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

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.

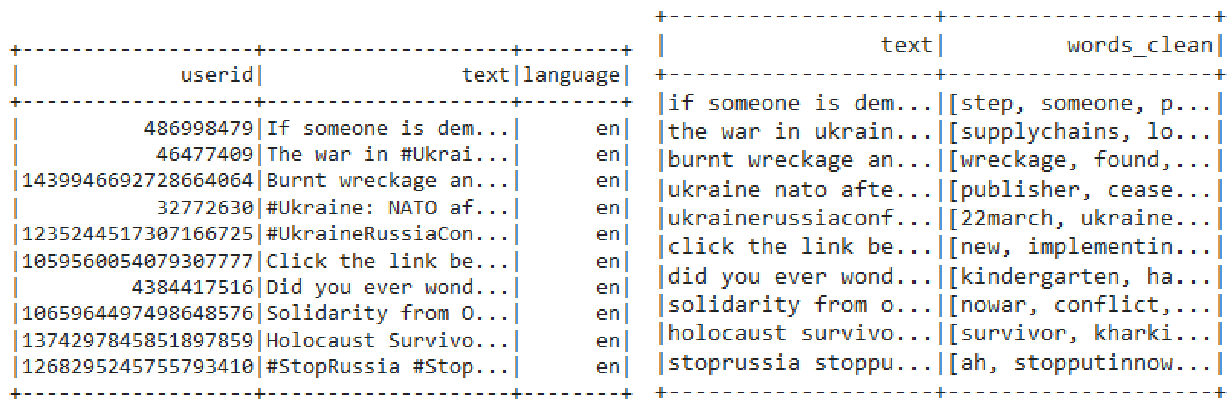
Introduction

This paper consits in implementing a market basket analysis on the tweets about Ukraine-Russia conflict during the period from 25th of March 2022 to 30th of March 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 from the very first day until the 31st of March, also including some extras. Of course the amount of data was substantial so I decided to choose the tweets from 25th to 30th as mentioned also above.
Each day from 25th of March to 30th of March 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. After all is said and done, we are left with 433'073 tweets.
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).

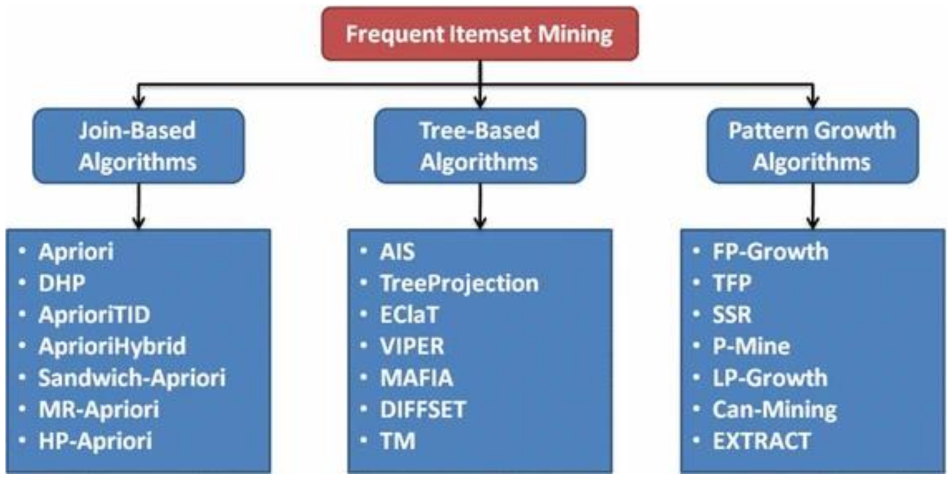


```
+-------------------+--------------------+--------+
|             userid|                text|language|
+-------------------+--------------------+--------+
|          486998479|If someone is dem...|      en|
|           46477409|The war in #Ukrai...|      en|
|1439946692728664064|Burnt wreckage an...|      en|
|           32772630|#Ukraine: NATO af...|      en|
|1235244517307166725|#UkraineRussiaCon...|      en|
|1059560054079307777|Click the link be...|      en|
|         4384417516|Did you ever wond...|      en|
|1065964497498648576|Solidarity from O...|      en|
|1374297845851897859|Holocaust Survivo...|      en|
|1268295245755793410|#StopRussia #Stop...|      en|
+-------------------+--------------------+--------+
```

```
+--------------------+--------------------+
|                text|         words_clean|
+--------------------+--------------------+
|if someone is dem...|[step, someone, p...|
|the war in ukrain...|[supplychains, lo...|
|burnt wreckage an...|[wreckage, found,...|
|ukraine nato afte...|[publisher, cease...|
|ukrainerussiaconf...|[22march, ukraine...|
|click the link be...|[new, implementin...|
|did you ever wond...|[kindergarten, ha...|
|solidarity from o...|[nowar, conflict,...|
|holocaust survivo...|[survivor, kharki...|
|stoprussia stoppu...|[ah, stopputinnow...|
+--------------------+--------------------+
```

Tokenized sentences from the tweets

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.



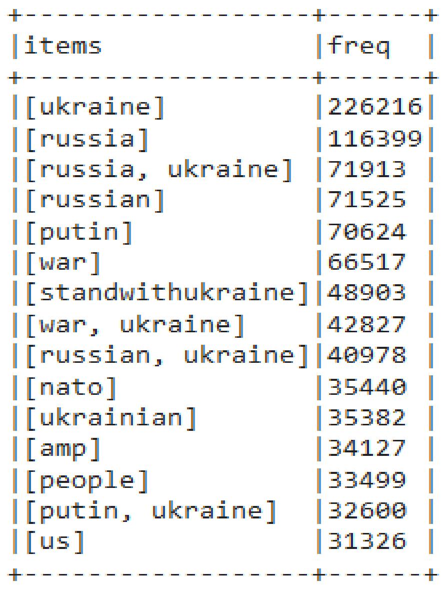
Just for simple pragmatic reasons, knowing that the FP-Growth is much better, I decided to just use the one and exclude the Apriori algorithm from my analysis. The idea behind the FP Growth algorithm is to find the frequent itemsets in a dataset while being faster than the Apriori algorithm. The Apriori 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.

This is the list of the mostly used words and and most of it is pretty intuitive. Ukraine is the most used word since the war takes place there and all the following are very similar. What got my attention was the missing of the Ukraine president, since he has been a very powerful mediatic person. I guess the reason for that is the days we are taking into consideration. Probably, if we analysed data closer to the beginning of the war, his name would appear in the top 15 list.



```
+------------------+------+
|items             |freq  |
+------------------+------+
|[ukraine]         |226216|
|[russia]          |116399|
|[russia, ukraine] |71913 |
|[russian]         |71525 |
|[putin]           |70624 |
|[war]             |66517 |
|[standwithukraine]|48903 |
|[war, ukraine]    |42827 |
|[russian, ukraine]|40978 |
|[nato]            |35440 |
|[ukrainian]       |35382 |
|[amp]             |34127 |
|[people]          |33499 |
|[putin, ukraine]  |32600 |
|[us]              |31326 |
+------------------+------+
```

Words with highest frequency of usage (top 15)

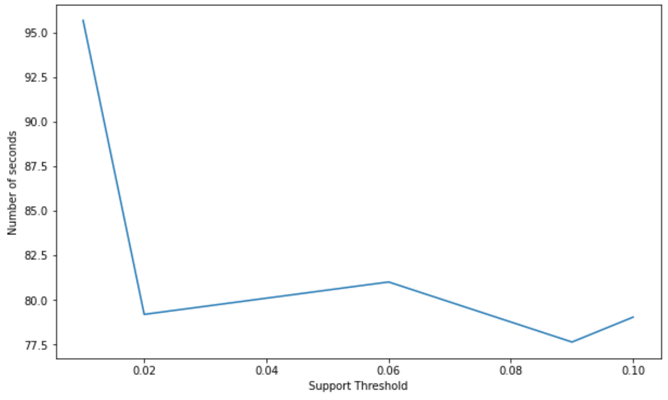
The association rules count the frequency of items that occur together, seeking to find associations that occur far more often than expected. This helps us uncover meaningful correlations between different products according to their co-occurrence in the data set. Need to say that most of the results of these computations are really common sense since the discussed topic leaves little place to flexibility. Of course that the consequent item for most of the words will be Ukraine as that’s the main issue.

```
+-----------------+----------+------------------+------------------+--------------------+
|antecedent       |consequent|confidence        |lift              |support             |
+-----------------+----------+------------------+------------------+--------------------+
|[close]          |[ukraine] |0.9196186987470341|1.4700341526219256|0.04318512798960054 |
|[close]          |[sky]     |0.8522665778628814|18.847905581204607|0.04002228455816954 |
|[humanitarian]   |[ukraine] |0.9286279335501451|1.4844356462709867|0.04130462404582018 |
|[stopputin]      |[ukraine] |0.8529093879602866|1.3633976028344963|0.06633174669885547 |
|[potus]          |[ukraine] |0.838149012680588 |1.3398027631512266|0.051940613607263986|
|[nato, russia]   |[ukraine] |0.837910816232063 |1.3394219999992354|0.04246185724199271 |
|[stopputin, stop]|[ukraine] |0.9507371007371007|1.5197777189095   |0.04538426201948922 |
|[stop]           |[ukraine] |0.8125150105892884|1.2988261512207226|0.07274539892291303 |
|[weapons]        |[ukraine] |0.8653136531365314|1.383226139892413 |0.04125575440071154 |
|[sky]            |[close]   |0.8860518457610248|18.84711486389766 |0.04002228455816954 |
|[sky]            |[ukraine] |0.9179902194140304|1.4674309865057775|0.04146491648177651 |
+-----------------+----------+------------------+------------------+--------------------+
```

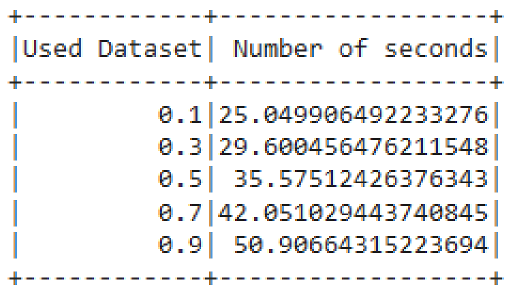
In simple words, support says that 4% of the whole tweets have the word “close” and “ukraine”. Confidence says that 91% of the users who wrote “close”, associated it with the word “ukraine”.

There is always a trade-off between the amount of information we receive and the computational resources we use. As we know, in order to assuma that a set of items is frequent we need to formalize it with a support threshold. What it means is that we say that a set of items is frequent if the support is at lest as the support threshold or more. The higher the threshold, the more items we consider as frequent and more information we get, but lowering the threshold we let more information flow into the boat and ergo more weight to be kept. Below there is the relationship between the quantity of the threshold and the number of seconds for the model to compute according to it.

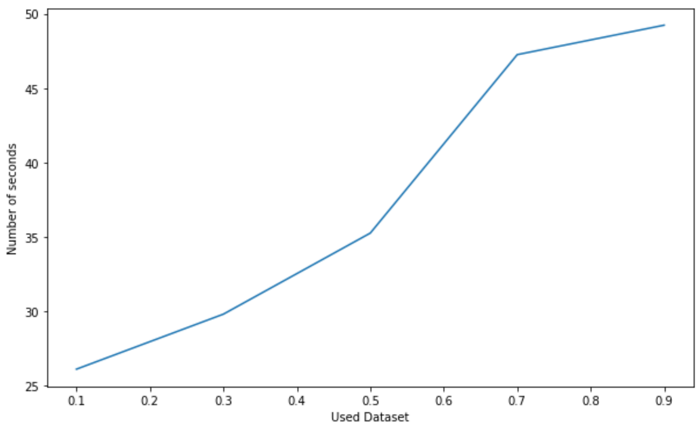
```
+-----------------+-----------------+
|Support Threshold|Number of Seconds|
+-----------------+-----------------+
|             0.01| 95.6715042591095|
|             0.02|79.18079376220703|
|             0.06|80.99832105636597|
|             0.09|77.63020896911621|
|              0.1|79.02651166915894|
+-----------------+-----------------+
```



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.



```
+------------+------------------+
|Used Dataset| Number of seconds|
+------------+------------------+
|         0.1|25.049906492233276|
|         0.3|29.600456476211548|
|         0.5| 35.57512426376343|
|         0.7|42.051029443740845|
|         0.9| 50.90664315223694|
+------------+------------------+
```



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.