The Ukraine Conflict

Algorithms for Massive Datasets

Name : Lo Bue Oddo Giulia

Student ID: 964543

I declare that the material which I now submit for assessment, is entirely my own work and has not been taken from the work of others. I understand that plagiarism, collusion, and copying are serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have not previously submitted this, or any part of this, assignment for assessment on this or any other study course.

Link to the repository

https://github.com/zulai98/AMD-project

Executive Summary

In light of the Russia-Ukraine conflict, the aim of this project is to analyse a collection of tweets (filtered on the Russia-Ukraine hashtags) and highlight the most frequent words (or combinations of words). Relying on the pre-existing Kaggle dataset for the tweets (12GB) we point out that, to produce results within a reasonable timeframe on a dataset of this size, we must use:

- a solid pre-processing pipeline (which reduces the number of items) and
- a (counting) algorithm which reduces the number of candidates to evaluate.

To this aim we also choose to analyse samples of the tweets instead of the whole 12GB dataset, analysing tweets for selected dates which are written in the English language. On the other hand, to generalise the applicability of our methodology to the whole dataset, therefore, to produce a sound and scalable experimental methodology (i.e. a methodology which can still be applied, with a reasonable run-time, to the massive 12GB dataset) we make heavy use of distributed computing through the PySpark module.

This paper is organised into the following sections:

Exploratory Data Analysis

After exploring the structure of the whole dataset to understand how data is organised, we narrow our analysis on tweets extracted on selected dates. After selecting tweets extracted on the 27th of February and performing exploratory data analysis, we stress the importance of a robust pre-processing pipeline (to remove hashes, emoticons and other 'spurious' characters). Moreover, as we wish to analyse the meaning of the most common words, we must remove the stop-words (i.e., common filler words). As stop-words depend on the selected language (e.g., "a", "the", "is" in English), we decide to select tweets which are written in a given language. Noticing that 60% of the tweets are in the English language, we choose to analyse the set of English tweets on selected days.

Constructing the Pre-Processing pipeline

We use SparkNLP to build the pre-processing pipeline.

The A-Priori algorithm and an overview of the experimental methodology

This section provides a theoretical interlude to the a-Priori algorithm and highlights the main features of how the pipeline was optimised (from ingesting data to cleaning and processing it) to ensure a meaningful, scalable and robust approach to analysing the whole collection of tweet-sets.

Selected experiments (description, results, analysis)

We perform several experiments to:

- Assess how the A-priori algorithm performs (computational time vs significance of results trade-off) as we increase the support (i.e., as we look for the "extremely" popular words).
- Assess how the A-priori algorithm performs (computational time vs significance of results trade-off) on datasets of different sizes.
- Identify which words appear most frequently (e.g., words which can help highlight topics of common interest/common feeling) and assess the evolution of the most frequent words as the war progresses (e.g., solidarity, indifference or anger for the plight of the Ukrainians).

Conclusion: Brief overview of the key results.

Further Works: (Non-exhaustive) list of improvements.

Exploratory Data Analysis

After downloading and uncompressing the zip file from Kaggle, we notice that most data is partitioned by date (i.e. the day tweets were collected). A notable exception to the rule occurs on the 27th of February where:

- Data published (on Twitter) and collected on the February 27th is stored in the "FEB_27.gzip".
- Data published (on Twitter) on the days leading to the 27th of February (i.e., between the 24th of February - day where Russia invaded Ukraine- and the 26th of February) is stored in the "20220227.gzip".

EDA on the 27th of February – schema, exploring text variable

Narrowing our analysis on tweets (which have been created on Twitter and archived) on the 27th of February we highlight the following observations (see Fig 1 to 4) :

Obs. 1: Data has been modelled in the following way (see schema):

- text variable (text of the tweets) and
- metadata (where ts stands for timestamp)
    - on the user (i.e., userid, username, acctdescription, location, usercreatedts, coordinates)
    - on the popularity/activity of the userid (i.e. following, followers, totaltweets)
    - on the tweet (i.e. tweetid, c0,tweetcreatedts, hashtags, language)
    - on the popularity of the tweets (i.e. retweetcount, favourite_count)
    - on the date the tweet was extracted (i.e., extractedts)

```
root
 |-- _c0: string (nullable = true)
 |-- userid: string (nullable = true)
 |-- username: string (nullable = true)
 |-- acctdesc: string (nullable = true)
 |-- location: string (nullable = true)
 |-- following: string (nullable = true)
 |-- followers: string (nullable = true)
 |-- totaltweets: string (nullable = true)
 |-- usercreatedts: string (nullable = true)
 |-- tweetid: string (nullable = true)
 |-- tweetcreatedts: string (nullable = true)
 |-- retweetcount: string (nullable = true)
 |-- text: string (nullable = true)
 |-- hashtags: string (nullable = true)
 |-- language: string (nullable = true)
 |-- coordinates: string (nullable = true)
 |-- favorite_count: string (nullable = true)
 |-- extractedts: string (nullable = true)
```

Fig. 1 – Schema

Obs. 2: Looking at the text column (Fig.2), we notice that

- some tweets are written in foreign languages (e.g., 3rd tweet)
- a strong pre-processing pipeline must remove hyperlinks, emoticons, carriage returns among other characters.

```
---------+------------------------------------------------------------------------------------------------------------------------------------------------
tweetcount|text
---------+------------------------------------------------------------------------------------------------------------------------------------------------
         |#UkraineRussiaWar Captured Russian soldiers\n\n"They said what their own battalion, where they are from and then told him it was just training and did not
 |Like everybody else I am rooting for Ukraine to fight off Russia, but seeing this video is truly sad. These soldiers are clueless and lost and are forced to fight f
      |#Ukrayna Güvenlik Servisi, #Rus finosu #Kadirov'un konvoyunun Gostomel yakınlarında imha edildiğini bildirdi.\n\n#SİHA Reis'e rastlamışlar.. :)\n#UkraineRussi
         |We elected a game show host and got a clown. #Ukraine elected a comedian and got a hero. (Source: Unknown)
      |Ukrainian soldiers wearing sunflowers in their hair\n\n🌻Sunflowers are the national flower of UkraineUA\n\n#Ukraine #Ukrania #UkraineRussiaWar https://t.co/
---------+------------------------------------------------------------------------------------------------------------------------------------------------
```

Fig. 2 – Visualising the text dataset.

Moreover, seeing as we wish to analyse the *significance* of the frequent words, our pre-processing pipeline must also remove stop-words (i.e., frequent "filler" words). As stop-words are generally unique to the language, we need to settle on a subset of homogenous tweets (i.e., tweets written in one language).

Obs. 3: Looking at Fig.3 we notice that around 60% of the tweets are written in English (the remaining 40% are split between the remaining 62 languages). We therefore take the raw dataframe to be the text column (tweets) written in the English language (see Fig.4)

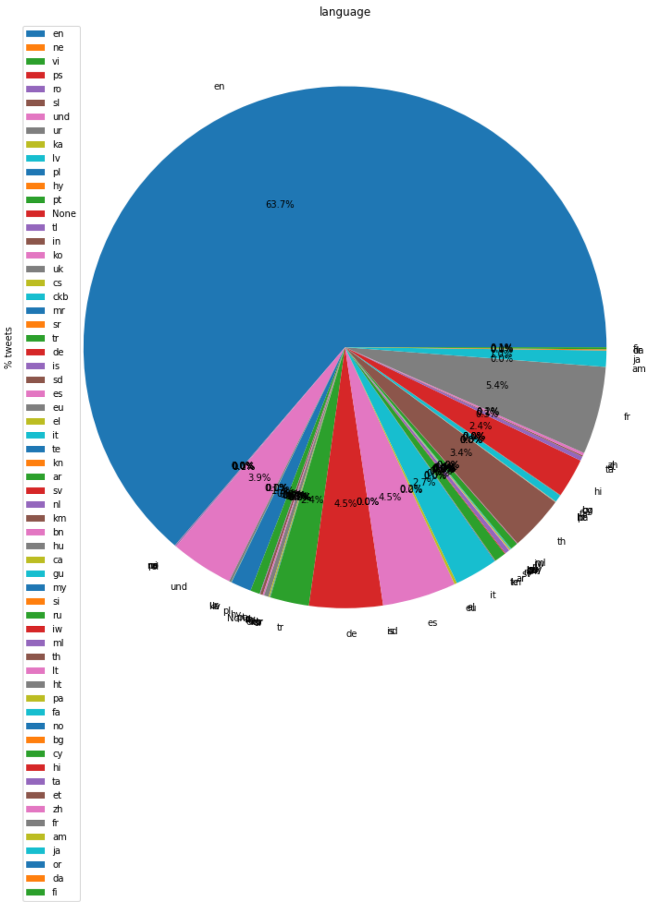

Fig. 3 – Distribution of languages (tweets)

```
+------------------------------------------------------------------------------------------------------------------------------------------
|text
+------------------------------------------------------------------------------------------------------------------------------------------
|#UkraineRussiaWar Captured Russian soldiers\n\n"They said what their own battalion, where they are from and then told him it was just training and did not know they
|Like everybody else I am rooting for Ukraine to fight off Russia, but seeing this video is truly sad. These soldiers are clueless and lost and are forced to fight fo
|We elected a game show host and got a clown. #Ukraine elected a comedian and got a hero. (Source: Unknown)
|Ukrainian soldiers wearing sunflowers in their hair\n\n🌻Sunflowers are the national flower of UkraineUA\n\n#Ukraine #Ukrania #UkraineRussiaWar https://t.co/md7nPCU
|russian troops destroyed a commercial oil base near Vasylkiv.  This is ecological catastrophe not only for Ukraine, but also for the all #Europe \n\n#StopRussia #Sto
|Some scenes from #Kyiv this morning: The building at Lobanovskyi Prospekt hit by a missile at 8am, (no dead), people in shelter (Perry the dog) and firemen trying to
|Meanwhile American "intelligence" pretended an emailed risotto recipe was the greatest national security threat we'd ever faced. 🙃 #UkraineRussiaWar https://t.co/U
|To remind the world: Gaza has lived through 4 wars in 12 years, The children of Gaza after the wars suffer from post-traumatic stress disorder, the world is not fair
|Please pass it on if you can spot the difference. 🤔\n#UkraineUnderAttack #Ukraine https://t.co/lgjRbF9Nvt
|In their coverage @FoxNews actually sound disappointed that #Ukraine is holding off Russia.\n\nThat's where the Republican party is in 2022.\n\nRooting for Russia.
|@KAT96060497 @KChimaev Please, stop involving the man's family. I don't want him to do anything crazy. People of Ukraine are dying and we are close to WW3. Everythin
|@christogrozev My beautiful #Kyiv. This is what we were all afraid of. May God protect #Ukraine and its defenders tonight. #russianinvasion #UnitedWithUkraine
|Long lines at Laika today, a Ukrainian-owned cheesecake &amp; espresso shop in San Antonio, as employees sold 1,400 cheesecake jars &amp; 288 slices of cheesecake. E
|The Int community has displayed their condemnation towards Russia, in light of the devastation in #Ukraine - while aiding and endorsing the genocidal regime in #Ethi
|#UkraineWar #UkraineRussia #StopRussia Anonymous prevent Russian occupiers from communicating https://t.co/UHo9Gfkz4d
|My heart is with everyone suffering in #Ukraine. May there be a fast and peaceful resolution that maintains their right to sovereignty and self-determination UA http
|Ghost of Kyiv locked our Capital's sky. This man have entered 10 battles with russian jets. He's 10-0. THATS SOME COVERAGE. #ghostofkyiv #Legend #Ukranine #UkraineRu
|The sky in Kyiv right now. #Ukraine https://t.co/fGgfycXGbL
|@Xenta777 I STAND WITH #UKRAINE AND UKRAINIAN PRESIDENT @ZelenskyyUa \n\n🌻🌻🌻🌻🌻🌻🌻🌻🌻\n\n#StandWithUkraine \n#PuckFutin #Kyiv https://t.co/WAQgSmRO9g
|We call all hackers and digital activists to be united as one. If this war is not won with weapons, it will be won with cyberweapons. Democracy and freedom will dest
+------------------------------------------------------------------------------------------------------------------------------------------
only showing top 20 rows
```

Fig. 4 – Raw dataframe

Building the Pre-processing Pipeline

For this use-case, the basic pre-processing pipeline (see Fig.5) should include the following elements:

- A document assembler which prepares the data read by Spark into a format that can be processed by SparkNLP
- A Tokeniser which extracts the words and/or symbols from a given tweet.
- Regex filters (removing links, spurious characters and punctuation)
- Stop-word remover

However, when applying this pipeline to the data, we notice that we are still dealing with a very large number of words (items) per tweet (baskets). Applying the a-Priori algorithm to this output would be unnecessarily intensive (computationally) as it could result in a lot of similar frequent words (e.g. soldier vs soldiers) being extracted. In a bid, to extract the most frequent *AND* relevant words, we choose to further reduce the number of words by extracting the meaningful part of words (adding the **Stemmer** SparkNLP element to the pre-processing pipeline). The resulting dataframe, shown in Fig.6, is then passed through a set (to filter out repeated words per tweet) and converted to a Resilient-Distributed Dataset. The resulting RDD is a list of sets, where the sets are the extracted tokens (words) in the given tweets (see Fig. 7 to see a basket).

```
def create_preprocessing_pipeline():
  documentAssembler = DocumentAssembler().setInputCol("text").setOutputCol("document")
  tokenizer = Tokenizer().setInputCols(["document"]).setOutputCol("token")
    #remove links
  linkRemover = Normalizer().setInputCols(["token"]).setOutputCol("tokensWoutLinks") \
  .setCleanupPatterns(["http\S+|www\S+|https\S+"]).setLowercase(True)
      #no whitespace or http (:) #[^\s]--keep spaces  #[^\d]--keep numbers #[^\w]--keep letters
      #remove punctuation
  punctuationRemover = Normalizer().setInputCols(["tokensWoutLinks"]).setOutputCol("tokensWoutLinksAndPuct") \
    .setCleanupPatterns(["&(amp)|@[A-Za-z0-9]+|[^\w ]|_|"])#regex remove all '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
  stopWordsCleaner =  StopWordsCleaner.pretrained('stopwords_en', 'en').setInputCols("tokensWoutLinksAndPuct") \
  .setOutputCol("cleanedTokens").setCaseSensitive(False)
  stemmer = Stemmer().setInputCols(["cleanedTokens"]).setOutputCol("cleanedStemmedTokens")

  return Pipeline().setStages([
    documentAssembler,
    tokenizer,
    linkRemover,
    punctuationRemover,
    stopWordsCleaner,
    stemmer
])
```

Fig. 5 – Constructing the Pre-processing Pipeline

```
+------------------------------------------------------------------------------------------------------------------------
|result
+------------------------------------------------------------------------------------------------------------------------
|[ukrainerussiawar, captur, russian, soldier, battalion, told, train, ukrain, russianarmi, soldier, ukrain, ukrainerussiawar]
|[root, ukrain, fight, russia, video, sad, soldier, clueless, lost, forc, fight, corrupt, dictat, vladimir, putin, dont, ukrainerussiawar, ukraineunderattack, ukrania
|[elect, game, show, host, clown, ukrain, elect, comedian, hero, sourc, unknown]
|[ukrainian, soldier, wear, sunflow, hair, sunflow, nation, flower, ukrain, ukrain, ukrania, ukrainerussiawar]
|[russian, troop, destroi, commerci, oil, base, vasylkiv, ecolog, catastroph, ukrain, europ, stoprussia, stoprussianaggress, ukrain]
|[scene, kyiv, morn, build, lobanovskyi, prospekt, hit, missil, 8am, dead, peopl, shelter, perri, dog, firemen, extinguish, fire, gym, underneath, ukrain]
|[american, intellig, pretend, email, risotto, recip, greatest, nation, secur, threat, wed, face, ukrainerussiawar]
|[remind, world, gaza, live, 4, war, 12, year, children, gaza, war, suffer, posttraumat, stress, disord, world, fair, ukrain, russiaukrainewar, russia, ukrainerussia]
|[pass, spot, differ, ukraineunderattack, ukrain]
|[coverag, sound, disappoint, ukrain, hold, russia, republican, parti, 2022, root, russia]
|[stop, involv, man, famili, dont, crazi, peopl, ukrain, dy, close, ww3, count, saynotowar, ukrainewar]
|[beauti, kyiv, afraid, god, protect, ukrain, defend, tonight, russianinvas, unitedwithukrain]
|[long, line, laika, todai, ukrainianown, cheesecak, espresso, shop, san, antonio, employe, sold, 1400, cheesecak, jar, 288, slice, cheesecak, employe, proce, donat,
|[int, commun, displai, condemn, russia, light, devast, ukrain, aid, endors, genocid, regim, ethiopia, select, outrag, leav, tigrayan, peopl, stuck, genocid, war, tig
|[ukrainewar, ukrainerussia, stoprussia, anonym, prevent, russian, occupi, commun]
```

Fig. 6 – Pre-processed dataframe

```
[['train',
  'captur',
  'ukrain',
  'russianarmi',
  'battalion',
  'soldier',
  'ukrainerussiawar',
  'told',
  'russian'],
```

Fig. 7 – Example of a basket.

The A-Priori algorithm and an overview of the experimental methodology

Please note that in our use-case, we will use the word "item" and "word" interchangeably along with the word "baskets" or "tweets" and the words "input" "whole tweetset", "initial dataset", "initial tweetset" and "transaction-set".

The A-priori algorithm is an iterative technique, used to identify the most frequently occurring (and meaningful) items in a dataset, in which the k-frequent itemsets are used to generate the k+1 frequent-itemsets. Taking advantage of the monotonicity property which states that if a *set* of items is *frequent* then *every subset* (of the set) must also be *frequent*, the algorithm reduces the search space thus the computational time needed to find the frequent 1 to k-uplets items. Indeed, after finding the frequent "k-uplets" (e.g. singlets) the algorithm only relies on this set (and not the whole dataset) to generate combinations of "k+1-uplets" (e.g. doublets) which could be frequent (a.k.a. candidates). It then filters and maps the "generated k-uplets set" to the original transaction-set (1 if the k-uplet element exists in the original transaction-set, 0 otherwise) before performing the count and filter operations.

The algorithm can be represented in the following steps (see Fig.8 - Fig.13)

Input:

- resilient-distributed dataset (RDD) of tweets for a given date (and where language = "English"). An RDD is well suited for scalability purposes as it can benefit from distributed computing (e.g. parallel computation performed on multiple (commodity-hardware) nodes) .
- Support_Threshold (%) or Support: number of times the given item(s) appear throughout the different tweets.

Process:

1. Find the most common singlets:
    - flatMap on x – we get a flattened list of all the words present in the tweets
    - map on (x,1) – we map each word to a key-value pair (word,1)
    - reducebyKey on x+y – we group the values by key (word) and perform the sum of the values (1). We therefore get the frequency of each unique word in the whole tweet-set (input).
    - Filter on x[1]>=MIN_SUPPORT– we filter out all of the words which appear less times than the support value.
- **Output** : frequent_items_and_count_RDD (of the 1-uplets)
2. While the number of frequent (k-1)uplets is greater than 0: *//not_empty*

a. Generate a set of possible candidates: *//get_all_possible_candidates*
    i. Extract the singlets from the frequent (k-1)uplets.
    ii. Use the itertools.combinations(single,n) function to produce the k-length tuples in sorted order (e.g. just take one out of the following items <A,B>, <B,A>) and with no repetition (e.g. no <A,A>).
- **Output** – k-tuple candidate set

b. Filter (on the initial transaction-set) and map (to 1 or 0) the candidates *//count_candidates*
    i. If the k-length tuples are present in the initial transaction set then map on (x,1) else map on (x,0), where x is an element of the k-length tuple.
- **Output** – filtered and mapped k-uple candidates

c. Find the most common k-uplets *//get_frequent_itemsets*
    i. flatMap on the filtered & mapped candidates – we get a flattened list of all the remaining k-uples of words present in the tweets

ii. reducebyKey on x+y – we group the values by key (word) and perform the sum of the values (1). We therefore get the frequency of each unique k-uple words in the whole tweet-set (input).
iii. Filter on x[1]>=MIN_SUPPORT– we filter out all of the k-uple words which appear less times than the support value.

- **Output** – frequent_items_and_count_RDD (of the k-uplets).

Note: in the implementation of the apriori algorithm (see Fig.8), the frequent_items are extracted from the frequent_items_and_count_RDD and appended to the frequent_itemsets_list (a Spark Resilient Distributed Dataset).

Output – frequent_items_and_count_RDD (of the k-uplets).

To provide scalability to the algorithm, we also cache the initial tweet-set, which is used several times by the algorithm, by saving it in the executors' memory. The content is discarded (unpersisted) once all the frequent words (and combination of words) are found (thus no further actions make use of the initial tweetset).

```
def apriori(input_rdd,MIN_SUPPORT):

    tweet_freqRDD=(input_rdd.flatMap(lambda x:x).map(lambda i: (i, 1)).reduceByKey(lambda a,b: a+b))
    ##Frequent_items_and_count_RDD shall be used to save-all the frequent items from 1->k
    frequent_items_and_count_RDD= tweet_freqRDD.map(lambda x: (x[0], x[1])).filter(lambda x: x[1]>=MIN_SUPPORT)
    frequent_items_RDD= frequent_items_and_count_RDD.map(lambda x:x[0])
    n=2
    input_rdd.cache()#taken several times
    frequent_itemsets_list = sc.parallelize([])
    frequent_itemsets_list+=frequent_items_RDD

    while not_empty(frequent_items_RDD):

      candidate_sets = get_all_possible_candidates(frequent_items_RDD, n)
      frequent_items_and_count_RDD = get_frequent_itemsets_and_count(candidate_sets, input_rdd, MIN_SUPPORT)
      frequent_items_RDD = frequent_items_and_count_RDD.map(lambda x:x[0])
      #Save all the frequent items from 1->k
      frequent_itemsets_list += frequent_items_RDD

      n += 1
    input_rdd.unpersist()#once i found all the frequent (1 to k)uples then unpersist
    return frequent_itemsets_list
```

Fig. 8 – Apriori Algorithm

```
[ ]  #Generate a priori algorithm
       #need a while loop
     def not_empty(freq_sets):
         return (freq_sets.count() > 0) and (freq_sets is not None)
```

Fig.9 – Supporting Functions – Apriori Algorithm – Freq_sets is Not_Empty

```
import itertools
def get_all_possible_candidates(frequents,n):#all possible combo.
  frequents=frequents.collect()#all of them
  #Get all of the single items -- remove duplicates
  if (n==2):#receive singlets
    singles = set()
    for item in frequents:#for item
      singles.add(item)#add if unique

  if (n>2):#will recieve n-uplets (split)
    singles=set()
    for itemsets in frequents:
      for item in itemsets: #Get each individual item>
        singles.add(item)

  print(singles)
  #Combine single to form n-uples (no repetition)
  candidates=[set(itemsets) for itemsets in list(itertools.combinations(singles, n))]
  print(candidates)
  return candidates
```

Fig.10 – Supporting Functions – A-priori Algorithm – Get_All_Possible_Candidates

```
[ ]  def get_frequent_itemsets(candidates, itemsets_rdd, MIN_SUPPORT):
        frequent_item_set = itemsets_rdd.flatMap(lambda x: count_candidates(x, candidates)).reduceByKey(lambda a, b: a+b) \
        .map(lambda x: x).filter(lambda x: x[1]>=MIN_SUPPORT)
        return frequent_item_set
```

Fig.11– Supporting Functions – A-priori Algorithm – Get_Frequent_Itemsets

```
#If given candidate (e.g. combo of item) is a subset of the initial transaction set then append 1 -- if freq
def count_candidates(transaction_set, candidates):
    c_list = []
    for itemsets in candidates:
        print(itemsets)
        if set(list(itemsets)).issubset(set(transaction_set)):#if candidate
            c_list.append([tuple(itemsets), 1])
            print(c_list)
        else:
            c_list.append([tuple(itemsets), 0])
    return c_list
```

Fig.12 – Supporting Functions – A-priori Algorithm – Count_Candidates (map and filter)

Looking at Fig. 13 we highlight the main steps:

Input: FILENAME, pre-processing pipeline, THRESHOLD_PERCENTAGE

- Read filename in CSV format into a Spark DataFrame, an immutable set of objects (distributed across nodes in a cluster) on which we can run SQL queries.
- Select the 'text' attribute (the tweets) of the DataFrame and filter the tweets by language, selecting those written in the English language (where).
- Apply the pre-processing pipeline (which has been constructed with SparkNLP) to the dataframe.
- Transform the dataframe to a rdd with no repeated words (per tweet). The resulting rdd will be a list of sets, where the sets are the extracted tokens (words) in the given tweets.
- Run the A-priori Algorithm and save the elapsed time.

Output: frequent_items_and_count_RDD (of the k-uplets).

```python
def generate_results(FILENAME, preprocessing_pipeline, THRESHOLD_PERCENTAGE):

  # Reading and filtering the dataset acc.g to language
  raw_df = spark.read.csv(FILENAME, header=True, escape="\"", quote="\"", multiLine=True)
  filtered_df = raw_df.where(raw_df.language == "en").select("text")

  # Apply preprocessing pipeline to dataframe.
  preprocessed_df = apply_preprocessing_pipeline(preprocessing_pipeline,filtered_df)
  # Preparing the RDD for the a_priori_algorithm
  input_rdd = preprocessed_df.rdd.map(lambda x: (list(set(x[0]))))

  # Defining the threshold
  n_of_baskets = input_rdd.count()
  print('No of tweets are ', n_of_baskets)
  THRESHOLD = math.ceil(n_of_baskets * THRESHOLD_PERCENTAGE)
  print('Min support is ' , THRESHOLD)
  # Print the runtime, return frequent 1-kuple
  import time
  start_time = time.time()
  result = apriori(input_rdd, THRESHOLD)
  end_time = time.time()
  print('Runtime is' , end_time - start_time)

  return result
```

Fig.13 – Complete Implementation – Generate_Results (time elapsed and 1->K frequent-uples)

Selected experiments (description, results, analysis)

We perform two sets of experiments with differing purposes.

Set of Experiments 1

Objective: We wish to select the 'best' support value in terms of trade-off between the significance of the extracted words (to our analysis) and the amount of computational resources used (e.g. elapsed time). This value shall then be used as the 'baseline' support value for the next set of tests.

Process: We select the tweets collected on 27th of February and perform a set of experiments on the dataset by varying the threshold (e.g., from 5-30% of the number of baskets).

Results and Analysis

SUPPORT	SINGLETS	DOUBLETS	TRIPLETS
5	kyiv, people, standwukrain, war, support,militari,anonym	('peopl' 'ukrain') , ('anonym' 'ukrain') ('ukrain','support') ('ukrain' 'putin')	('russia', 'ukrain' 'war')
10	kyiv, peopl, war	('ukrain' 'war'), ('ukrain', 'russia')	
15	Ukrain, Russia, putin, russia	('russia, 'ukrain')	
Common results	Ukrain, Russia,putin	('ukrain','russia')	

Table 1 – Experiment Set 1 (evolution of the frequent words vs Support value)

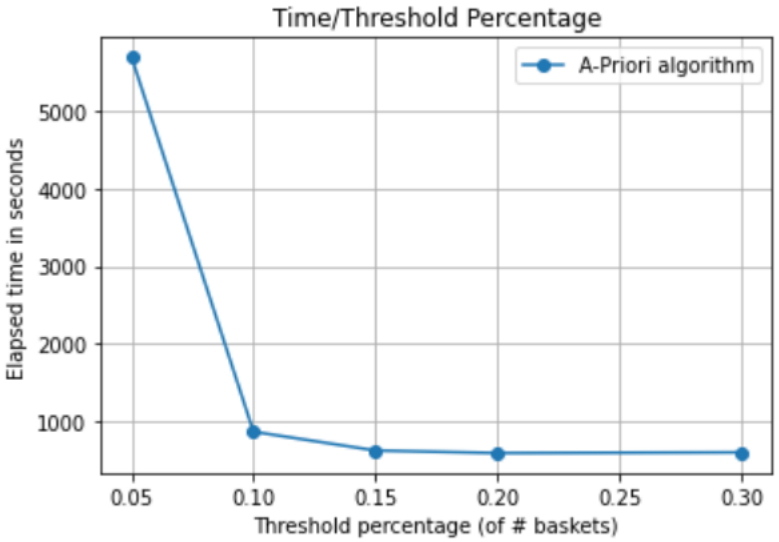

Fig. 14 – Elapsed time vs Threshold percentage

Referring to Table.1 and Fig.14 we note the following observations:

Obs. 1: Some words (such as 'ukrain' and ukrainian along with 'russia' and 'russian') are counted as separate words. Although these words are relevant, they should be treated equivalently (i.e. counted as one word). We therefore highlight the need for a better stemmer in the pre-processing pipeline.

Obs. 2: The significant frequent words which are common to all of the selected support values are "ukrain, russian, putin" w.r.t. singlets and "ukrain, russia" w.r.t doublets.

Obs. 3: As the support increases, the number of singlets (hence n+1-uplets) decrease.

We get rather detailed results with a support threshold of 5% of the total number of baskets, with

- “militari” appearing among the singlets
- “anonym”, “support” appearing alongside “ukrain” as a doublet (and therefore also a singlet)
- and some frequent triplets appearing such as “russia, ukrain, war”.

We notice support for Ukraine, and after a quick Google search (filtering by date) we infer that

- the word “anonym”, appearing among the singlets (and the doublet), refers to the “Anonymous” cyber group declaring cyber-war against Russia on the 27th of February (see https://cybernews.com/news/anonymous-leaks-database-of-the-russian-ministry-of-defence/ , https://greekreporter.com/2022/02/27/anonymous-declares-cyber-war-putin-russia/)

As we can see when the support is set to be 10% of the total number of baskets, we lose some keywords (however, the a-Priori algorithm takes 1/7 th of the time). However, we still keep track of significant details such as the

- location (“kyiv”) and subjects (“people”, “war”) in the singlets.

After a quick Google search (filtering by date) we infer that

- the word “kyiv” refers to the Russian forces advancing on Kyiv on the fourth day of fighting (https://www.theguardian.com/world/2022/feb/27/kyiv-surrounded-says-mayor-fighting-on-fourth-day-of-russian-invasion-of-ukraine, https://www.rferl.org/a/ukraine-russia-invasion-kharkiv-kyiv-fighting-zelenskiy/31725938.html)

When we increase the threshold to 15% or above, we cannot see any *special* frequent words aside from the common results (e.g., putin, russia). Although the computation occurs faster, we do not believe that the reduction in time is significant enough to compensate for the lack of detail.

Conclusion: We therefore select 10% as the “baseline” support value.

Set of Experiments 2

Objective: We wish to assess whether there have been major shifts in (collective) thoughts (e.g. from pro-Ukraine to an anti-Ukraine feeling) and Observe how the most frequent words have changed as the war evolved. We also wish to assess how the algorithm scales up with sample size.

Input - Selected datasets: For a given month we select a date which will result in a collection of initial tweet-sets of differing size. This will allow us to assess how the algorithm scales up with sample size. The initial tweet-sets we choose for the month(s) of

- February, April, and May range from around 227,000- 260,000 tweets (baskets)
- March is around 400,000 tweets (baskets)
- June is around 183,00 tweets (baskets)
- July around 90,000 tweets
- August to October are around 23,00-48,000 tweets.

Input - Selected support: 10% of the total number of baskets (unless specified).

Process: After choosing 9 dates across the February – October period following the 1nce per month period (e.g. 27.02, 07.03, 05.10), we run the a-priori algorithm on each RDD.

Results and Analysis

Mon	Singlets	Doublets
Feb	kyiv,people,war	(ukrain,war),(putin,war)
Mar	stop,war,humanitarian,innoc,civilian,defend,standwukrain	(‘ukrain’,’stop’)(‘ukrain’,’civilian’) (‘provid’,’ukrain’)(‘humanitarian’,ukrain’)(‘putin,’stop’)
Apr	standwukrain,bucha	
May	tigrai	
Jun	tigrai	
Jul	tigrai	
Aug*	militari,biden,standwukrain,russiaisterrorist, china,people	
Sep*	biden,world,mobil,peopl	(‘russia’,’mobil’)
Oct*	standwukrain,kherson,region,nato,forc	
Common	russia,ukrain,putin,war	(‘russia’,’ukrain’), (‘putin’,’ukrain’)(‘ukrain’,’war’)

Table 2– Experiment Set 2 (evolution of frequent words with time)

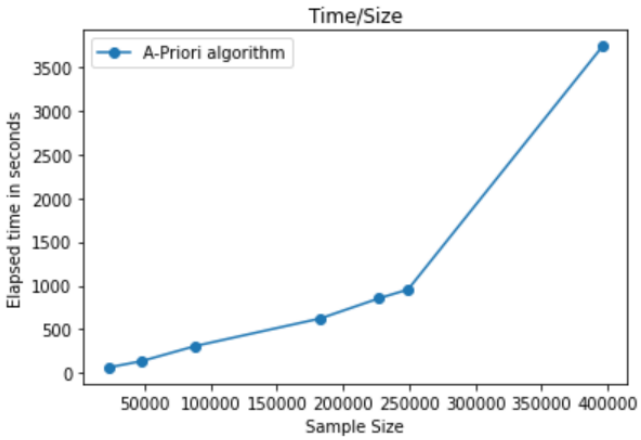

Fig. 15 – Elapsed time vs Sample Size

Referring to Table.2 and Fig.15 we note the following Observations:

Obs. 1: Again, as very similar words count as separate words, we highlight the need for a better stemmer in the pre-processing pipeline.

Obs. 2: The significant frequent words common to all tweet-sets are “ukrain, russian, putin, war” w.r.t. singlets and “ukrain”,” russia” and “ukrain”, war” w.r.t doublets.

Obs. 3: The number of frequent words (no. doublets …) is generally related to the size of the dataset.

Indeed, while we get relatively detailed results for a dataset of around 400,000 baskets (e.g. 6th of March), we find no ‘significant frequent special’ words when running the algorithm on datasets which have less than 50,000 baskets (such as those in August, September and October). However, while the March dataset takes around 1 hour to run, the ones from August-October take no more than 3 minutes. We can therefore run the algorithm with a lower threshold (5% of the number of baskets) to produce relevant and frequent word-sets.

Obs. 4: After analysing the evolution of the frequent words, we highlight the following major trends:

- February: the most common words depict the location of the events (‘kyiv’) and the topics of interest (‘peopl’, ‘war’).
- March: The sense of solidarity to the Ukrainian plight increases, as ‘standwukrain’ becomes a relevant singlet. Moreover, most people seem to be
    - averse to the war, with ‘stop’, ‘putin’ popping up as a frequent doublet
    - angered by the violence (‘ukrain’,’civilian’) and
    - proactive in aiding the Ukrainians (‘provid’,’ukrain’ and ‘humanitarian’,’ukrain’).
- April: The sense of solidarity to the Ukranian plight persists (‘standwukrain’) and a new word ‘bucha’ appears (which refers to the first efforts with regards to evacuation from Bucha Kyiv attempts to negotiate evacuation from Bucha and Hostomel | Ukrayinska Pravda).
- May to July: The outrage with regards to the Ukrainian-Russian war tends to disappear with words such as ‘standwukrain’ falling out of favour. The word ‘tigrai’, which seems to point to the Tigray region, appears as users point to the Ethiopia war and the ethnic cleansing in western Tigray. Google searches tend to suggest that some backlash is developing because of the indifference of the Tigrayan massacre when compared to the Ukraine-Russia war (https://www.aei.org/op-eds/why-do-western-leaders-support-ukraine-but-ignore-tigrays-genocide/)
- August: The interest in the Ethiopian war tends to fade, as the support for Ukraine tends to increase (‘standwukraine’ reappears in the frequent words). Moreover, the Russia-Ukrainian war seems to have fully affected global politics, with ‘biden’ and ‘china’ appearing in the frequent word-set. Google Searches point to Biden’s financial support to Ukraine (https://www.usnews.com/news/world/articles/2022-08-18/biden-administration-readies-about-800-million-in-additional-security-aid-for-ukraine-sources) and China’s support for Russia (https://www.aljazeera.com/news/2022/8/18/china-to-send-troops-to-russia-for-joint-military-exercises).
- September: Words such as ‘biden’, ‘world’ and ‘peopl’ appear in the frequent list along with the (‘russia’,’mobil’) frequent pair (which refers to the escalation of the war and the partial mobilization of the Russian troops https://www.nbcnews.com/news/world/putin-announces-partial-mobilization-russian-military-ukraine-war-rcna48585).
- October – Solidarity with Ukraine keeps up both from a citizen’s-perspective (with words such as ‘standwithukrain’ appearing) and a governmental perspective (with the word ‘NATO’ appearing – see https://www.aa.com.tr/en/russia-ukraine-war/nato-says-it-will-continue-supporting-ukraine-against-kremlins-aggression/2707588). Moreover as the word ‘kherson’

starts to appears we deduce a change in the Russian strategy, i.e. a shift in interest from targeting Kyiv in February to launching an offensive attack in Kherson in October (https://www.euronews.com/2022/10/10/ukraine-crisis-kherson-offensive).

Obs. 5: The A-Priori algorithm does not scale up that well when the number of baskets exceeds 300,000.

Referring to Fig. 15, we note that a 10-fold increase in the number of baskets (e.g., from 23,000-230,000) results in the algorithm taking 15 minutes (instead of 1 minute). Although both times could still be considered reasonable, a 2-fold increase in size from 200,000 to 460,000 results in the algorithm approximately taking 1 hour to run, resulting in a slope (on the elapsed time – size graph) which is at least twice as a big as the previous one. We therefore suggest that, if most datasets of interest exceed 300,000 baskets, the user selects an algorithm which scales-up better with size (e.g. the FP growth tree algorithm).

Conclusion

We conclude by noting the main points relevant to our analysis:

- The A-Priori algorithm scales up quite well for transaction-sets up to 300,000 baskets.
- The optimal support is taken to be 10% of the total number of baskets.
- From the beginning of the war (February) to the end of April, most people stand in solidarity with Ukraine. May to July however see most people highlighting another 'ignored' war (notably the Ugandan war) and expressing their discontent to the Western countries lack of support (the word 'standwukrain' fades off the list). In the months from August to October, the impact of the war seems to have affected key players (with Biden and NATO siding with Ukraine and China with Russia). September sees the mobilization of the Russian troops (action) and the results of such are highlighted in October where we notice the major Russian offensive shifting to Kherson (effect).

Further Works

We list a set of further works which could be of interest:

- Further studies could consider the "trustfulness" of the user (e.g., filtering out any bots by looking at the date of creation, number of followers …).
- Further studies could consider how the evolving frequent word-sets (e.g., common feelings) changes depending on the culture (e.g., the language it is written in).
- Further studies could make use of a better stemmer (to the pre-processing pipeline) and encode the words (items) to hashes (for memory-efficiency reasons). An interested user could then check whether the whole pipeline (from ingestion to results) can be computationally optimized with the A-Priori algorithm (or whether another algorithm should be used).

Appendix

Experiments Set 1

Output: Result_27_02_C005.collect()

['ukrain', 'ukrainerussiawar', 'russian', 'ukraineunderattack', 'putin', 'russia', 'ukrainian', 'kyiv', 'peopl', 'world', 'war', 'russiaukrainewar', 'ukrainewar', 'anonym', 'standwithukrain', 'support', 'militari', 'countri',('russia', 'russian'), ('ukrainian', 'russian'), ('ukrain', 'russian'), ('putin', 'russian'), ('russia', 'war'), ('russia', 'ukrainian'), ('russia', 'ukrain'), ('putin', 'russia'), ('peopl', 'ukrain'), ('ukrain', 'war'), ('anonym', 'ukrain'), ('countri', 'ukrain'), ('ukrain', 'kyiv'), ('ukrain', 'support'), ('ukrain', 'ukrainian'), ('ukrainerussiawar', 'ukrain'), ('ukrainewar', 'ukrain'), ('ukrain', 'ukraineunderattack'), ('ukrain', 'world'), ('putin', 'ukrain'),('russia', 'ukrain', 'russian'), ('ukrain', 'ukrainian', 'russian'), ('russia', 'ukrain', 'war'), ('russia', 'ukrain', 'ukrainian'), ('putin', 'russia', 'ukrain')]

Output: Result_27_02_C01.collect()

['ukrain', 'ukrainerussiawar', 'russian', 'putin', 'russia', 'ukrainian', 'kyiv', 'peopl', 'war',('russia', 'russian'), ('ukrain', 'russian'), ('russia', 'ukrain'), ('ukrain', 'war'), ('ukrain', 'ukrainian'), ('putin', 'ukrain')]

Output: Result_27_02_C015.collect()

['ukrain', 'russian', 'putin', 'russia', 'ukrainian',('ukrain', 'russian'), ('russia', 'ukrain')]

Output: Result_27_02_C02.collect()

['ukrain', 'russian', 'putin', 'russia', ('russia', 'ukrain'), ('ukrain', 'russian')]

Output: Result_27_02_C03.collect()

['ukrain', 'russia']

Experiments Set 2

Feb

```
Result_27_02_C01.collect()
```

['ukrain', 'ukrainerussiawar', 'russian', 'putin', 'russia', 'ukrainian', 'kyiv', 'peopl', 'war', ('russia', 'russian'), ('ukrain', 'russian'), ('russia', 'ukrain'), ('ukrain', 'war'), ('ukrain', 'ukrainian'), ('putin', 'ukrain')]

Mar

```
result_06_03_C01.collect()
```
['russia', 'russian', 'putin', 'ukrain', 'stop', 'war', 'ukrainian', 'innoc', 'provid', 'weapon', 'defend', 'humanitarian', 'civilian', 'peopl', 'ukrainerussianwar', 'standwithukrain', ('ukrain', 'russian'), ('russia', 'ukrain'), ('stop', 'ukrain'), ('putin', 'stop'), ('ukrain', 'civilian'), ('provid', 'ukrain'), ('humanitarian', 'ukrain'), ('putin', 'ukrain')]

Apr

```
result_07_04_C01.collect()
```
['peopl', 'ukrain', 'russia', 'war', 'standwithukrain', 'ukrainian', 'bucha', 'russian', ('ukrain', 'russian'), ('russia', 'ukrain'), ('ukrain', 'ukrainian')]

May

```
result_16_05_C01.collect()
```
['ukrain', 'ukrainian', 'russia', 'russian', 'tigrai', ('ukrain', 'russian')]

June

```
result_10_06_C01.collect()
```
['war', 'ukrain', 'ukrainian', 'russia', 'russian', 'tigrai', ('ukrain', 'russian'), ('russia', 'ukrain'), ('ukrain', 'ukrainian')]

July

```
result_22_07_C01.collect()
```
['russia', 'ukrain', 'war', 'russian', 'putin', 'ukrainian', 'standwithukrain', 'world', ('ukrain', 'russian'), ('russia', 'ukrain')]

August

```
results_08_18_C_01.collect()
```
['ukrain', 'russia', 'russian', ukrainian, war, ('russia', 'ukrain')

```
results_18_08_C_005.collect()
```
['ukrain', 'russia', 'russian', 'militari', 'biden', 'putin', 'standwithukrain', 'russiaisaterroriststst', 'ukrainian', 'war', 'china', 'russiaisateroriststst', 'peopl', 'ukrainerussiawar', ('russia', 'russian'), ('ukrain', 'russian'), ('russia', 'ukrain'), ('ukrain', 'war'), ('ukrain', 'ukrainian')]

September

```
results_21_09_C_01.collect()
```
['ukrain', 'war', 'putin', 'russia', 'russian', 'ukrainerussiawar', ('russia', 'ukrain'), ('putin', 'russia'), ('putin', 'ukrain')]

```
results_21_09_C_005.collect()
```
['ukrain', 'war', 'putin', 'russia', 'biden', 'russian', 'ukrainerussiawar', 'ukrainian', 'world', 'mobil', 'peopl', ('russia', 'russian'), ('ukrain', 'russian'), ('putin', 'russian'), ('russia', 'war'), ('russia', 'mobil'), ('ukrainerussiawar', 'russia'), ('russia', 'ukrain'), ('putin', 'russia'), ('ukrain', 'war'), ('putin', 'war'), ('ukrainerussiawar', 'ukrain'), ('putin', 'ukrain'), ('putin', 'russia', 'ukrain')]

October

```
results_05_10_C_01.collect()
```
['ukrain', 'war', 'russia', 'russian', 'putin', 'ukrainian', ('ukrain', 'russian'), ('russia', 'ukrain')]

```
results_05_10_C_005.collect()
```
['ukrain', 'kherson', 'standwithukrain', 'war', 'russia', 'region', 'ukrainerussiawar', 'russian', 'forc', 'nato', 'ukrainewar', 'putin', 'ukrainian', 'world', ('russia', 'russian'), ('ukrain', 'russian'), ('putin', 'russia'), ('russia', 'ukrain'), ('ukrain', 'war'), ('ukrain', 'ukrainian'), ('putin', 'ukrain')]