Data Mining Project Report

Final Report

Elia Cunegatti
MSc Computer Science 1st Year
University of Trento
Trento, Italy
Department of Information Engineering and Computer Science (DISI)
elia.cunegatti@studenti.unitn.it

ABSTRACT

Data mining processes have become always more used throughout the last decade. Their point of strength is that they can be useful for a lot of various purposes. A lot of data, that belong to different branches, are usually analyzed with some kind of Data Mining process in order to understand them better. In this project, we are going to demonstrate how a Data Mining process can be applied to the analysis of tweets.

1 INTRODUCTION

Nowadays, social media's power it's always more impressive all over the world. Every day, billions of people use some kind of social network to interact with others, sharing photos, reading information, and writing their opinion about any type of topic. The most powerful social media used to share opinions is Twitter[1].

Let's try to understand better what Twitter is with the definition available in the dictionary: "Twitter, online microblogging service for distributing short messages among groups of recipients via personal computer or mobile telephone. Twitter incorporates aspects of social networking Web sites, such as Myspace and Facebook, with instant messaging technologies to create networks of users who can communicate throughout the day with brief messages, or tweets"[2]. So it's clear that it is an online platform where people share opinions and can communicate among them. Now we can move on by focusing on the main subject of the whole project, the Tweet. Let's always go to see some dictionary's definitions:

- "a short remark or piece of information published on Twitter";
- "a message put on Twitter to let people know what you are doing, thinking, feeling, etc..";
- "to communicate on Twitter using quick short messages".

The definitions are directly taken from Cambridge Dictionary[3].

It's also important to specify how a tweet appears: it's composed of a maximum of 290 characters in which can be included emojis, links, hashtags, words, and the chance to mention other persons using @username.

An amazing number of tweets are published every day from any part of the world. Overall the number of information we have collected is a big amount of data and would be very interesting analyzing them afterwards, in order to try to understand what is going on all over the world. This is exactly the topic of this project: take a dataset containing a large number of tweets, and analyzing them.

[1]https://twitter.com
[2][3, Humanitas. 2020. Twitter. Encyclopædia Britannica. https://www.britannica.com/topic/Twitter]
[3]https://dictionary.cambridge.org

In detail, the focus of this work is to explain how to discover some words related to each other, i.e topics, that appear frequently in tweets during some periods of time. In other words; popular consistent topics. In fact, some words become very popular only when they appear in a couple. Hence, we are not interested in finding the biggest topics on Twitter, instead, we want to discover some smaller topics not so popular constantly over time, but very popular at certain timespan.

2 MOTIVATION

Every day a lot of thoughts are shared on Twitter, from different places and from various people with different ideas about the topic and, each of them shares tweets only about something of its interest. However in our mobile-application, we can't see everything that has been published, we only see what is published by the people we are following and what Twitter suggest to us. Finally, we will never have entire knowledge about what is shared on social media.

Therefore, as we can imagine, it's impossible to understand what is going on in the world scrolling the own Twitter page, either because we are following only some persons/pages, hence we can't see all information inside the social media, or because the data are a lot and it's tough to make a summary in our head. But thanks to the technology we can take all the data available on Twitter and processing them afterwards. In this way, we will provide a better system to visualize and understand the information shared on Twitter. Through a Data Mining process, we can develop some algorithms in order to take the data and extract only the information we need.

With this project, we are going to take a large dataset contained about 1.6 million tweets, shared during 2020 and, we are going to process them to have a better way to understand the content. This is the main topic: it doesn't care what we have in our hands if we can't understand it. It's fundamental to understand it and to visualize it correctly. However, as just said before, we are curious about some constant topics that are a lot popular only in some periods in time. The aim is to be able to provide everyone the idea of what happening rarely but repeatedly in a large timespan because from the moment this does not happen always can be easily forgotten.

The fact mentioned above is one of the biggest problems of our times. Since we are living in a world always faster, we don't pay attention to details, but we focus only on the biggest trend. Climate change is the most typical example of it: only a few people are really interested in the topic, but when a disaster happens everybody speaks/shares about that, but always only for a small period of time i.e. is a popular consistent topic. Through this project, we want exactly to pull out some common consistent topics in order to be able to better understand what is going on, without being blinded by trend topics.

3 RELATED WORK

The methods used to extract the constant topics over time, from the dataset, are the frequent itemsets and the association rules. The frequent itemsets provide a way to find the pairs of words that are present several times throughout the dataset. The times that a pair of words is present is computed by a function called *support*. The support is the fraction of the times the pair of words appear in tweets among all available tweets.

$$SUPPORT(A,B) = \frac{number\ of\ tweets\ in\ which\ A, B\ appears}{total\ number\ of\ tweets};$$

$$range:\ [0,1]$$

In order to keep only the frequent itemsets that appear many times, a threshold k is setting and will be kept only the frequent itemsets with *support* greater than k.

After had selected a threshold and had computed the frequent itemsets with *support* greater than k, these frequent itemsets will be used to compute the association rules. Association rules are "if-then" statements, that help to show the probability of relationships between words. It has been used to find the interesting one's relationship between words.

It is going to be computed the metrics called *confidence* and the *lift* (also known as interest)[4].

The confidence represents how likely word the B is present when word A is also present.

$$CONFIDENCE(A \longrightarrow B) = \frac{SUPPORT(A \longrightarrow B)}{SUPPORT(A)},\ range:\ [0,1]$$

Unlikely not all high-*confidence* rules are interesting, perhaps because one of the two words is very popular. For this reason, it's needed to compute the *lift* which described how likely word B is present when also word A is present, while controlling for how popular the word B is.

$$LIFT(A \longrightarrow B) = \frac{SUPPORT(A,B)}{SUPPORT(A) * SUPPORT(B)},\ range:\ [0,\infty]$$

or

$$LIFT(A \longrightarrow B) = \frac{CONFIDENCE(A \longrightarrow B)}{SUPPORT(B)},\ range:\ [0,\infty]$$

A *lift* value greater than 1 means that word B is likely to be found if word A is present, while a value less than 1 means that word B unlikely to be found if word A is also present and, a *lift* equals 1 means that the words are independent. Also here will be used some thresholds on these metrics, one for the *confidence* and another one for the *lift*. In this way, it is going to be kept only the interesting one's items (i.e pairs of words).

4 PROBLEM STATEMENT

We are given a dataset that contained a huge number of tweets. The only fields we are interested in from the dataset are the *"date"* and the *"text"*. Each tweet present in the dataset is composed by the *date* which is when it has published and the *text* i.e the content of the tweet. So in the end, the input dataset will be a matrix having as rows the various tweets and for columns, the field *"date"* and *"text"* related to that row's tweet. So given a dataset containing these fields (if there are some other fields they will not be taken into consideration) we expect to have as output a set of consistent topics over time, i.e. a set of correlated words that appear frequently together but sporadically in a large timespan.

[4]Interest and lift are the same metrics with different name and approach but the main idea is the same. In this project will be used the lift and the notion of it since the algorithm that will be explain later only include lift metric instead of interest.

So, now we can go down to detail and, seeing what is our input, how it will be processed, and what we expect as output.

Input:

$$D = Dataset$$
$$T = Set\ of\ texts$$
$$DD = Set\ of\ Dates$$

$$text = set\ of\ k\ words$$
$$text = \{word_1, word_2, ..., word_k\}$$
$$date = yyyy - mm - dd$$
$$Dates = \{date_1, data_2, ..., date_j\}\ \forall i\ \exists j\ date_i < date_{i+1}$$

$$Tweet = \{date, text\}$$

$$text \subseteq T$$
$$date \subseteq Dates \subseteq DD$$
$$Tweet \subseteq D$$

$$Tweet_i = \{date_i, text_i\}\ \ \forall i\ \ s.t\ \ 1 \le i \le |n|\ \ (\ddagger)$$

$$D = \begin{bmatrix} date & text \\ date_1 & text_1 \\ date_2 & text_2 \\ .... & .... \\ .... & .... \\ date_n & text_n \end{bmatrix}$$

Process:

$$Processing\ Function\ \mathbf{p} : D \longrightarrow Output$$

Output:

$$Consistent\ topic = \{word1, word2\}$$
$$x = number\ of\ consistent\ topics\ \ (*)$$
$$Output = x\ consistent\ topics\ = \ list\ of\ correlated\ words$$
$$Output = \{\{word1_1, word2_1\}, ..., \{word1_x, word2_x\}\}$$

5 SOLUTION

The solution developed to produce as output the consistent topics over time is a combination of what has been explained in the related work section. Remind that the dataset is composed of 1.6 million tweets and a tweet is composed of a *"data"* field and a *"text"* field. The *"text"* field, how it has have been explained, it's a vector of words. In order to discover some pattern, the tweets refer to a specific day, i.e the tweets that have the same *"data"* field, have been joined together into a vector of tweets' text. In this way, we obtained that to each day correspond a set of tweets that have been published on that day.

(‡) n = |D|
(*) x cannot be selected directly by the user, the program itself will output the x consistent topics.

Thanks to this each day can be computed individually. The total number of days is 123 and it's for this reason we have decided to group tweets day by day. But this will better explain later in the evaluation section.

After having done these groupings, the program will compute the frequent itemsets and the association rules. Each group of tweets, i.e array of tweets' text corresponding to a specific day, has been processed separately according to the day the results have been merging afterwards.

Now we can go into detail with the solution.

5.1 Frequent Itemsets

For each day, one by one has been applied, to the vector of tweets correspond to that day, the frequent itemsets algorithm in order to discover the most common itemsets(pairs of words).

Since the tweet has been grouped by day it is going to be calculated the frequent itemsets related to the tweets of the day taken into consideration. Since we are interested in the correlation between words the output of the frequent itemsets has been filtered and has kept only the items with exactly two words. The frequent itemsets algorithm has been done using the support threshold set to 0.003, so a combination of words it's popular if appears at least 3 times over 1000 tweets. The motivation of this choice will be explained in the evaluation section. After having computed the frequent itemsets and have filtered the output of that, all the item remained have been appended to a list.

The list is updated day by day, so at the end, it will be the concatenation of all frequent itemsets for each day.

5.2 Association Rules

For each day, after had computed the frequent itemsets and had filtered them, has been computed the association rules of the frequent itemsets output. This step is fundamental to extract the only itemsets with an interesting relationship between the words that composed them. In fact, the threshold has been set to 0.5 for the confidence and to 10 for the lift.

The association_rules for each frequent itemset$(word_1, word_1)$ compute the rules in both directions, so $(A \longrightarrow B)$ and $(B \longrightarrow A)$. We filter them and if an item appears in both forms we kept it only once. After that, the items present in the association rules' output have been added to another list. Even this list is updated day by day, so at the end, it will be the concatenation of all output items of the association rules for each day.

5.3 Checking and Selection

The two processes described above work in sequence day by day, and they stop after have scanned all possible days. Hence, in the end, after having processed the frequent itemsets and the association rules for each day we obtain two lists.

The first one contained all the frequent itemsets, so there will be a lot of items that are present more than once if they have been found frequent on different days.

The second one contained also the frequent itemsets found each day, but only which have been save after have computed the association rules of the frequent itemset for that specific day.

Keeping these two list separated while we computed the frequent itemset and the association rules it's fundamental. The reason it's because the first list contains all the frequent itemsets found, in the second only the frequent itemsets that we consider interesting. Having two different lists afford to keep a counter for how many times a topic is frequent and in parallel searching only the interesting ones. In this way, after having run the whole process, we can group the list one by items, counts how many time they appear. In other words, count how many times an item (a couple of words) is considered frequent throughout the days available in the dataset.

This can be done by the intersection of the two lists, in this way only the items that appear in both of them are present. Each element has as attributes the *frequency counts* and the *interesting counts*. Thus, is possible to know the difference between been frequent and been interested in any items. From this intersection we will keep only the items and their *frequency counts*. Through this step, we obtain a list of interesting items linked with the number of times they are actually frequent.

This step is essential since a term can be frequent multiple times but can be selected by the association rules fewer times. This is because the association rules only depend on the frequent itemset of that day, so the values will change from a day to another. Doing this checking-step between the two lists afford us to be sure that each interesting item is associated with the correct times that it is frequent.

The last process is to decide how many items give as output and which. The number of the items that are given as output is not constrained given by the user but it's something the program decides itself. The program will output only the items which are popular at least 15% of the days and at most 75% of the days. This choice has been done in order to avoid the consistent topics that are too much unpopular since can be randomness, and also the topics that are too popular since it's not the purpose of the project.

Algorithm 1: Algorithm Solution

```
Input  : Dataset with N tweets
Output : K Consistent Topic
         $(word1_1, word2_1), ....., (word1_k, word2_k)$
1  $list\_day, list\_frequent, list\_rules, results = []$
2  foreach $day$ do
3      $list\_day \leftarrow tweets\ of\ the\ day$
4  end
5  foreach $j$ in $list\_day$ do
6      frequent_itemsets(j)
7      $list\_frequent \leftarrow frequent\_itemsets(j)$
8      association_rules(frequent_itemsets(j))
9      $list\_rules \leftarrow$
         $association\_rules(frequent\_itemsets(j))$
10 end
11 $list\_frequent \leftarrow groupby.counts(list\_frequent)$
12 $list\_rules \leftarrow groupby.count(list\_rules)$
13 $final\_list = list\_rules \cap list_frequent$
14 foreach $i$ in $final\_list$ do
15     if $totalnumberofdays * 15 < i.counts <$
         $totalnumberofdays * 75$
16      then $results \leftarrow i.itemset$ ;
17 end
18 return $results$
```

6 IMPLEMENTATION

The solution has been developed using the programming language Python[5]. For handling the dataset has been used Pandas[6]. It is a software library written for the Python programming language for data manipulation and analysis of big datasets. In fact, Pandas provides an in-memory 2d table object called Dataframe, which has been used both to read the input dataset and to store temporary files (as the two lists described in the solution section), hence overall it was fit perfectly with our needs. Moreover, Pandas offers a lot of features to work with the data e.g group element, search element, delete and, update values.

For the purpose of developing an algorithm able to discover consistent topics, it was necessary to use a package that provided some tool to compute the frequent itemsets and the association rules. The final decision was using mlxtend[7]. In particular, the mlxtend.TransactionEncoder was used to transform the vector of tweets of each day into a DataFrame format suitable for computed the frequent itemsets. The TransactionEncoder learns the words present in each tweet of the day taken into consideration and, it transforms the input vector(a Python list of lists) into a one-hot encoded NumPy boolean array, which in a second moment can be converted for our convenience into a pandas DataFrame.

Thus, we obtain a matrix with m columns according to m words found, all the possible words present in the tweets regard the day and n rows as the number of the tweets for that day. The value of each matrix entry (θ) can be either True if the word appears in the tweet, or False if it doesn't appear.

$$\theta_{i,j} = \begin{cases} True & if \ \ word_j \in tweet_i \\ False & otherwise \end{cases}$$

After that has been used mlxtend.frequent_pattern which provides both the algorithm for the frequent itemset and the association rules. For the frequent itemsets are available 3 types of algorithms called: Apriori, FP-Growth, and F-max. From these has been decided to use FP-Growth.

FP-Growth is an algorithm for extracting frequent itemsets with applications in association rule learning that emerged as a popular alternative to the established Apriori algorithm. In particular, and what makes it different from the Apriori frequent pattern mining algorithm, FP-Growth is a frequent pattern mining algorithm that does not require candidate generation. Internally, it uses a so-called FP-tree (frequent pattern tree) datastrucure without generating the candidate sets explicitly, which makes it particularly attractive for large datasets.

Hence, as can been seen from the article 'An implementation of the FP-growth algorithm'[8] FP-growth it's the faster algorithm available.

In order to work the FP-Growth needs to have as input a one-hot encoded NumPy boolean array, in our case later transform in pandas DataFrame. This is the motivation for the use of the TransactionEncoder described above. The associated_rules algorithm instead works smoothly with the output of the FP-Growth algorithm.

Both the algorithms provided by mxtlend allow setting the thresholds described in Section 3. to discover and to keep only the elements that agreed with the thresholds decided.

[5]https://www.python.org

[6]https://pandas.pydata.org

[7]http://rasbt.github.io/mlxtend/

[8][1, OSDM '05: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementationsAugust 2005 Pages 1–5https://doi.org/10.1145/1133905.1133907]

All other types of works as the Checking and Selection described in Section 5.3 have been done always using pandas tools. Also, all the filter after the frequent itemsets in order to keep only the items of exactly two words has been done with pandas tools. The rest of the code is only the combination of tools provided by Python.

7 DATASET

The final dataset given as input to the code in order to extract the consistent topics over time is the result of a merger of different datasets and some preprocessing work done for having cleaner data to work with.

7.1 Sources and Union

The datasets have been taken directly from Kaggle[9]and are still available to the following links:

- https://www.kaggle.com/yazanshannak/us-covid-tweets
- https://www.kaggle.com/gpreda/covid19-tweets
- https://www.kaggle.com/bugraayan1/covid19-250000-tweets?select=covid19_en.csv
- https://www.kaggle.com/yazanshannak/us-covid-tweets

After having downloaded the different datasets, which had different fields, the datasets have been merged into a single dataset in order to run the preprocessing code just one time. In the merged dataset have been kept only the field *"date"* which indicates the day and the hour the tweet has been published and the *"text"* field which contains the text of the regarding tweet.

7.2 Preprocessing

The pre-processing of the dataset it's a crucial point in order to prepare as best as possible the data to improve the quality of the output later on. The first step made was to delete the hour keeping only the day from the *"date"* field.

Thereafter the main work has been done into the *"text"* field. The *"text"* field contains the text of the tweet and obviously, in a tweet, there are many things we are not interested in. For this reason from the *"text"* field have been removed all the emojis, all possible links and all character different from [a-z,A-Z], so also the punctuation and all the number has been deleted. After having done some noisy elimination we start to work on the tweet's text in order to keep only the interesting words. Successively all the words have been put in lowercase and have been removed the stopwords using NLTK(Natural Language Toolkit)[10] package.

After that has been decided to lemmatizing the words always using the NTLK package. Lemmatization is the process of converting a word to its base form considering the context. It is an extremely resource-hard (either time and space on RAM) process but it's fundamental since there are a lot of ways to write down the same concept e.g eat, eating. It's essential to have all the words in the same form if we want to discover some pattern in the tweets' words, if this process had not been done we will have results completely wrong since two words that mean the same things are processed as two different words.

Hence, has been deleted also the words that appeared more than one time in a tweet since we are searching for correlations between words in tweets so it doesn't matter if a word appears two times.

[9]https://www.kaggle.com

[10]https://www.nltk.org/#

An important decision has been done to delete also the most 10 common words among all the available tweets. The words deleted are the following: coronavirus, say, covid, go, people, time, get, like, need, pandemic. The choice has been taken because since we are interested in topics not always popular, hence the most 10 popular words surely will not appear in the consistent topics, since there are very popular. Moreover eliminating these words has been discovered the perfect trade-off in order to speed up the code for finding the consistent topics in a second moment. Also, the top 10 rare words have been deleted since it will be no possibility they will become frequent.

At a later time have been deleted all the word made by a single character, maybe they were the result of some noise and in any case, a word made by one character is not something interested to extract. Last but not least have been deleted also all the tweets either empty or with a single word. To conclude the text of all tweets has been tokenized to have a vector of words.

The final step was to sort the tweets in ascending order according to the date regarding the tweet and save the final pre-processed dataset into a .pkl (pickle) file.

The pre-processed dataset is available in the /data folder [§] while the code is available in the /src folder.

8 EXPERIMENTAL EVALUATION

In this section, an evaluation is conducted regarding the results and how the code was implemented. The section is divided into two different parts.

The subjective evaluation part where the results obtained will be evaluated based on subjective and personal metrics.

In the objective evaluation, all implementation choices as tools, thresholds, dataset format, and execution time will be explained in detail.

In the last section, are present some personal consideration about the results achieved.

8.1 Subjective Evaluation

8.1.1 Results. The subjective evaluation of the results consists of checking whether the consistent topics produced correspond to reality. A singular perspective of various users is needed to make a comparison. For this reason, 5 different people were asked to read the results and give their opinion if they were consistent topics. The 5 people involved have responded all confirming the results. As you can see from Table 1, where a small set of consistent topics are shown, you can see that with personal knowledge of the situation we live in, these are topics that are not always popular but when mentioned over a timespan they become so.

The list of all consistent topics produced as audio is available in the results_project_dataset.csv [‖] file. As was to be expected, most of the results are related to the Coronavirus pandemic. Despite this other interesting results can be noticed and as soon as they are read they will certainly come back to the mind of those who are evaluating them as actual consistent topics.

8.1.2 Number of items as output. The number of results as output is completely user-independent, it only depends on the arguments of the dataset. The choice of not inserting a variable selected by the user, but let the code decide, was made to have impartiality and keep all results that through the code get out to be interesting.

word1	word1
distancing	social
hand	wash
toll	death
state	united
white	house
distance	social
front	line
trump	donald
paper	toilet
take	seriously
conference	press
task	force
supply	chain
hand	sanitizer
cruise	ship
case	total
small	business
test	kit
grocery	store
rate	mortality
health	department
precaution	take
market	stock
save	life
mask	face
stay	home
prevent	spread
death	total
worker	healthcare
hospital	patient
case	official
china	wuhan

Table 1: Sample of consistent topics [†]

The output depends only on the implementation of the code and on the arguments that it finds inside the dataset given as input. Obviously, the bigger is the dataset the more will be the number of returned results for a simple mathematical matter of proportionality. The threshold of frequency for which are considered valid results only the elements with a frequency between 15% and 75% of the total number of days is, in my opinion, the correct choice. The frequency means in how many days the correlation between words has been found frequent.

In the file present in the Evaluation folder, there is a total_freq.csv [‡‡] file in which all the elements found without setting the final frequency threshold are shown. It is possible to see how the elements with a frequency greater than 75% of 123 (days), that is 92 (days), are correlations of obvious words. While those with a frequency less than 15% of 123 (days) that is 18 (days) are almost very causal and are not consistent topics very interesting and relevant. Further lowering the threshold would have resulted in too many results. The graph shown in Figure 1 represents the number of outputs found as the support threshold varies. is very interesting to analyze because it shows that as the threshold

(§) Files are available both in .csv format (for direct viewing) and in pickle format (used by the code).

(‖) The file can be found inside /bin folder.

(†) The complete set of the output consistent topics is available in the result.csv file.

(‡‡) The files can be found inside /bin/Evaluation folder.

of support increases, the number of results obtained decreases almost exponentially. This graph reinforces the choice of the support threshold at 0.003, which we will discuss shortly.

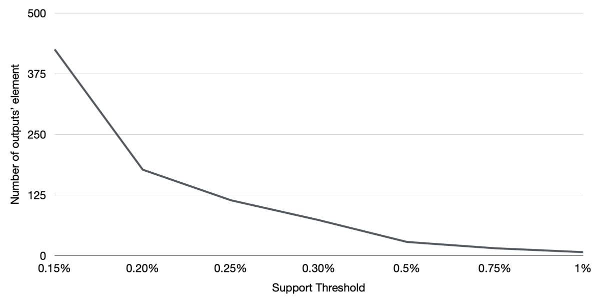

Figure 1: Number of outputs' elements

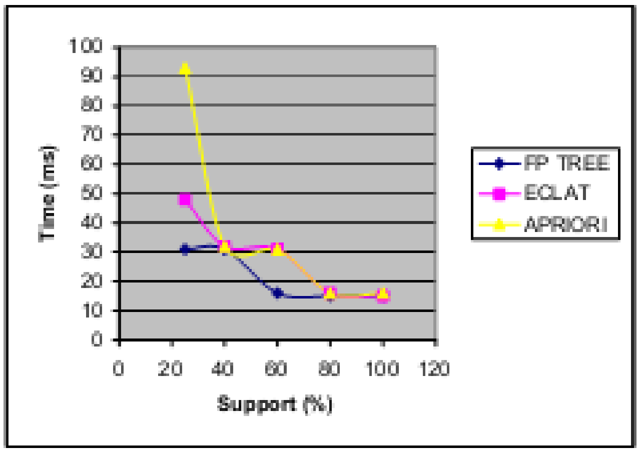

Figure 2: Execution time FP-Growth vs Apriori

8.1.3 Timespan. Since the goal of the project was to find consistent topics over time, it was essential to define what a period is. Working with a dataset that contains 123 different days it was possible to consider one day as a period. In this way, each day corresponded to a different period.

Have been made some tests to consider a period as the set of several days close together. But since all this work was done on a very large database the results did not change at all. On the contrary, the execution performance was worse because it had to calculate the frequent itemsets and association rules of a larger number of tweets at a time, the set of tweets of the days considered close together instead of the tweets of a single day.

This choice was possible since using a huge dataset in which many different days are available the results will be normalized over time. If I had instead used a dataset with fewer days, I would have had to consider the periods differently from considering a period a single day.

8.2 Objective Evaluation

8.2.1 FP-Growth vs Apriori. In this project, it was decided to use the FP-Growth algorithm, instead of Apriori, because of its speed of execution with very low support thresholds. As you can see from Figure 1, which plot the different computational time of the algorithm according to the selected threshold, taken directly from the publication of Kanwal Garg and Deepak Kumar[11].

Since throughout all this work we have gone into some very low threshold for the support, the use of this type of algorithm, compared to the others available, turn out to be the best choice.

8.2.2 Metrics. The choice of thresholds for the support, confidence, and lift metrics were made based on some objective and subjective evaluations. The subjective evaluations were made by running the program countless times until the results converged into meaningful results. Besides, averages of the various metrics were taken to understand how the values presented themselves. Always in the folder bin/Evaluation are available some files named rules_###.csv [**] and support.csv[‡‡] where the various values of support during the frequent itemset and the rules of association (with lift and confidence) during the execution of the code through which it has been possible through a subjective and visual choice to understand which would have been the correct thresholds to select.

Instead, the objective evaluation was done on a mathematical basis. Considering that there are 1.6 million tweets and 123 days, this means that on average there are 13 thousand tweets per day. Considering a threshold of support equal to 0.003 which means that a pair of words is considered frequent if it appears in at least 40 tweets in a day. Although it may not seem much, it is important to remember that having on average 10 words each tweet using the combinatory calculus rule the number of possible word pairs per tweet is about 45. This means that the probability of finding the same couple of words of exactly 2 words in various tweets is very low. Therefore, the selected threshold is found to be correct in the end.

The other reason is explained by Figure 3. It is shown through the graph as the number of uncommon elements and the correlation strength between itemsets is inversely proportional. As the support increases, the number of "non-traditional" results decreases exponentially while the correlation strength between the words that compose an item increases. In this way, the perfect trade-off occurs where the two curves of the graph touch, i.e. with a support threshold at 0.003.

Lift and confidence instead don't depend on the dataset but only on the frequent items found. For this reason, the threshold for the confidence has been set to 0.5 which is universally recognized as the correct threshold, as we have seen during the course.

The threshold for the lift was set to 10 because I was interested in keeping only the really interesting items since I was analyzing a large dataset. The other reason why the lift metric was set to 10 is to have a reasonable number of outputs. Decreasing the lift increased the output results in a polynomial way. After various tests and various attempts, the choice has fallen on the setting to 10 which had the best trade-off between the number of results and interesting results.

Always in the folder Evaluation, there are files called result_valuesupport_valueconfidence_valuelift.csv [¶] in which you can see how by setting the metrics to different values you could not get satisfactory results.

[11] [2, KanwalGargandDeepakKumar.2013.ComparingthePerformanceofFrequent Pattern Mining Algorithms. International Journal of Computer Applications 69 (05 2013), 21–28. https://doi.org/10.5120/12129-8502]

(**) The files can be found inside /bin/Evaluation/rules folder.
(¶) The files can be found inside /bin/Evaluation/threshold folder.

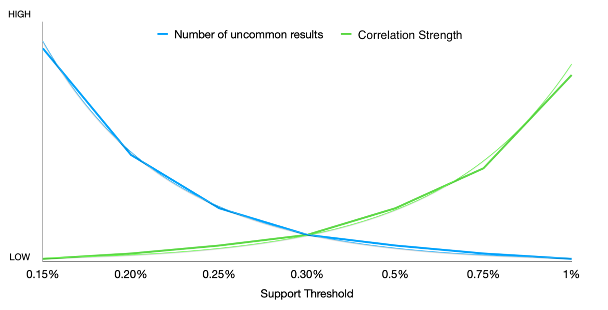

Figure 3: "Uncommon" results vs Correlation strength

8.2.3 Dataset's Format. The final processed dataset has been saved into a pickle[12] file. The pickle module implements binary protocols for serializing and de-serializing a Python object structure. The choice has been made to speed the execution of the code. It has been proved that the pickle files are faster to process both to load, to work with, and to save files.

All the information and the benchmarking of pickle compared to the other types of file in which a dataset can be saved is available at **this link**, coming from the reliable and well-known website TowardsDataScience[13] owned by Medium[14].

8.2.4 Execution Time. The code, to computed the consistent topics for 1.6 million tweets split into 123 days takes about 7 minutes. To make some experiments during the development has been used also some smaller datasets achieved kept only some tweets from the final dataset. So it's possible to see in Figure 2 the different times required according to the number of rows (i.e the tweets) present in the dataset.

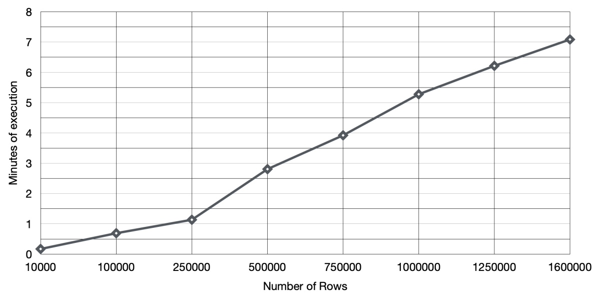

Figure 4: Execution Time

8.3 Final Consideration

I believe that the objective required by the project has been achieved with good results. Having decided to work with a large dataset the execution time is slightly high, but the results are worth the wait.

Were found many consistent topics and were kept only those interesting. You can see from the results how the facts related to Covid influence a lot the topics that are discussed on social. Besides, many correlations that came out surprised me as they are words that are used (e.g. sign-petition or supply-chain) together but not just the first thing that pops into mind when thinking about it. This was precisely the goal required by the project and was achieved satisfactorily.

The only note I would make is that perhaps it would have been necessary to find a library to remove, in the pre-processing phase, the cities. Many American cities (e.g. New York, Los Angeles) are composed of two words that are obviously always together in the tweets and therefore appear in the results.

If you are interested in the project code you can refer to this **link** which directs you to the GitHub folder containing code/data and execution instructions.

REFERENCES

[1] Christian Borgelt. 2005. An Implementation of the FP-Growth Algorithm. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations (OSDM '05)*. Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/1133905.1133907
[2] Kanwal Garg and Deepak Kumar. 2013. Comparing the Performance of Frequent Pattern Mining Algorithms. *International Journal of Computer Applications* 69 (05 2013), 21–28. https://doi.org/10.5120/12129-8502
[3] Humanitas. 2020. *Twitter.* Encyclopædia Britannica. https://www.britannica.com/topic/Twitter

[12]https://docs.python.org/3/library/pickle.html
[13]https://towardsdatascience.com
[14]https://medium.com