Association

Rules in different articles volume

3.3 Apriori Algorithm: Highlights text

To verify the performance of pySpark FP Growth model, standard Pandas library Efficient Apriori algorithm was applied over the highlights text. Data preprocessing was identical for both dataframes.

Table 2: Comparison of the execution times based on different algorithm.

	FP Growth		Efficient Apriori	
Support	0.05	0.005	0.05	0.005
Time	7.62	10.8	0.29	119.02

When setting the Efficient Apriori model parameter of minimum support of 0.0005 we were receiving timeout error in the google Colab.

```
start = timeit.default_timer()
itemsets, rules = apriori(transactions, min_support=0.0005, min_confidence=1)
print(rules)
highlights_pandas_005 = timeit.default_timer() - start
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

3.4 Conclusions and observations

The pySpark FP Growth algorithm shows superior performance over standard Pandas Efficient Apriori. For the minimum support of 0.005 execution time in pySpark was around 10 seconds and only slightly increased from the threshold of 0.05. For the minimum support of 0.05 Efficient Apriori performed better, however the for lower threshold the performance became quickly really poor.

Interestingly both models Google Colab poses restrictions in terms of available resources and low threshold for query execution time, which limited the scope of this project to 'smaller' datasets. On the local machine Spark setup can be adjusted based on the dataset volume and parameters, which enables longer processing times and would allow to process the whole volume of the articles text.

4 Disclaimer

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

References

[1] Agrawal, A., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. (1996). Fast Discovery of Association Rules. In: Fayyad et al. (1996), 307–328

[2] Improving Efficiency of Apriori Algorithm Using Transaction Reduction International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013

5 Resources

1. https://spark.apache.org/docs/latest/ml-frequent-pattern-mining.html
2. https://efficient-apriori.readthedocs.io/en/latest/

6 Project details

1. Dataset: https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail
2. Github for the project: https://github.com/olalepek/PySpark$_{C}NN_{A}rticle_{F}requent_{I}tems/blob/main/A$