Finding frequent itemsets (Market Basket Analysis) in tweets based on the current conflict between Russia and Ukraine

UNIVERSITÀ
DEGLI STUDI
DI MILANO

Data Science and Economics

Algorithms For Massive Data

Enki Muca

964653

June 2022

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/our 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.

Introduction

This paper consits in implementing a market basket analysis on the tweets about Ukraine-Russia conflict during the period from 1st of June 2022 to 3rd of June 2022.
Market basket analysis is a data mining technique used by retailers to increase sales by better understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase history, to reveal product groupings, as well as products that are likely to be purchased together.
Let's put it in the context of our project now. Just like the grocery items in a supermarket, we now consider the content as baskets and each single word as an item.
The dataset was given beforehand in the project requirements, it linked to a kaggle repository, which included a very big list of tweets about the conflict between Ukraine and Russia. Of course the amount of data was substantial so I decided to choose the tweets from 1st of June to 3rd of June as mentioned also above.
Each day has its own csv file with the same structure. Dataset has 17 features in total however we will use only 3 of them for this analysis. In particular, our dataset will have the following features:

- userid : a unique identifier assigned to each user.
- text : the textual content of the tweet.
- language : the language of the text

Pre-Processing

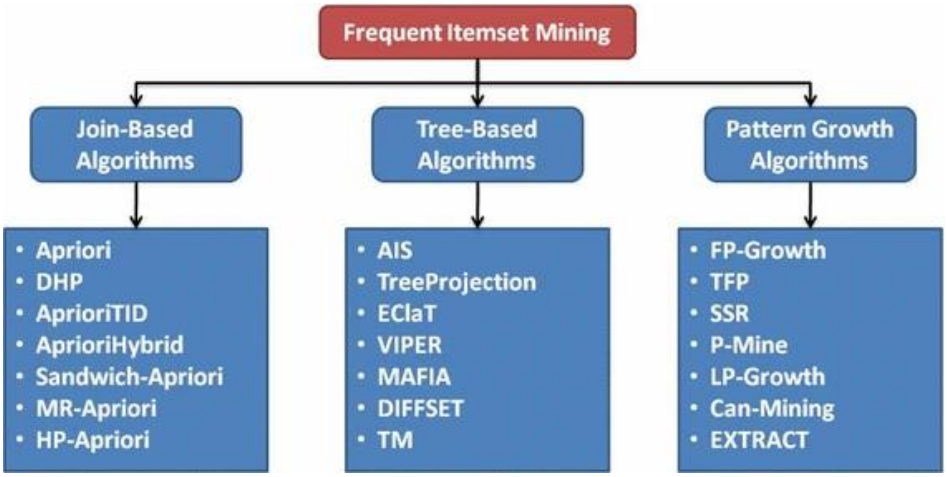
Once again need to mention the fact that every word in every tweet will represent an item just like milk or oranges in the supermarket. Basically, in this analysis, words are our oil so we need to take care of them. First thing to do is to remove dublicates. Another step in this process which is more optional is to choose only tweets and words in english. This is not a must, but just something done for the sake of the project and the fact that during the interpretation stage, interpreting english words would be easier for my part. Stripping the tweets is necessary to remove unnecessary spaces from each tweet, which leaves the necessary contents. As we said, each single word in these tweets is like oil for us, and it should be refined before "feeding" it to the model. The tweets are compressed of multiple words and of course the authors used them in sentences, but in order to use our algorithm we need words and not sentences. First of all, as mentioned during the lecturers, each basket needs to contain only unique items (words). We also need to clear white spaces, and any kind of elements that are part of a sentence, but play no beneficiary role in our work. At the end, we will have in our hand some nice baskets, each of them containing unique items and tokenized (no more sentences, but words).

	userid	cleantext
0	1525285584	still not a single battlefront tweet from n...
1	1132795518	watch ka52 attack helicopters ludicrous stunt...
2	3537816269	the president of the united states talking ...
3	764546052	another heartbreaking story a 5year old maryna...
4	296005306	did everything to avert global food crisiseve...
...	...	...
218461	15226381	germany would be more than happy with an eve...
218462	1284332827	has confirmed that it is helping ukraine with...
218463	631816995	an extremely rare video of ukrainian s300pt s...
218464	1529523042300354567	this is how the help of the western world save...
218465	1305865621880569856	russian strategy of scorched earth and indiscr...

218466 rows × 2 columns

Clean data on content only related to the war

The goal of Frequent Itemset Mining is to identify often occurring product combinations with a fast and efficient algorithm. There are different algorithms for this. One of the foundational algorithms is the Apriori algorithm.The FP Growth algorithm can be seen as Apriori's modern version, as it is faster and more efficient while obtaining the same goal.



Just for simple pragmatic reasons, knowing that the FP-Growth is much better, I decided to just use the one and exclude the Apriori algorithm from my analysis. The idea behind the FP Growth algorithm is to find the frequent itemsets in a dataset while being faster than the Apriori algorithm. The Apriori algorithm basically goes back and forth to the data set to check for the co-occurrence of products in the data set. In order to be faster, the FP algorithm changed the organization of the data into a tree rather than sets. This tree data structure allows for faster scanning, and this is where the algorithm wins time, in fact, FP-tree requires only one scan of the database in its beginning steps so it consumes less time. In the FP-Growth algorithm, the algorithm represents the data in a tree structure. It is a lexicographic tree structure that we call the FP-tree. Which is responsible for maintaining the association information between the frequent items. After making the FP-Tree, it is segregated into the set of conditional FP-Trees for every frequent item. A set of conditional FP-Trees further can be mined and measured separately. For example, the database is similar to the dataset we used in the apriori algorithm.

1. **Counting the occurrences of individual items** → The first step of the FP Growth algorithm is to count the occurrences of individual items.
2. **Filter out non-frequent items using minimum support** → You need to decide on a value for the minimum support**:** every item or item set with fewer occurrences than the minimum support will be excluded.

3. **Order the itemsets based on individual occurrences** → For the remaining items, we will create an ordered table. This table will contain the items that have not been rejected yet, and the items will be ordered based on individual product occurrence.
4. **Create the tree and add the transactions one by one** → Now, we can create the tree starting with the first basket.

FP Growth algorithm implementation chosen for this work is the pyspark implementation. After trying many different thresholds, we decided to use a minimum support of 0.04 since lower values resulted in irrelevant items while the higher values returned a few items.

- *Support* is a measure that indicates the frequent appearance of a variable set or itemset in a database
- *Confidence* is a measure that indicates how often a rule appears to be true.
- *Lift* is the ratio between target response and average response, in other words the ratio between confidence and the expected confidence.

The results show that the word Russian holds the highest count at 13032 words. Ukraine follows with 8429 and so forth. This provides a general understanding of how each word appears in the tweets. Further analysis might reveal the reason for the frequent appearance of these words.

	items	incident_count
0	russian	13032
1	ukraine	8429
2	war	7615
3	amp	7384
4	ukrainian	5760

Words with highest frequency of usage (top 5)

The association rules count the frequency of items that occur together, seeking to find associations that occur far more often than expected. This helps us uncover meaningful correlations between different products according to their co-occurrence in the data set. Need to say that most of the results of these computations are really common sense since

the discussed topic leaves little place to flexibility. Displaying the association rules shows an overview of the different rules. The antecedents and consequents show the content of each rule. Different metrics are also available to study the association rules. They include support for each section of the rules, general support, confidence, lift, leverage, and conviction. These metrics are necessary for further investigation of the rules.

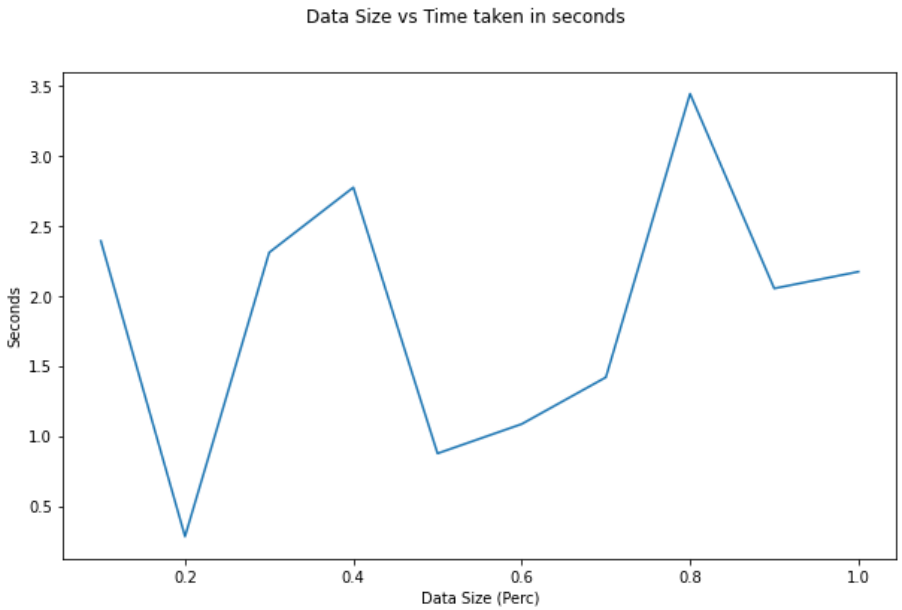
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(russian)	(ukrainian)	0.21714	0.11014	0.04946	0.227779	2.068089	0.025544	1.152339
1	(ukrainian)	(russian)	0.11014	0.21714	0.04946	0.449065	2.068089	0.025544	1.420966
2	(russian)	(war)	0.21714	0.13414	0.03000	0.138160	1.029967	0.000873	1.004664
3	(war)	(russian)	0.13414	0.21714	0.03000	0.223647	1.029967	0.000873	1.008381
4	(region)	(russian)	0.07310	0.21714	0.03028	0.414227	1.907650	0.014407	1.336456
...	...	...	...	...	...	...	...	...	...
131	(russian, forced)	(donetsk)	0.02852	0.03044	0.02724	0.955119	31.377110	0.026372	21.603009
132	(donetsk, forced)	(russian)	0.02724	0.21714	0.02724	1.000000	4.605324	0.021325	inf
133	(russian)	(donetsk, forced)	0.21714	0.02724	0.02724	0.125449	4.605324	0.021325	1.112297
134	(donetsk)	(russian, forced)	0.03044	0.02852	0.02724	0.894875	31.377110	0.026372	9.241204
135	(forced)	(russian, donetsk)	0.03700	0.02760	0.02724	0.736216	26.674501	0.026219	3.686352

136 rows × 9 columns

A look at the rules sorted by the support yet shows a different appearance of the rules. The typical Ukrain and Russian keywords dominate the rules by support. However, the appearance of the word 'Kherson' shows that there might have happened some activities in the area around this time. The word 'Captured also appears at the fifth position, which can interpret as some individuals captured during the war at this time. Another interesting appearance is the word 'kids', an indicator of an action that might have related to children.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(russian)	(ukrainian)	0.21714	0.11014	0.04946	0.227779	2.068089	0.025544	1.152339
1	(ukrainian)	(russian)	0.11014	0.21714	0.04946	0.449065	2.068089	0.025544	1.420966
6	(russian)	(kherson)	0.21714	0.05272	0.04674	0.215253	4.082944	0.035292	1.207115
7	(kherson)	(russian)	0.05272	0.21714	0.04674	0.886571	4.082944	0.035292	6.901736
119	(captured)	(russian)	0.05274	0.21714	0.04668	0.885097	4.076157	0.035228	6.813207
118	(russian)	(captured)	0.21714	0.05274	0.04668	0.214977	4.076157	0.035228	1.206665
17	(kids)	(russian)	0.05206	0.21714	0.04570	0.877833	4.042706	0.034396	6.408128
16	(russian)	(kids)	0.21714	0.05206	0.04570	0.210463	4.042706	0.034396	1.200628
108	(forces)	(russian)	0.07522	0.21714	0.03388	0.450412	2.074294	0.017547	1.424449
109	(russian)	(forces)	0.21714	0.07522	0.03388	0.156028	2.074294	0.017547	1.095748

The last part of this projects concerns answering the question regarding the scalability of the algorithm. In order to do that, we check the performance of the model using different fractions of the data we are using. As we can see, it is very understanding that as we grow the part of the data we are using, the algorithm needs more time to proccess all of it. However, we can say with confidence that when it comes to the question weather FP-Growth is scalable or not, we can say YES.



Conclusion

Using FP-Growth algorithm to perform market basket analysis should be considered a very reliable method and pobably one of the best. This is of course not just about the result as many other algorithms can produce these results, but mostly in the sense of time consumption and scalability. In real-life analysis, computational resources are as important as the output so an algorithm to be considered good must have other aspects besides from the performance regarding the result. Also the scalability is a must have as dataset can get bigger and bigger as more information comes in and if the model stops working after a certain point, that is considered as a huge disadvantage.