Analysis of Frequently Co-occurring Terms on Twitter Data

Mithat Sinan Ergen
mithatsinan.ergen@studenti.unitn.it

1 INTRODUCTION & MOTIVATION

Twitter is a prominent social platform where approximately 500 million tweets are published by various users everyday. The topics of tweets include a myriad of different topics such as personal life, trending events in the world, advertisement, sports and much more. This not only makes Twitter a trove of information but also turns it into an essential data source for in depth analysis. Hot topics of analysis on Twitter data include sentiment analysis, topic modelling, knowledge discovery and frequent term analysis.

In this study, we investigate set of terms that become frequent together throughout time. We use the COVID-19, Pfizer Vaccine and World Cup 2018 datasets available on Kaggle website to perform our analysis. After importing the data, we perform preprocessing steps to remove features that are not used and words, URLs that might interfere with the analysis. The preprocessing steps will be explained in detail in the dataset section of this paper. After preprocessing, we build the co-occurrence matrix of words in tweets one day at a time to observe terms that appear together frequently over time.

In this research, since the main topics of the tweets are COVID-19 and World Cup 2018, it is assumed, before any work is performed, that tweets would be generally focused on health and sports related issues over time. This hypothesis will be verified or rejected by the end of the evaluation.

In the next section, we will introduce the related work on this topic, then we will define our problem statement. In the solution section, we will explain our solution steps. We will continue with the implementation section in which we describe the tools used to reach the solution. To conclude, we will evaluate our results.

2 RELATED WORK

As topic clustering is becoming popular, there are various researches on this field including search queries, word co-occurrence and document clustering. Mathiesen et al. (2014) [3] worked on statistics of co-occurring keywords on Twitter. They concentrated on tweets that are including famous brands such as Apple, IBM, Starbucks and keywords associated with tweets that are about them. They included time frames to elaborate their research and found results such as the brand name "Starbucks" and the word "sleep" appearing together in tweets in the morning and at night. This was due to the need of coffee having a peak at these times.

Another study, by Jing Kong et al. (2016) [2] worked on search queries to apply word co-occurrence clustering. First, they used standard text cleaning in which stopwords and punctuation marks were removed. Then they applied stemming and segmented the queries into bags of words. Next, they created a hashmap from the bag of words, they assigned words to keys and values to queries having those words. After creating the hashmap, they initialized topics from a subselection of keys of the hashmap. This subselection was made according to lift scores which was calculated according to user actions taken related to the word. By using lift scores, they managed to avoid initializing unimportant topics. Finally, they expanded the topics with words by using word co-occurrence.

Since tweets are texts that are limited to 280 characters(140 characters before 2017), they are considered as short text. Therefore, tweets show similar properties to search queries which are generally short. One of the problems that is faced include data sparsity problem which can be defined as difficulty of correctly categorizing the topic of a tweet due to the low number of words. In long documents, it is relatively easier to detect the relationship between the words than in short documents which have low word count. Weng et al.(2010) [7] studied this issue and aggregated all the tweets belonging to the same user to a single document. Hong et al. (2010) [1] handled this matter similarly but they also aggregated tweets with same words. They both achieved better results than traditional methods. Inspired by these previous studies, this problem was also tackled by Yan et al. (2013) [8] who used biterm topic model(BTM) in order to generate topics from unordered word pairs in the data considering the weight of each term as equal. To solve the problem of sparsity, they used aggregated patterns in the whole corpus.

In general, studies include clustering techniques to categorize tweets by using hashtags. Yan et al. (2013) [8] declares that hashtags have three different types. First type of hashtags such as #burningman either mark events or topics like a concert or political campaign, second type includes hashtags like #beyourway defines the type of content and the third type is for contacting a company like Facebook using #fb hashtag. Since this research was mainly focused on determining topics, they chose 50 frequent hashtags from the first type of hashtags. They moved documents with the same hashtags in the same cluster and assumed that these clusters should have low intra-cluster distances and high inter-cluster distances.

In this study, we are planning to follow a similar approach to previous studies and put small portions of their ideas together to form a model to determine terms that are co-occurring frequently in time.

3 PROBLEM STATEMENT

In this section, we define the problem, explain the inputs and the outputs. We use 3 different data from Kaggle website. They are all in csv format. During our work, they are imported and stored as a dataframe. This dataframe has 2 fields which are tweet text represented as string and date as datetime. Our main model takes as input two parameters: a dataframe of tweets, which are a set of words related to COVID-19, Pfizer's vaccine and World Cup 2018, and number of terms as integer which represents how many frequent term couples we want to extract from data for each date.

The output of this model is a dictionary object with frequent tuples as a list of strings, their count as integer, date as string and frequency as integer. We set a frequency threshold of 0.1 and evaluate tuples that have more than 0.1 frequency as frequent for a given date. Frequency is calculated by dividing the number of co-occurrences divided to the number of tweets for the day. Inputs and outputs of the main model are given in Table 1.

Key problems are deliberate spelling errors or abbreviations in tweets and meaningless tweets from some users that might

Model for Frequent Terms in Time			
I/O	**Object**	**Type**	**Content**
Input	Tweets	dataframe	Tweet text as string, tweet date as datetime
Output	Terms	dictionary	Frequently co-occurring terms as tuples, dates as datetime, counts as integer, frequency as integer

Table 1: Inputs and outputs of the main model

create noise if we want to infer a meaning out of the frequent terms in time.

4 SOLUTION

In this section, we explain our solution to the problem defined in the problem statement. We will explain our steps of preprocessing, how we created the co-occurrence matrix for terms that became popular in time with details and the problems we encountered during implementation. We also show some of the results but details of evaluation are given in the evaluation section. A very general outline of our solution is given in Figure 1.

First of all, we import the necessary libraries of Python that will be useful in our study. Some of these libraries include pandas, numpy, matplotlib, nltk and sklearn. We will use pandas for data analysis and manipulation, numpy for creating and modifying arrays, matplotlib for visualizing our results, nltk for natural language processing and finally sklearn for clustering and topic modelling.

After we import the data from the csv files, we notice that there are some fields which are not going to be used according to the purpose of this study. So, in order to save time and reduce computational complexity, we remove these unused features. We will concentrate mainly on the tweet text and dates to determine the most frequent terms that are appearing together in time. Also, we expand our data with another set of tweets about World Cup 2018. We also import a set of tweets about Pfizer's COVID-19 vaccine. They are not in the same format as our original dataset, so the first step is to regularize the format and aggregate them together in one data frame. The details of the work we performed on data are given in the dataset section of this study.

After the aggregation of datasets into one big data frame, we can start our preprocessing steps. First of all, we define a function to remove emojis. Since emojis are defined in Unicode blocks, it is possible to define them as intervals of Unicode and remove them from tweet texts. Next, we define a function for tokenization. Tokenization is basicly splitting a text into smaller parts called tokens. In our case, we split texts by words. Then, we define another function which converts text to lowercase, removes mentions, URLs, punctuation marks, English stopwords. We also use the function emoji removing function we defined before in this function. An important point to emphasize is that the order and the way of removal is essential in preprocessing. Since tweets are not always written in a grammatically and orthographically correct way, we may have inaccurate results if we are not careful about some key concepts. For example, some users do not leave a blank space after using punctuation marks. That is why, when we remove punctuation marks if we do not put a blank space we

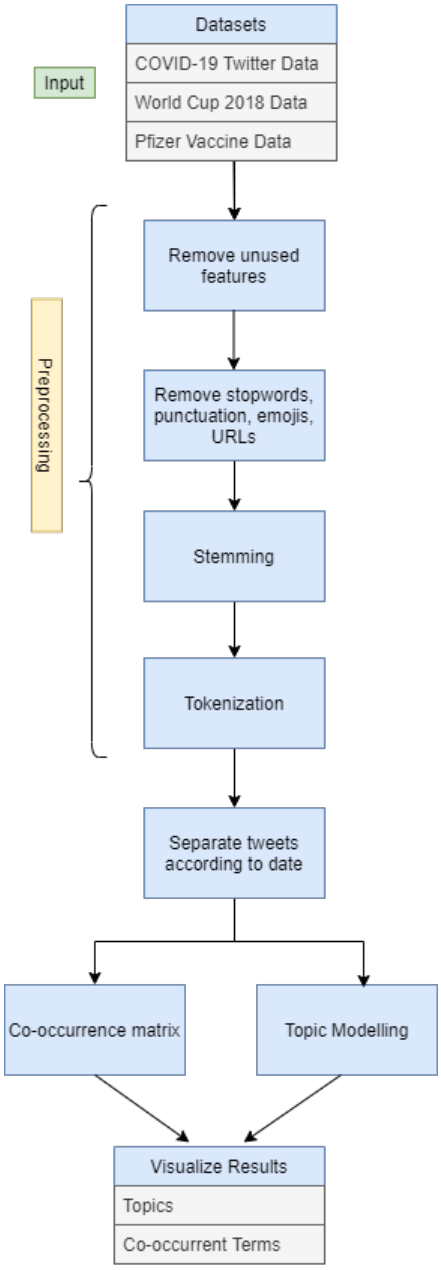

Figure 1: General Outline

may actually combine two words together which may not only create a meaningless word but also reduce the count of these two separate words in the word count. Examples are shown in Table 2. So, in order to avoid this problem, we replace every punctuation mark in the tweets with a blank space. We use a readily available punctuation mark set that is in string library of Python. We notice that there are some punctuation marks in the tweets that are not available in this set, so we manually add them to the list and remove them. We use the same library to remove digits in the tweets. After removing digits, tweets do not contain any dates or numbers.

After the punctuation marks are removed, we also have to remove the URLs in the tweets since they do not provide any

Removing Punctuation Marks		
Original Text	**After Removal**	**Desired Output**
imagine...going back	imaginegoing	imagine going back
meanwhile,absurd	meanwhileabsurd	meanwhile absurd
president:Trump	presidentTrump	president Trump
he said:"we"	he saidwe	he said we

Table 2: Removing punctuation marks in tweets, a possible error and desired output

useful information on our goal. In order to do this, we use regular expressions and remove the parts that start with the pattern "http".

Next step is to remove the English stopwords which are frequently used in tweets. Stopwords can be defined as commonly used words in a specific language that generally do not help to infer meaning from a sentence. An example set of English stopwords are given in Table 3 . We use Python's gensim library to remove English stopwords from tweets.

English Stopwords
a | about | above | across | after | afterwards | again | against
all | almost | alone | along | already | also | although | always
am | among | amongst | amoungst | amount | an | and | another
many | may | me | meanwhile | might | mill | mine | more
moreover | most | mostly | move | much | must | my | myself
was | we | well | were | what | whatever | when | you | your | yours

Table 3: An example set of English stopwords

After we remove the English stopwords from tweets, they are reduced significantly and we can start working on them. First we tokenize the tweets, then we use stemming to reduce the words. Stemming is the process of reducing a word to its root. By using stemming we can have a more accurate word count since the words will be counted together even if they are in a different form. Alternatively, we can also use lemmatization which is very similar to stemming but it reduces the word into an actual meaningful word while stemming output might have no meaning. Examples of stemming and lemmatization are shown in Table 4.

After stemming, we count the frequency of unigrams and plot the graph of top 20 most frequent unigrams in the dataset. They are shown in Figure 2. We can notice that "covid" is the most frequent word in the tweets and the second most frequent word is "franc" which is the stemmed form of "France". We can expect to have these words in frequent terms appearing together in time.

After observing the graphs with unigrams, we proceed to creating a co-occurrence matrix for terms appearing together

Stemming vs Lemmatization		
Word	**Stemming**	**Lemmatization**
getting	get	get
changed	chang	change
studies	studi	study
climbing	climb	climb
meeting	meet	meet

Table 4: Examples of stemming and lemmatization

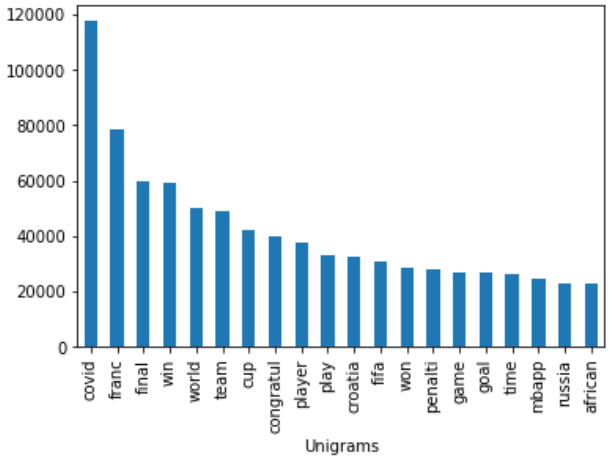

Figure 2: Top 20 Unigrams in the dataset

Co-occurring_terms	Count	Date	Frequency
('cup', 'world')	2652	30/06/2018	0.09
('argentina', 'franc')	2293	30/06/2018	0.08
('messi', 'ronaldo')	2026	30/06/2018	0.07
('penalti', 'save')	13360	01/07/2018	0.13
('final', 'quarter')	8589	01/07/2018	0.09
('penalti', 'shootout')	8445	01/07/2018	0.08
('case', 'covid')	1193	26/07/2020	0.16
('coronaviru', 'covid')	748	26/07/2020	0.1
('covid', 'new')	653	26/07/2020	0.09

Table 5: Co-occurring Terms Sample for 3 different dates

in tweets. We extract top 3 co-occurring terms for each date. A sample output for 3 dates is given in Table 5. We can observe that the most frequent unigrams "covid" and "franc" appear also in co-occurring terms. The algorithm for co-occurrent terms is given below in Algorithm 1.

Algorithm 1 Find Number of Co-occurrence

Require: dataframe of tweet text
  $com \leftarrow$ empty dictionary
  while tweet exists in dataframe **do**
    $words \leftarrow$ split tweet to words
    for i in range(length of words - 1) **do**
      for j in range(i+1, length of words) **do**
        $word1 \leftarrow words[i]$
        $word2 \leftarrow words[j]$
        if $word1 \neq word2$ **then**
          $com[word1][word2] + +$
        end if
      end for
    end for
  end while

The main goal of this work is to find the terms that occur together multiple times on different dates. Therefore, at the next step, we observe which tuples co-occurred several times on different dates. Results are shown in Table 6.

After we create the co-occurrence matrix and find the most frequent co-occurrent terms on different dates, we use Latent

Co-occurring_terms	Dates
('case', 'covid')	24/07/2020, 25/07/2020, 26/07/2020, 27/07/2020, 28/07/2020, 29/07/2020, 30/07/2020, 31/07/2020, 01/08/2020, 02/08/2020, 04/08/2020, 06/08/2020, 07/08/2020, 08/08/2020, 09/08/2020, 10/08/2020, 11/08/2020, 12/08/2020, 13/08/2020, 14/08/2020, 16/08/2020, 17/08/2020, 18/08/2020, 22/08/2020, 29/08/2020, 30/08/2020
('case', 'new')	25/07/2020, 01/08/2020, 02/08/2020, 29/08/2020, 30/08/2020
('coronaviru', 'covid')	24/07/2020, 25/07/2020, 26/07/2020, 27/07/2020, 28/07/2020, 30/07/2020, 31/07/2020, 01/08/2020, 02/08/2020, 04/08/2020, 06/08/2020, 07/08/2020, 08/08/2020, 09/08/2020, 10/08/2020, 12/08/2020,13/08/2020, 14/08/2020, 16/08/2020, 17/08/2020,18/08/2020, 22/08/2020, 30/08/2020
('covid', 'new')	26/07/2020, 27/07/2020, 28/07/2020, 29/07/2020, 30/07/2020, 31/07/2020, 06/08/2020, 07/08/2020, 08/08/2020, 09/08/2020, 10/08/2020, 12/08/2020, 13/08/2020, 14/08/2020, 16/08/2020, 17/08/2020, 18/08/2020, 22/08/2020
('covid', 'pfizerbiontech')	12/12/2020, 14/12/2020, 15/12/2020, 16/12/2020, 18/12/2020, 19/12/2020, 21/12/2020, 22/12/2020, 24/12/2020, 25/12/2020
('covid', 'test')	29/07/2020, 04/08/2020
('covid', 'vaccin')	11/08/2020, 12/12/2020, 13/12/2020, 14/12/2020, 15/12/2020, 16/12/2020, 18/12/2020, 19/12/2020, 20/12/2020, 21/12/2020, 22/12/2020, 23/12/2020, 24/12/2020, 25/12/2020, 26/12/2020
('cup', 'world')	30/06/2018, 02/07/2018
('penalti', 'save')	01/07/2018, 02/07/2018
('penalti', 'shootout')	01/07/2018, 02/07/2018
('pfizer', 'vaccin')	13/12/2020, 17/12/2020, 20/12/2020, 26/12/2020
('pfizerbiontech', 'vaccin')	12/12/2020, 13/12/2020, 14/12/2020, 15/12/2020, 16/12/2020, 17/12/2020, 18/12/2020, 19/12/2020, 20/12/2020, 21/12/2020, 22/12/2020, 23/12/2020, 24/12/2020, 25/12/2020, 26/12/2020

Table 6: Co-occurring Terms in Time

Dirichlet Allocation(LDA) for tweet groups of each date, which is a commonly used statistical model to detect topics in documents. We implement it on tweets of different dates to infer 2 of the prevalent topics of each day, then we compare these topics with the most frequent co-occurring terms we obtained before. The algorithm's pseudocode is shared in Algorithm 2. Resulting topics are shown in Table 7. By looking at the results, it is possible to comment that the dominant topics are related to COVID-19, World Cup final and penalties in World Cup 2018.

Algorithm 2 Finding Topics in Tweets

Require: dataframe of tweets(df), number of topics(nOfTopics), number of words(nOfWords)
$topics \leftarrow$ empty list
$vectorizer \leftarrow$ initialize CountVectorizer object
$text\_sample \leftarrow df.stemmed\_text$
$date \leftarrow df.date$
$tf \leftarrow$ vectorize text sample
$feature\_names \leftarrow get\_feature\_names()$
$model \leftarrow LatentDirichletAllocation(nOfTopics, random\_state = 0)$
$topic\_model \leftarrow model.fit(tf)$

5 IMPLEMENTATION

In this section, we introduce the tools we used for the implementation of our solution and for what we used them.

First of all, we used Python programming language for this project. First Python library we used was pandas. Pandas library allows data import and data manipulation operations like merging and reshaping. We used it to import Twitter data, store them in a dataframe, for merging multiple csv data into one dataframe and for removing unused features. Next, we used numpy, which is an open-source Python library that contains array and matrix data structures. We used nltk library for stemming, sklearn library for Latent Dirichlet Allocation(LDA), time library to calculate execution times, collections library to create a default dictionary, string library to remove digits and punctuation marks from tweets, gensim library to import English stopwords, operator library to get items in a dictionary and re library for regular expressions.

In terms of visualisation, we opted to use Excel to draw graphs for frequencies of word pairs.

6 DATASET

The original dataset for this research is COVID-19 Twitter data from Kaggle website[4]. We added two more datasets to this which include Pfizer Vaccine Twitter data[5] and World Cup 2018 Twitter data[6] from Kaggle. The original form of each dataset is shown in Table 8, Table 9 and Table 10.

Dataset : COVID-19			
Field	**Type**	**Field**	**Type**
user_name	object	user_verified	bool
user_location	object	date	object
user_description	object	text	object
user_created	object	hashtags	object
user_followers	int64	source	object
user_friends	int64	is_retweet	bool
user_favourites	int64		

Table 8: Structure of COVID-19 Twitter Dataset

Dataset : Pfizer Vaccine			
Field	**Type**	**Field**	**Type**
id	int64	user_verified	bool
user_name	object	date	object
user_location	object	text	object
user_description	object	hashtags	object
user_created	object	source	object
user_followers	int64	retweets	int64
user_friends	int64	favorites	int64
user_favourites	int64	is_retweet	bool

Table 9: Structure of Pfizer Vaccine Twitter Dataset

Dataset : World Cup 2018			
Field	**Type**	**Field**	**Type**
ID	int64	RTs	int64
lang	object	Hashtags	object
Date	object	UserMentionNames	object
Source	object	UserMentionID	object
len	int64	Name	object
Orig_Tweet	object	Place	object
Tweet	object	Followers	int64
Likes	int64	Friends	int64

Table 10: Structure of World Cup 2018 Twitter Dataset

Since our main goal is to identify terms that are appearing together frequently for different dates, we concentrate on the tweet text and date of each dataset. That is why, we are not giving detailed explanation of every field. We reduce each dataset to 2 fields which include tweet text and date. COVID-19 and Pfizer vaccine datasets have a similar structure but World Cup 2018 dataset has different field names. So we merge tweet texts and dates from these datasets, by respecting the field names in each dataset, into one dataset with two fields called text and date.

COVID-19 dataset originally has 179108 rows and 13 columns, Pfizer Vaccine dataset has 1591 rows and 16 columns, World Cup 2018 dataset has 530000 rows and 16 columns. We used 1/3 of the World Cup 2018 data. Aggregated dataset has 357201 rows and 2 columns.

Dataset	Size
COVID-19	179108 rows x 13 columns
Pfizer Vaccine	1591 rows x 16 columns
World Cup 2018	530000 rows x 16 columns

Table 11: Dataset Sizes

Preprocessing steps performed to obtain the final form of the dataset are:

- Removing unused fields in datasets
- Merging datasets
- Converting tweet texts to lowercase
- Removing missing values, English stopwords, emojis, punctuation marks, URLs, digits, mentions from tweets
- Tokenization
- Stemming(also possible to use lemmatization)

The final form of the dataset used in this research is given in Table 12. It has a simple structure since we are investigating tweet texts from different dates and we are not interested in location or user information.

Final Dataset	
Field	**Type**
text	string
date	datetime

357201 rows x 2 columns

(1/3 of World Cup 2018 data used)

Table 12: Final Form of the Dataset

7 EXPERIMENTAL EVALUATION

In this section, we visualize and evaluate our results at each step. First of all, we are going to show how the frequency of different co-occurring terms change in time. As previously shown in this work in Table 5, there are 12 different co-occurring term sets that are occurring together again in multiple dates. They have different frequency in every date. In Figure 3, the frequency of terms 'case' and 'covid' co-occurring is shown. We can observe that the frequency starts as low as 0.06, then raises to 0.16 and stays in average between 0.08 and 0.1. The average co-occurrence frequency of these terms is 0.085 and number of times these terms became frequent together is 26.

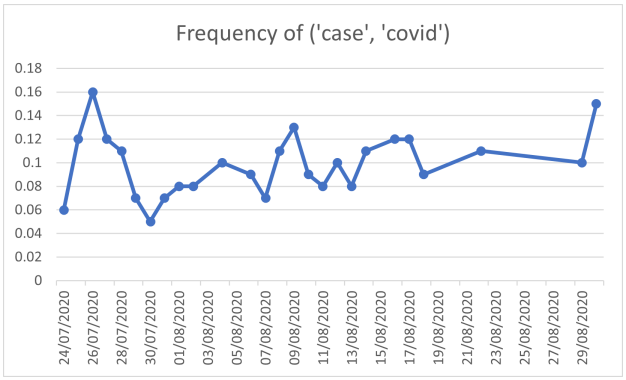

Figure 3: Frequency of terms 'case' and 'covid' on different dates

In Figure 4, the frequency of terms 'case' and 'new' co-occurring is shown. We can observe that the frequency starts from 0.07 and stays there for 3 observations. We can notice a significant date difference until these terms become frequent together again and the next time they become frequent together, their frequency increases by 57% to 0.11. The average co-occurrence frequency of these terms is 0.08 and number of times these terms became frequent together is 5.

Next, we observe the terms 'coronaviru' and 'covid'. Since we applied stemming, the letter s in the end of coronavirus is omitted. We can see the frequency results in Figure 5. Their frequency starts with 0.08 and stays constant for the next observation. Then it raises to 0.1 and follows a constant decrease to 0.06 for 3 observations. The average co-occurrence frequency of these terms is 0.075 and number of times these terms became frequent together is 23. We can observe that these terms stayed constant for consecutive days, thus they may represent an ongoing situation.

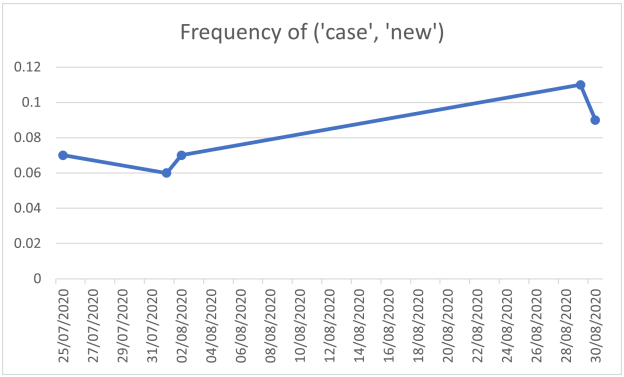

Figure 4: Frequency of terms 'case' and 'new' on different dates

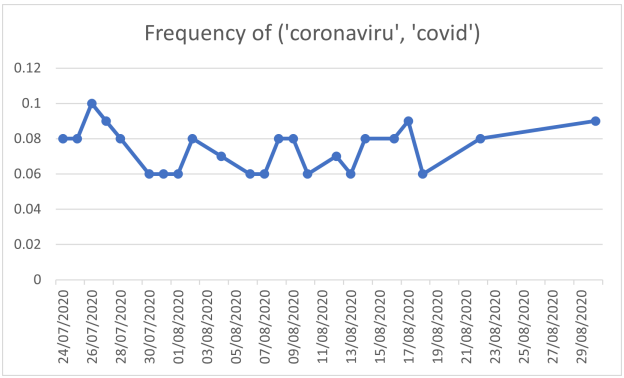

Figure 5: Frequency of terms 'coronaviru' and 'covid' on different dates

Next tuple of frequently co-occurring terms are 'covid' and 'vaccin'. This tuple has interesting results shown in Figure 6. Results are interesting because these terms become frequent on the date 11.08.2020 and does not become frequent again for 4 months. So we can say that the topic of vaccine was discussed in August 11 but then it was not very popular for the following days until December when it became very popular with an average frequency of 0.36 in December if we count the frequency on August 11 as an outlier. The average frequency in total is 0.24 and number of times these terms became frequent together is 15.

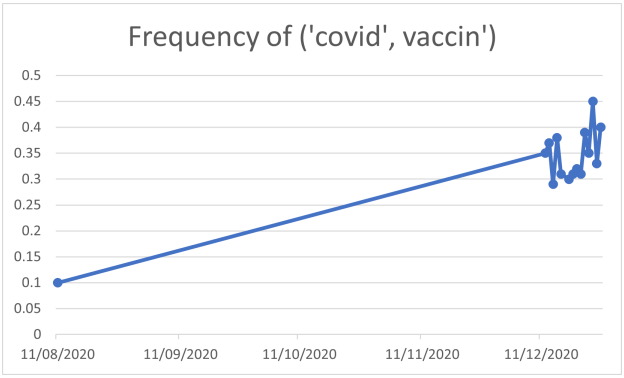

Figure 6: Frequency of terms 'covid' and 'vaccin' on different dates

Next, we are going to observe co-occurrent terms with high frequency. We defined our threshold value to be 0.1 and we filtered unique tuples with 0.1 or higher frequency. There were 15 results but we will show the ones that had only one appearance on a specific date. They are shown in Table 13. What we can infer from these results is that there was an event that became popular on that date and then it ended. For example, on July 1 the terms 'penalti' and 'save' became popular, thus we can say that penalties were saved on that date in the World Cup 2018. When we Google "july 1 2018 penalty save", we see that the goalkeeper Kasper Schemeichel saved 3 penalties that day and that is why it became viral on Twitter on that specific date but never became popular again.

Co-occurring_terms	Count	Date	Frequency
('penalti', 'save')	13360	01/07/2018	0.13
('charact', 'proud')	2479	03/07/2018	0.12
('bottl', 'proud')	2475	03/07/2018	0.12
('bottl', 'charact')	2475	03/07/2018	0.12
('power', 'song')	6678	04/07/2018	0.45
('exo', 'power')	11491	04/07/2018	0.77
('music', 'power')	5720	04/07/2018	0.38
('dubai', 'vaccin')	25	23/12/2020	0.2

Table 13: Co-occurring Terms with High Frequency on a Single Date

After observing co-occurrent terms with high frequency, we are going to compare the prevalent topics of different dates inferred with LDA to the co-occurring terms. As stated before, we can observe the results of LDA in Table 7. We can see that the topics proposed by the model for 30.06.2018 are related to word pairs ('mbapp', 'goal'), ('messi', 'ronaldo'), ('final', 'quarter'). In the co-occurring terms table we only have ('cup', 'world') terms occurring frequently together, so there is no evident correlation on this date. But on 01.07.2018, we can see that the proposed topics are related to ('croatia', 'denmark'), ('save', 'penalti') and ('penalti', 'final'). In the frequently co-occurring terms table we have a match for ('penalti', 'save') pair. So maybe we can say that in the match between Croatia and Denmark, there were some penalty shots taken and some of these shots were saved. For the tweets in 2020, there are multiple similarities between the frequent co-occurring terms and topics inferred by LDA such as ('covid', 'vaccin') pair in 11.08.2020, ('covid', 'test') pair in 04.08.2020 and ('case', 'covid') in 26.07.2020. So we can say that co-occurring terms are related to topics inferred by LDA.

In terms of computational complexity, we measured the execution time of tasks by running the code with different parameters at each time. Parameters used for each case are given in Table 14. For all cases, execution times are shown in Table 15. We can observe that preprocessing task takes around 21 seconds and stemming task takes around 85 seconds. We can see that topic modelling takes around 618 seconds if we try to infer 3 topics, and 569 seconds if we look for 2 topics in a dataset of 357201 rows and 2 columns. However, if we run the code with number of topics parameter set to 1, it takes around 155 seconds to finish. So we can comment that the number of topics parameter has a considerable influence on the execution time and it should be set to 1 if a faster execution is required.

Next, it takes around 92 seconds to find co-occurrent terms and the number of co-occurring terms does not have a considerable

impact on the execution time. In terms of scalability, it may require a lot of computational power to use Latent Dirichlet Allocation for larger datasets but it is possible to use it with number of topics set to 1. In addition, if we run the code on GPU instead of CPU, we would have faster execution times.

	# of topics	# of words in topic	# of co-occurring terms
Case 1	3	2	3
Case 2	1	2	4
Case 3	1	3	5
Case 4	2	2	3
Case 5	1	2	3

Table 14: Execution Cases

Tasks	Case 1	Case 2	Case 3	Case 4	Case 5
Preprocessing	20.98	21.83	21.51	21.36	21.2
Stemming	84.92	90.45	86.92	82.87	81.22
Finding co-occurrent terms	91.92	90.71	90.07	86.03	96.54
Topic modelling	618.2	155.3	156.5	568.8	151.6
From start to finish	819.5	361.7	358.5	762.5	354

Table 15: Execution Time of Tasks in seconds

We believe that the main topics were detected by the topic modelling algorithm and frequent co-occurring terms. In the introduction section, we had a hypothesis about the tweets being about health and sports related issues, we can verify this by looking at the general topics found. So in general, we can comment that results were accurate but a better way to evaluate the results in a future work would be to add tweets about different topics from a similar date interval as COVID-19 data. In addition, another approach for future work would be detecting co-occurring hashtags in time instead of words.

REFERENCES

[1] Liangjie Hong and Brian D. Davison. 2010. Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics (SOMA '10)*. Association for Computing Machinery, New York, NY, USA, 80–88. https://doi.org/10.1145/1964858.1964870

[2] J. Kong, A. Scott, and Georg M. Goerg. 2016. Improving semantic topic clustering for search queries with word co-occurrence and bigraph co-clustering.

[3] Joachim Mathiesen, Luiza Angheluta, and Mogens Jensen. 2014. Statistics of co-occurring keywords on Twitter. (01 2014).

[4] Gabriel Preda. 2020. *COVID19 Tweets.* https://www.kaggle.com/gpreda/covid19-tweets

[5] Gabriel Preda. 2020. *Pfizer Vaccine Tweets.* https://www.kaggle.com/gpreda/pfizer-vaccine-tweets

[6] Rituparna. 2018. *World Cup 2018 Tweets.* https://www.kaggle.com/rgupta09/world-cup-2018-tweets

[7] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. TwitterRank: Finding Topic-Sensitive Influential Twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10)*. Association for Computing Machinery, New York, NY, USA, 261–270. https://doi.org/10.1145/1718487.1718520

[8] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, 1445–1456. https://doi.org/10.1145/2488388.2488514

Date	Topic 0 words	Topic 1 words	Topic 2 words
30/06/2018	mbapp	messi	final
30/06/2018	goal	ronaldo	quarter
01/07/2018	croatia	save	penalti
01/07/2018	denmark	penalti	final
02/07/2018	team	power	penalti
02/07/2018	gg	world	save
03/07/2018	proud	england	penalti
03/07/2018	bottl	penalti	southgat
04/07/2018	world	power	england
04/07/2018	cup	exo	final
24/07/2020	covid	coronaviru	covid
24/07/2020	coronaviru	covid	coronaviru
25/07/2020	covid	covid	covid
25/07/2020	peopl	test	case
27/07/2020	covid	covid	covid
27/07/2020	mask	amp	case
28/07/2020	covid	covid	covid
28/07/2020	pandem	test	case
29/07/2020	covid	covid	covid
29/07/2020	test	like	pandem
30/07/2020	covid	covid	covid
30/07/2020	new	peopl	amp
01/08/2020	covid	covid	covid
01/08/2020	mask	pandem	case
02/08/2020	covid	covid	covid
02/08/2020	case	test	coronaviru
04/08/2020	covid	covid	covid
04/08/2020	amp	test	case
08/08/2020	covid	covid	covid
08/08/2020	coronaviru	case	amp
11/08/2020	covid	covid	covid
11/08/2020	vaccin	pandem	case
12/08/2020	covid	covid	covid
12/08/2020	amp	coronaviru	case
13/08/2020	covid	covid	covid
13/08/2020	case	peopl	amp
14/08/2020	covid	covid	covid
14/08/2020	case	peopl	coronaviru
16/08/2020	covid	covid	covid
16/08/2020	pandem	case	die
17/08/2020	covid	covid	covid
17/08/2020	test	case	pandem
18/08/2020	covid	covid	covid
18/08/2020	amp	case	coronaviru
22/08/2020	covid	covid	covid
22/08/2020	coronaviru	case	case
29/08/2020	amp	covid	covid
29/08/2020	covid	pandem	case
30/08/2020	covid	covid	covid
30/08/2020	mask	case	peopl
12/12/2020	vaccin	pfizerbiontech	vaccin
12/12/2020	pfizerbiontech	vaccin	pfizerbiontech
13/12/2020	vaccin	pfizerbiontech	pfizerbiontech
13/12/2020	covid	vaccin	covid
14/12/2020	vaccin	pfizerbiontech	covid
14/12/2020	covid	vaccin	pfizerbiontech
15/12/2020	vaccin	pfizerbiontech	covid
15/12/2020	covid	vaccin	pfizerbiontech
16/12/2020	covid	vaccin	pfizerbiontech
16/12/2020	vaccin	covid	covidvaccin
17/12/2020	pfizerbiontech	pfizer	covid
17/12/2020	vaccin	vaccin	vaccin

Table 7: Latent Dirichlet Allocation(LDA) Results for some dates