Belcao Matteo – 10503336

USDE optional project: final report

1. Overall Description

The project work consists into analyzing some tweets about the Brexit topic, starting from some information about the tweets stored in a CSV database available at https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/KP4XRP.
Using this information in the pre-proccessing phase to locate a subset of users and their related tweets, for each tweet a python script will extrapolate the most salient words applying filtering and transformations in the middle steps of the elaboration, and will store the results (single-tweet and user-aggregate arrays of tuples <word,count>) in a MongoDB database.
Finally, with different programs written in python will be analyzed the most frequent used words for the English tweets and plotted several graphs that takes into account different parameters like stance, sentiment, language.
Tweets in the 4 main European languages (IT,FR,DE,ES) will also be analyzed, but in this case the analysis will be limited to describing the most used words for each one.

2. Analysis Stages

2.1 Preliminary Operations

Since the dataset is huge, I decided to focus on users who haven't written too many tweets on the brexit topic and don't seem to be fake accounts.
After a preliminary analysis using the python notebook *explorartive_analysis.ipynb*, have been selected the users (and their related tweets) that respects the following parameters:

- pretty sure that are humans : bot_score < 0.5 :
- a clear opinion on the topic : keep only users with "leave" or "remain" stance
- not excessive tweeters : deletion of outliers that made a number of tweets above the 0.90 quantile

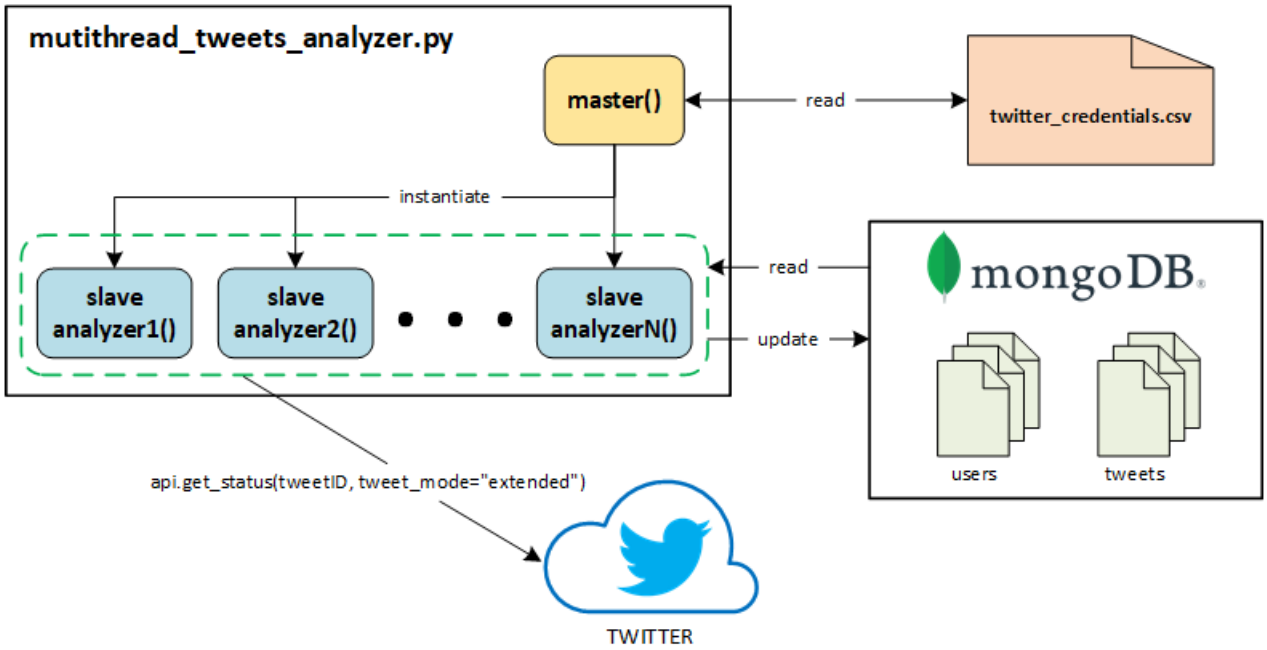
After these preliminary operations, remains about 500K users (28% "leave" and 72% "remain", each of them with 1 to 5 written tweets), which will be saved in the *users_filetered.csv* file.
The tweets are also filtered, and in this case only those written by the remaining users are kept and stored in the *tweets_filtered.csv* file.

2.2 MongoDB set-up

It's created a new database "brexit", then the two files *users_filetered.csv* and *tweets_filetered.csv* are imported into the DB as two different collections.
In order to check the DB updates performed in the next phase, I've decided to use the program "MongoDB Compass", which has a GUI that allows to easily explore the dataset.

2.3 Salient Words Extraction

Now, for each tweet we have to extract the salient words, and the script in the file *multithread_tweets_analyzer.py* does this job.
First of all it starts a *master()* that reads from a csv file all the credentials keys of the multiple accounts that interacts with the twitter API, and then for each one instantiates a thread with the *slaveTweetsAnalyzer()* method that contains the analysis code.



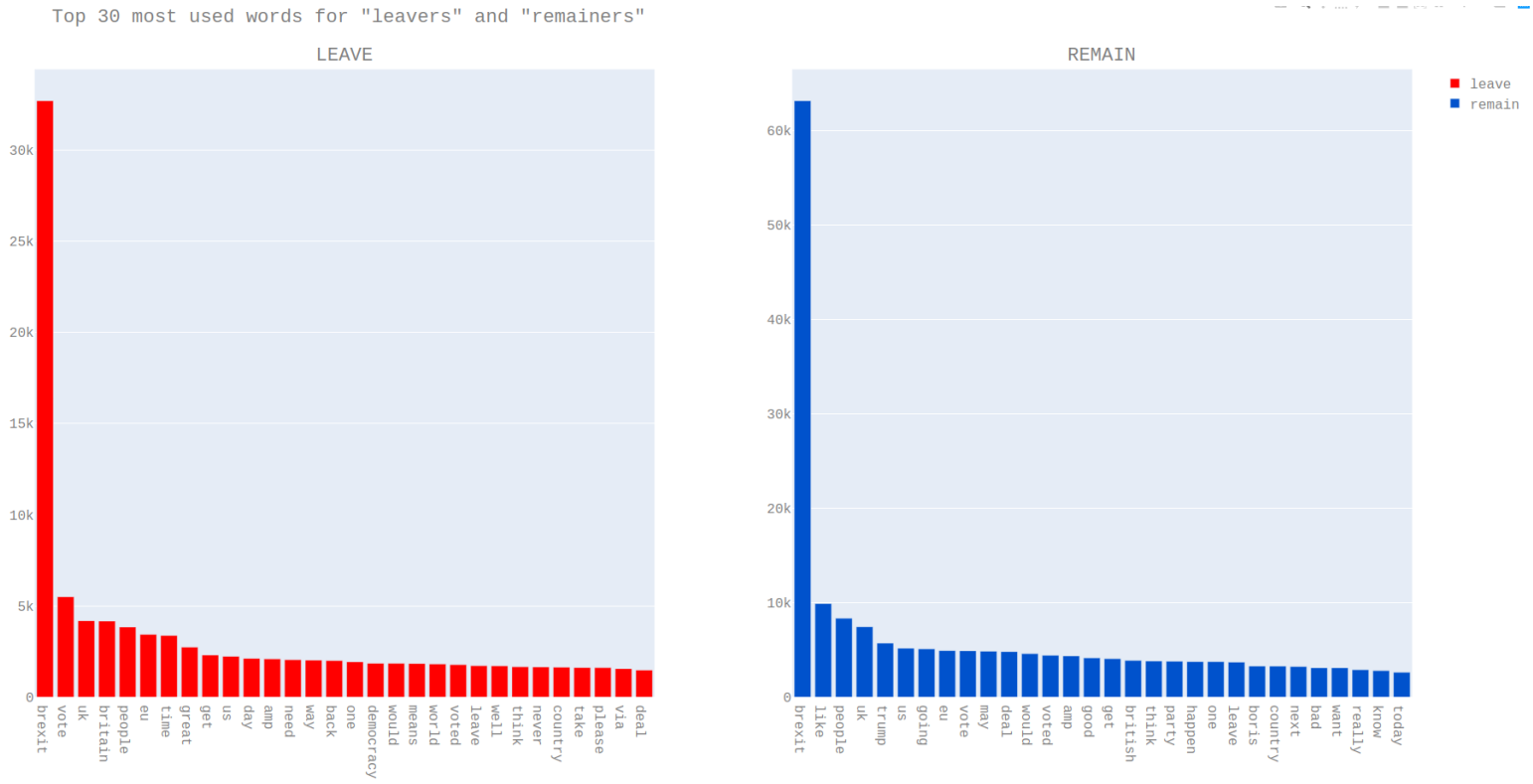
Each *slaveTweetsAnalyzer()* instance will randomly retrieve from the MongoDB "users" collection one id of a user that hasn't been analyzed yet (no salient word in the document related to the user) and the related tweets IDs in the MongoDB "tweets" collection.
Then, for each user's tweet is fetched the related text using the tweepy library, that is a simplified way for interacting with the twitter REST APIs,
Only if the tweet has a language that is within the scope of the analysis (EN, FR, IT, ES, DE) is considered: the text is stripped from undesired characters and URLs, then its words are splitted and removed if stop-words (using NLTK library), and finally the occurrences of each word are counted and inserted in an array composed of tuples <word,count>.
The 5 most salient words and the tweet language are added to the related "tweet" document using a MogoDB update operation. In case of ties are kept all the words with the occurrence equal to the 5th.
Finally, when all the user's tweets have been analyzed, all the tweets counters are summed up in order to compute the aggregated word count of the user, and the 5 most salient words are added in the related document in the "users" collection with a MongoDB update operation. Also in this case in case of ties are kept all the words with the occurrence equal to the 5th. It's also added the primary language of those tweets, that is the most common used language computed by majority-voting.

When all the results are put in the DB, a new random user that hasn't been analyzed yet (no array of word counts in his document) is picked from the "users" collection, and the iteration restarts.

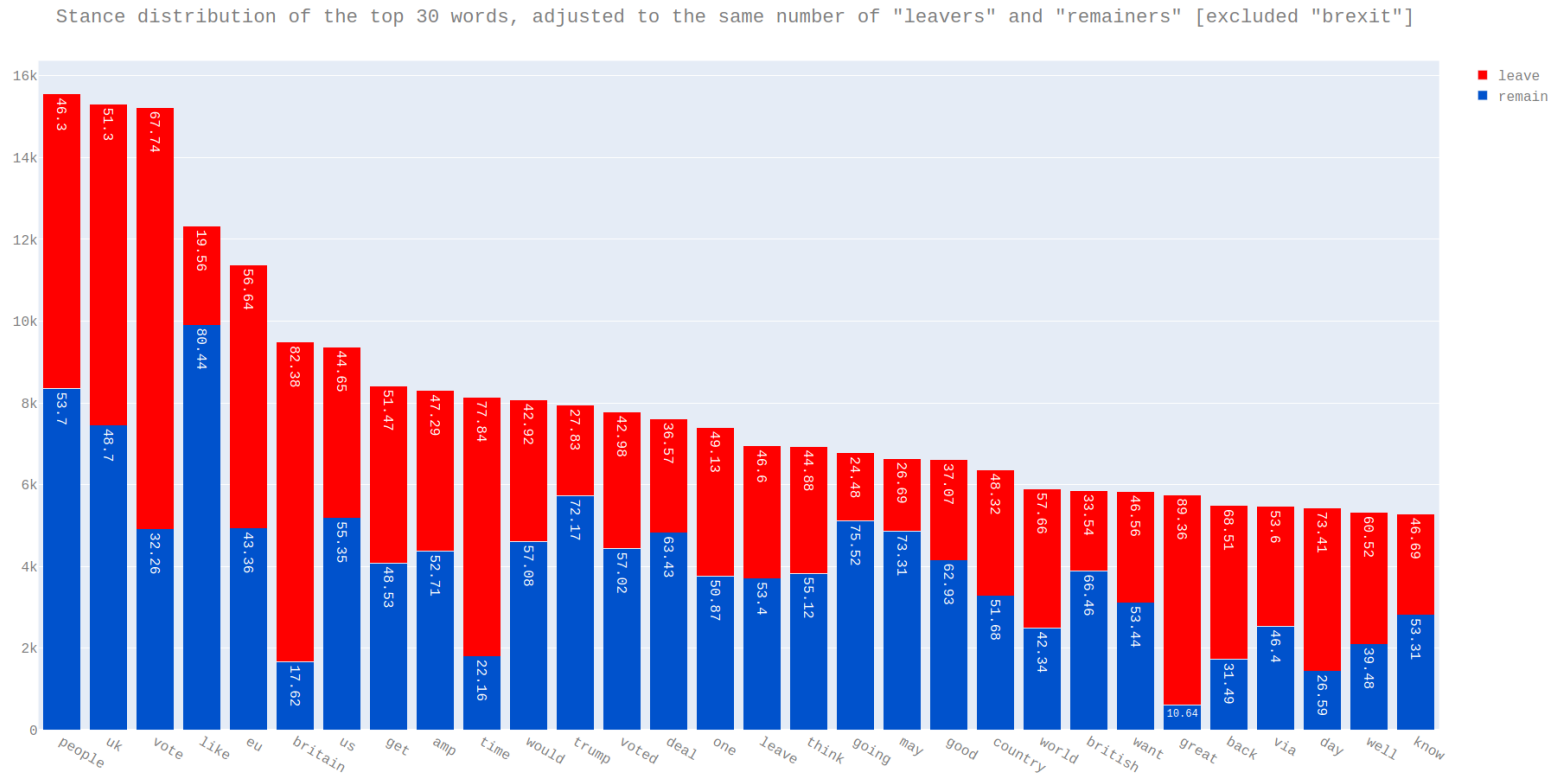
2.4 Data Analysis and Graphs Plot

When the “user” and “tweets” collections are enough populated, it can be started the analysis phase that is performed some MAP-REDUCE queries with MongoDB.
I’ve decided to use the MAP-REDUCE approach because allows to achieve better performances respect to using a python script (>150K tweets words vectors analyzed in less than 30 seconds) and to write query codes that allows to split the counts based by stance (“leave” or “remain”), by sentiment (“positive”, “neutral”, “negative”) and by language (EN , FR, DE, IT, ES).
The main analysis regards the most used words in English tweets, and has been made from both the "tweets" and "users"(aggregate data) MongoDB collections, but in order to be more concise in the next pages are reported only the ones obtained from the "tweets" collection, because the results are asymptotically identical (the "user" collection contains aggregated and approximate data from of the "tweets" collection).

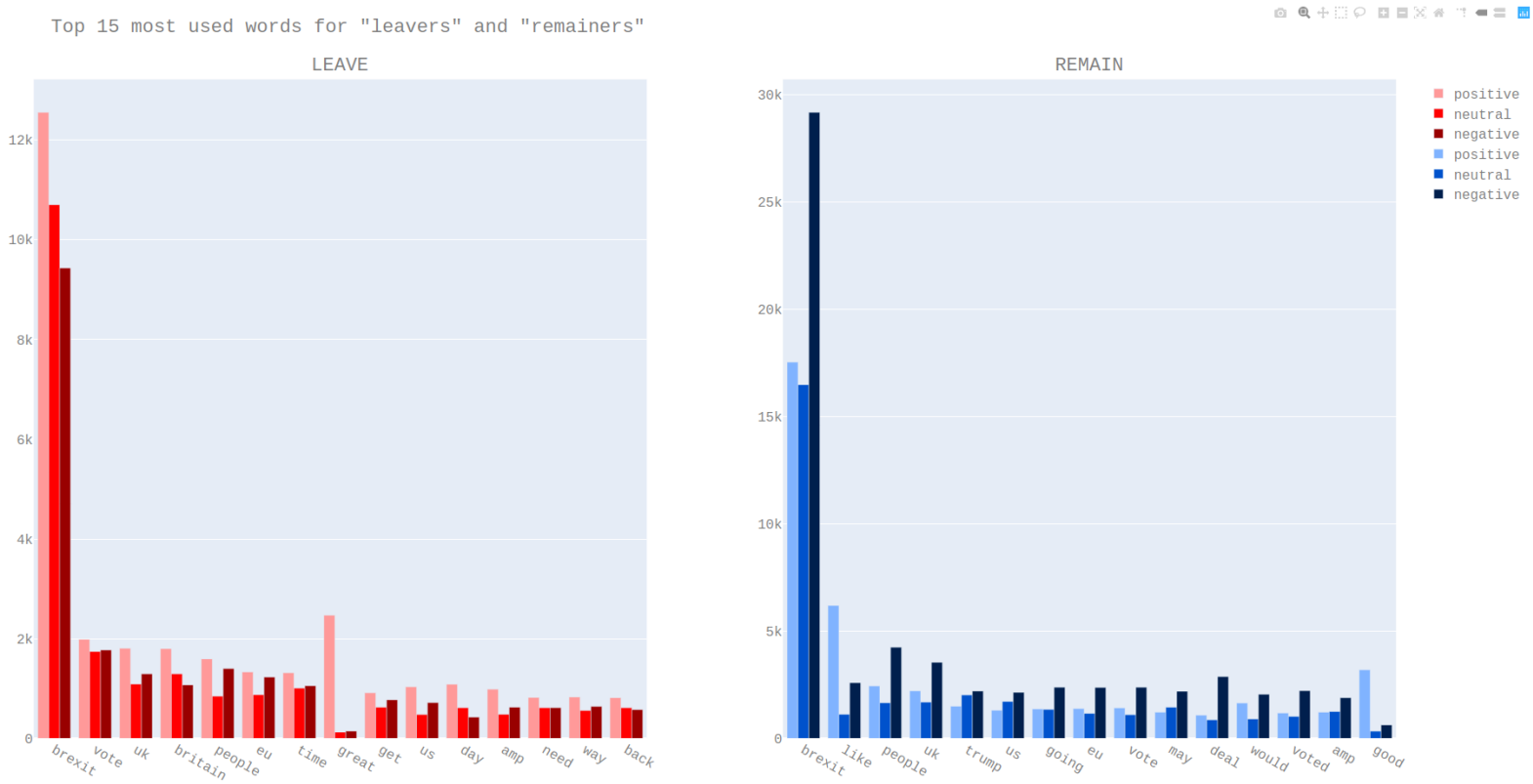
FIRST ANALYSIS: plots of the most used words used in the analyzed tweets, taking into account the tweet stance. Script name: *LeaveRemainTweetsAnalysis_EN.py*

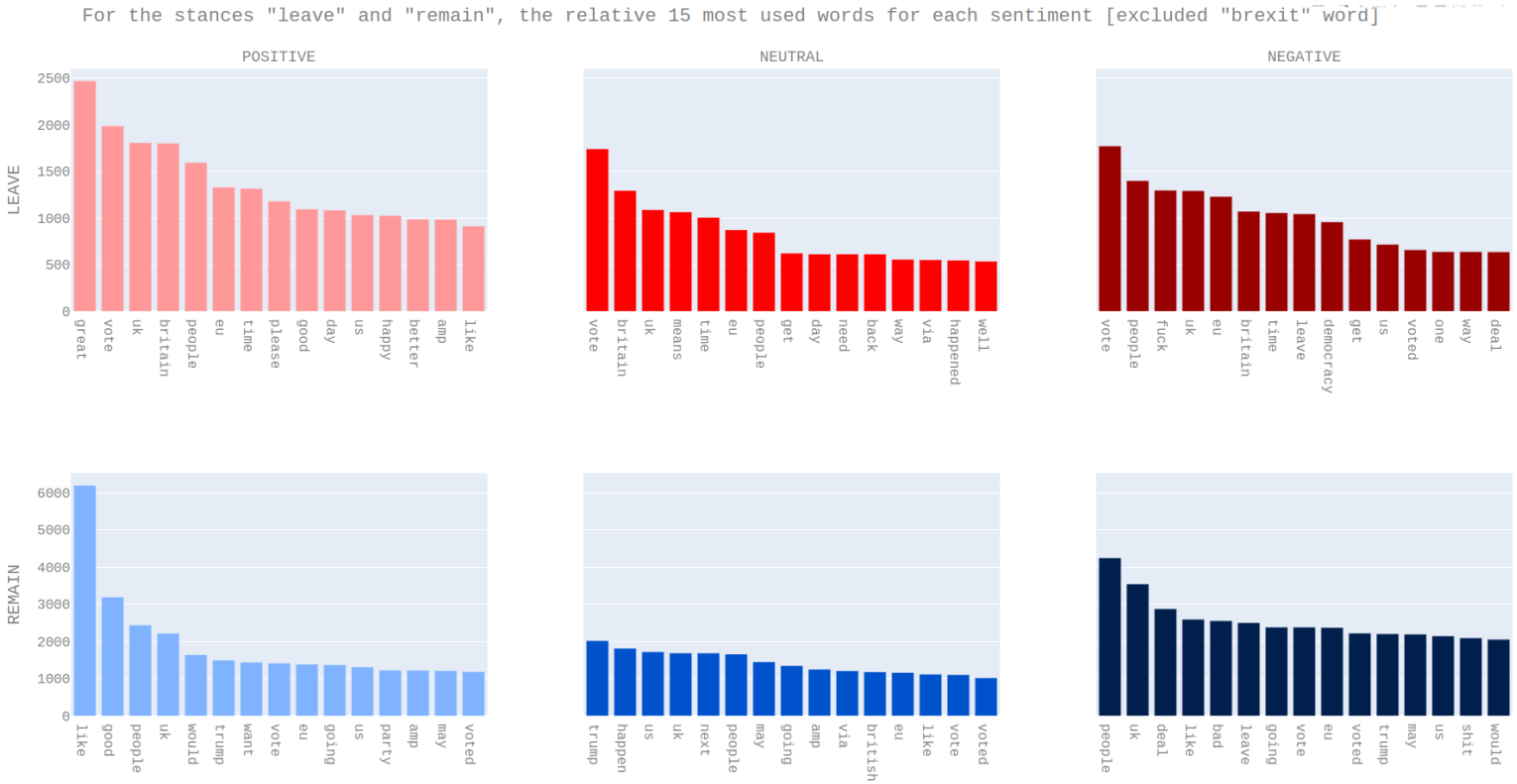


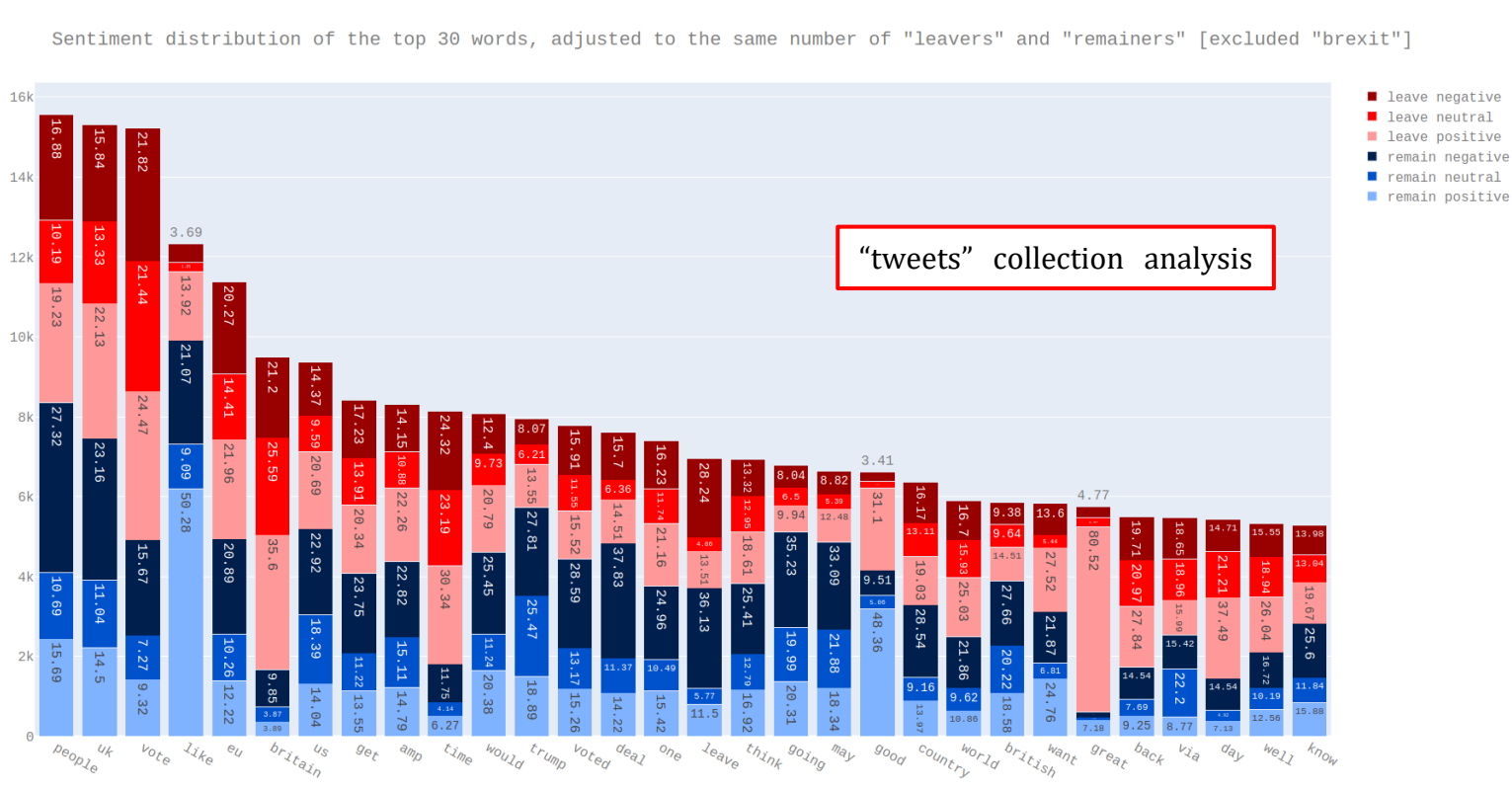
Observation : Since during the preliminary analysis of the users I have noticed how the number of users with "remain" stance is unbalanced respects to "leave" stance (28% vs 72%), I decided to compute on real-time an adjustment factor that basically is a proportion between the number of tweets analyzed belonging to the two stances.



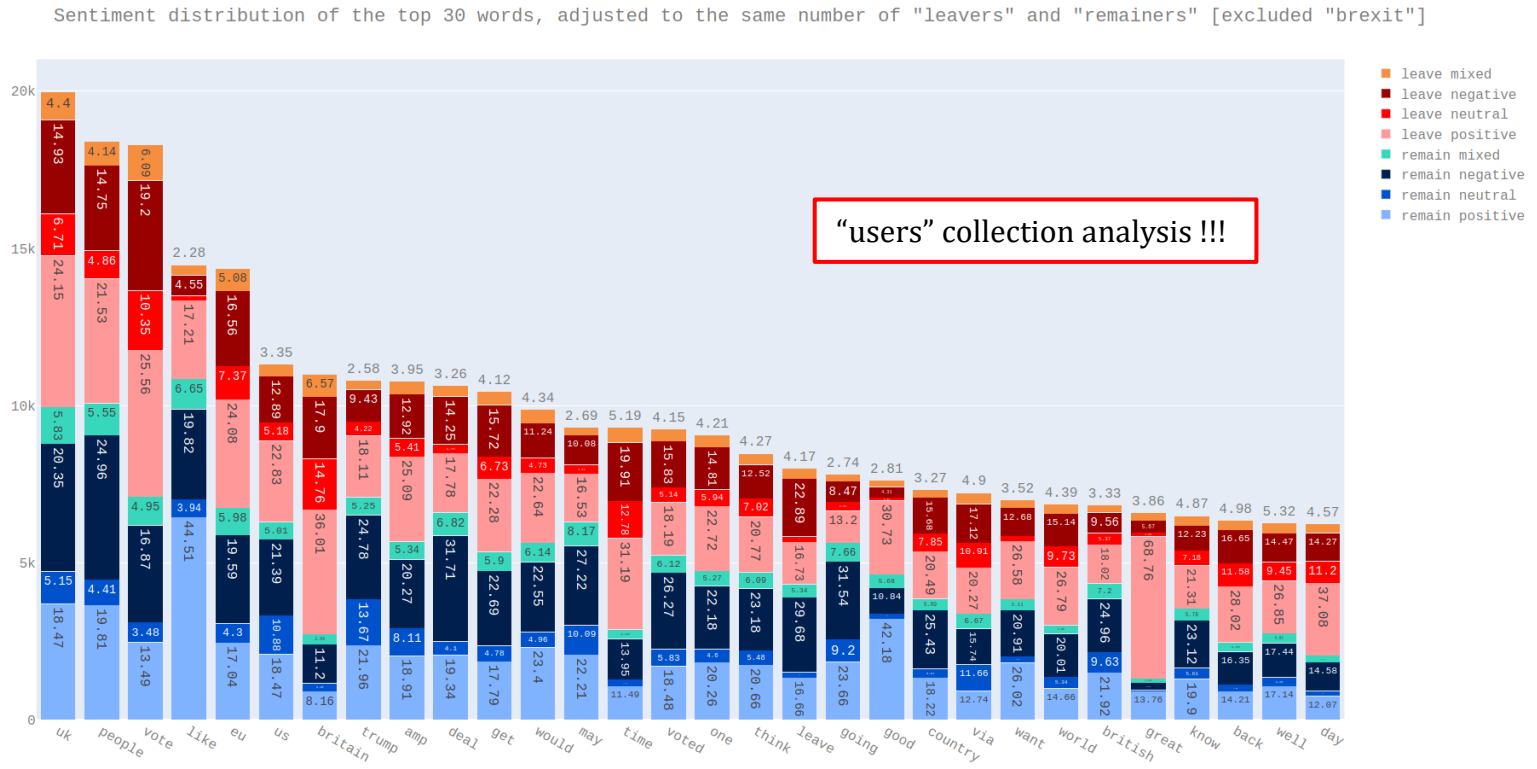
SECOND ANALYSIS: plots of the most used words used in the analyzed tweets, taking into both tweet stance and sentiment. Script name: *LeaveRemainTweetsSentimentAnalysis_EN.py*



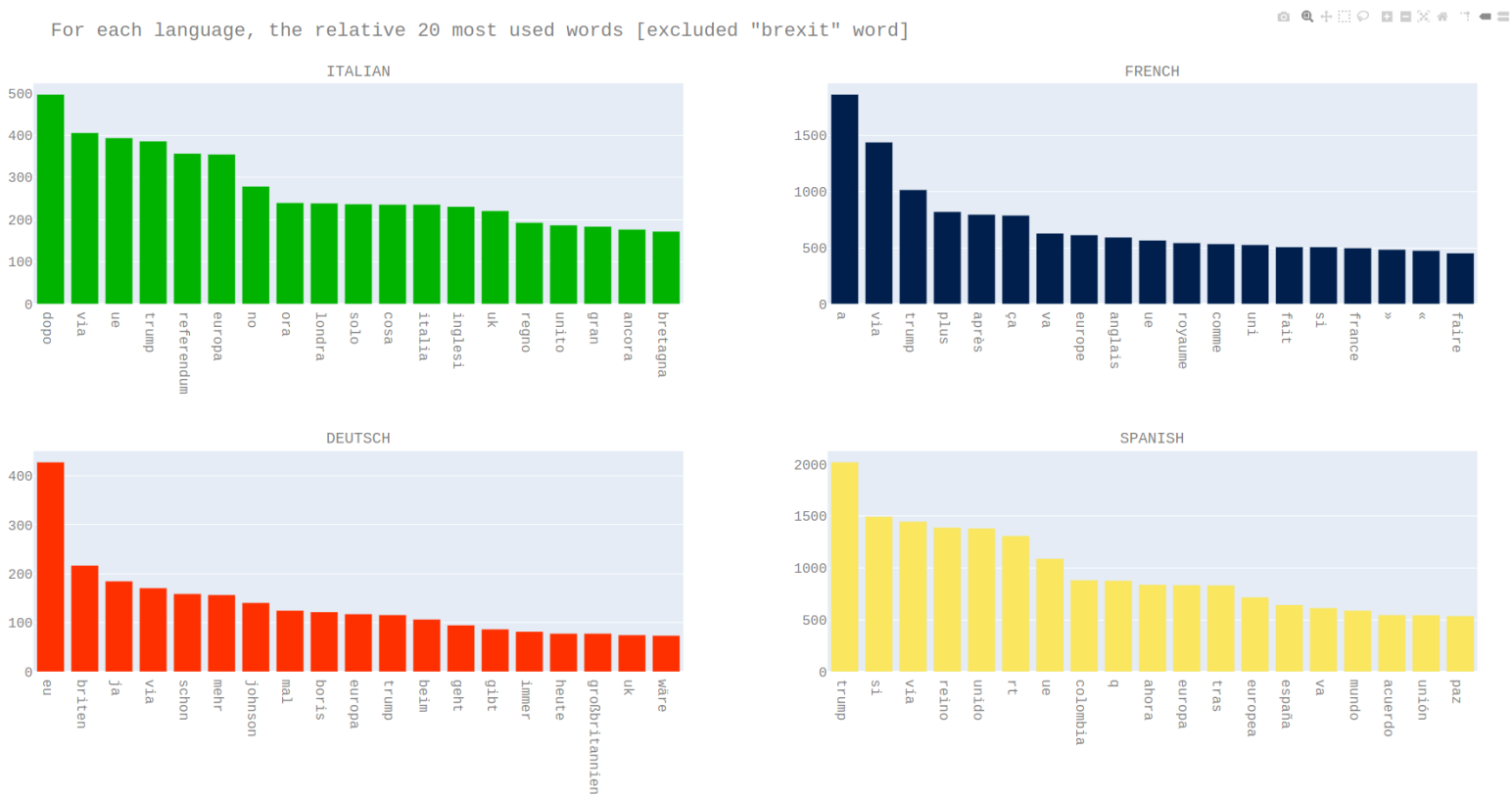




Observation: we can see that the percentages of the above graph that analyses tweets documents doesn't differ so much respect the below one that analyses users documents (in which the aggregate word counts are cropped)



THIRD ANALYSIS : plots of the most used words used in the major European languages (excluded English). Script name: *MostUsedWords_IT_FR_DE_ES.py*



Unfortunately, in this case it was not possible to distinguish between “leave” and “remain” or sentiment, because almost all the tweets had "remain" stances, probably due to a wrong classification of non-english tweets using english words as evaluation metric.

3. Conclusions

The various development phases of this project have been very challenging, and in particular from this experience I gained a little bit of practical experience on managing the following technical aspects:

- Execution of a simple explorative analysis on the initial data in order to select a "nice" subset of the initial data according to some criteria.
- Text extraction and analysis: using a reduced NLP pipeline, it was possible to start from raw tweets text and finally obtaining the word-counting tuples stored in the MongoDB database.
  In particular, using the script *multithread_tweets_analyzer.py*, the overall requests throughput using 4 parallel accounts has been around *10K requests/h*.
- Management of missing values: some users that were stored in the DB, nowadays are not still present on twitter, so it has been needed to handle the API exception error. The same problem has affected several tweets, which turned out to have been deleted or without text (e.g. retweets of famous people tweets, web links).
  Just to give an idea: in the tweets database we have around 200K non-null tweets over 450K API requests.
- Learn how to install and use in practice the MongoDB database and the pymongo library to interact with it inside a python script
- Execution of MAP-REDUCE queries, in order to obtain the word counting based on different parameters (stance, sentiment, tweets languages)
- Data visualization with some bar-plots in order to show the analysis outcomes