USDE PROJECT REPORT

Analysis of data from tweets related to Brexit.

Introduction

Brexit has been a polarizing political event; in this work, we try to show what is the extent of this opposition, starting from data coming from tweets that were related to the keyword "Brexit".

The objective is to display how the interactions between users on twitter relate to their political stance, and to see which are the main hubs in discussions regarding the Brexit topic.

Results show that the groups of people that match the political stance "Remain" and those that match the stance "Leave" indeed form communities in the graph theory meaning of the word.

Approach

To study these phenomena, data was acquired from a dataset of tweets that linked tweets with stance, sentiment, and other data. This data was then imported in graph format in a Neo4J database. Visualizations were finally computed using the NeoVis library.

As data coming from the dataset was incomplete, scraping was used on different websites to retrieve the missing information.

Data acquisition and pre-processing

Twitter dataset about Brexit

A large portion of the data has been retrieved from the "Twitter dataset about Brexit" from Harvard dataverse[1].
The dataset contains a list of tweet identifiers and relative user identifiers for tweets that matched the keyword "Brexit". Together with the identifiers, for each tweet and for each unique user a sentiment and stance score was also provided.

This data has been lightly pre-processed to solve a few issues:

1. Files where originally tilde separated and provided no headers, they have been converted to a standard format and headers were generated.
2. Stance was not consistent for tweet and for user files, as in the former "other" stance was used but, in the latter, "others" was used instead.

Tweet text scraping

Because the data from "Twitter dataset about Brexit" did not contain tweet text but only their ID, scraping was required to obtain it. The process has been performed with a purpose-built scraper that is able to retrieve, given the id of a tweet, both text of the tweet and tag of the tweeting user. If the tweet was deleted or unavailable, text is left null for the specific row.

Text data from tweets has been used to find both mentions and hashtags by means of regular expression over the tweet text. Two files were then created starting from this data, one that links tweet IDs to each mention in it and another that links them to each hashtag used in the tweet.

Tweet user ID scraping

Text scraping allowed to link each tweet to the user tag of the users that the tweet mentions, however, in our dataset, users are referred to by ID. To link user at-tags and their ID, a second purpose-built scraper was used, this time sending requests to gettwitterid[2]. This website offers a free service whereby inputting a user tag, the service returns his/her ID, however, the service does not offer an API, so scraping had to be used.

Finally, this data allowed for the creation of a file that links each user to their tag, albeit deleted users' id were not retrievable using the service, so their ID was set to null in the file.

User stance

To improve performance in subsequent steps, a file was created by looking up in "*user stance sentiment botscore*

[1] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/KP4XRP

[2] http://gettwitterid.com/

tweetcounts” only for users that were author of a tweet present in the “*withText*” file.

Resulting dataset

The final dataset contains the following files:

File	Description
withText	File containing tweets and information about them, including their text if available.
users	Standardized version of user_stance_sentiment_botscore_tweetcounts
userStance	Lookup table for user stance indexed by user id for users author of a tweet in “withText”
mentionedUsers	File containing data retrieved from gettwitterid[2] including mentioned users’ tags
mentions	List of all mentions of each tweet
hashtags	List of all hashtags of each tweet

Graph model

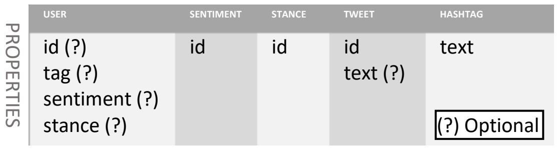
Import into Neo4J

Data was imported into Neo4J via a series of import queries. Relation data is almost completely intrinsic to the dataset and little further processing had to be performed. The resulting graph model is shown in the image.

Hashtags were all transformed into their lowercase form before importing.

Sentiment, stance, user, and tweet nodes are addressed by ID, while hashtag nodes are addressed by their text property.

In addition to “HasSentiment” and “HasStance” relationships, matching properties have been applied to User nodes to aid the process in the visualization tool usage downstream.



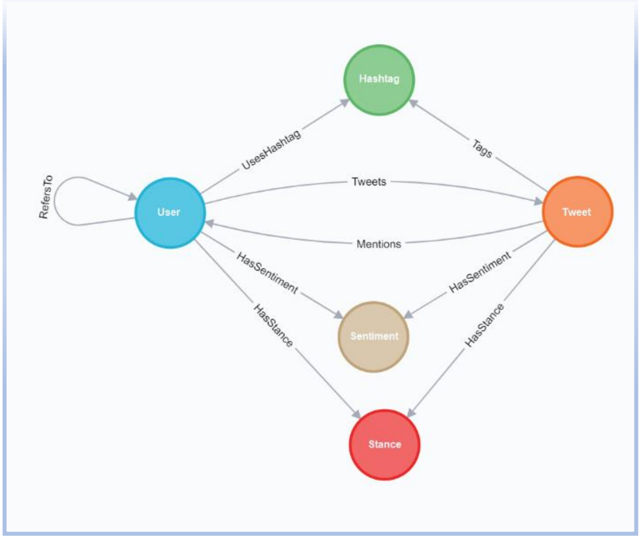
	USER	SENTIMENT	STANCE	TWEET	HASHTAG
PROPERTIES	id (?) tag (?) sentiment (?) stance (?)	id	id	id text (?)	text (?) Optional

Derivate relationships and properties

To obtain a map of connections between tweets, two queries were run, one to link users with the relationship “RefersTo” for each user that writes a tweet which mentions the target user, and another for the relationship “UsesHashtag”, computed by matching all users that authored a tweet that tags that specific hashtag.

A numeric property was added to user nodes, which directly maps an integer to its stance. This was because the visualization tools required an integer property to define communities and it was not able to recognize a string property correctly for the same purpose.

Finally, in-degree was computed for nodes of type “User” for relationship “RefersTo” to identify hubs in the network.



Analysis and visualizations

In the dataset, most of the data points for users and tweets are classified as “other” stance and “neutral” sentiment. Analysis was hence focused only on the samples that were classified as “leave” or “remain” stance to provide a result over the samples that were more polarized.

Because of limits in computational power, analysis was limited to the first 500’000 tweets in the dataset.

Visualization was performed using “neovis.js”; an open source library that builds upon “vis.js” and implements Neo4J integration.

The visualizations that are reported here are available in full resolution format in the additional material.

Referring between communities

The result is a graph of user and “RefersTo” relationships between them. Shown users are the users that have polarized stance and the users they refer to.

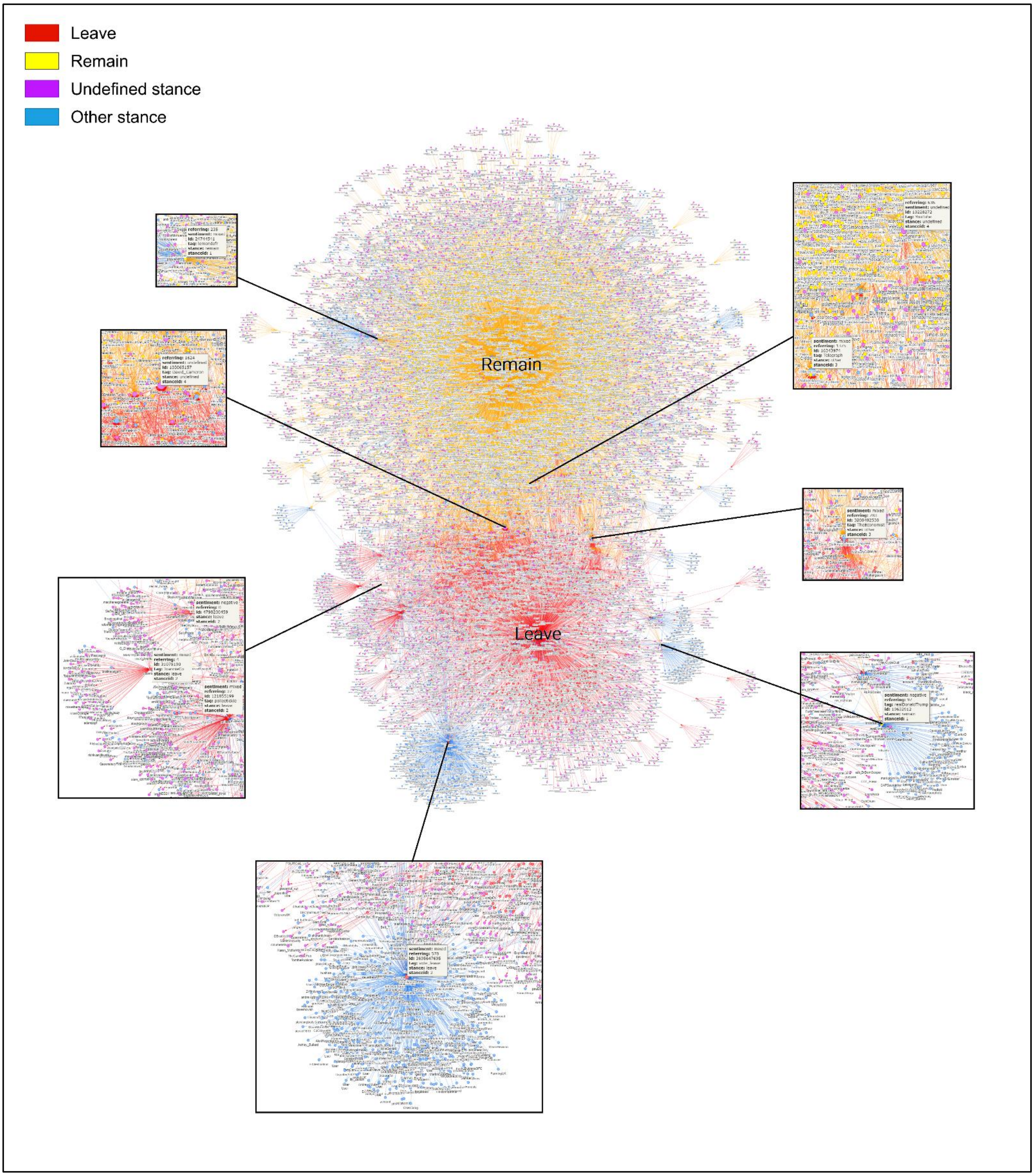
Size of the nodes represent how many users refer to that specific user in the database, this statistic also includes nodes not visible in the graph (In-degree of relationship “RefersTo”).

Color of nodes represents their class, and within the same class (User) color difference is used to demark between stances.

Note that relationships had to be also explicitly returned as the visualization tools needs the references to display them.

The Cypher query used to retrieve the data was:

```
MATCH (n:User)-[r:HasStance]->(m:Stance)
WHERE m.id = 'remain' or m.id = 'leave'
MATCH (n)-[h:RefersTo]-(k)
RETURN n,m,r,h,k
```



Hashtag usage between communities

The result is a graph of users, hashtags and “UsesHashtag” relationships between them. Shown users are the users that have polarized stance and the hashtags they use in their tweets.

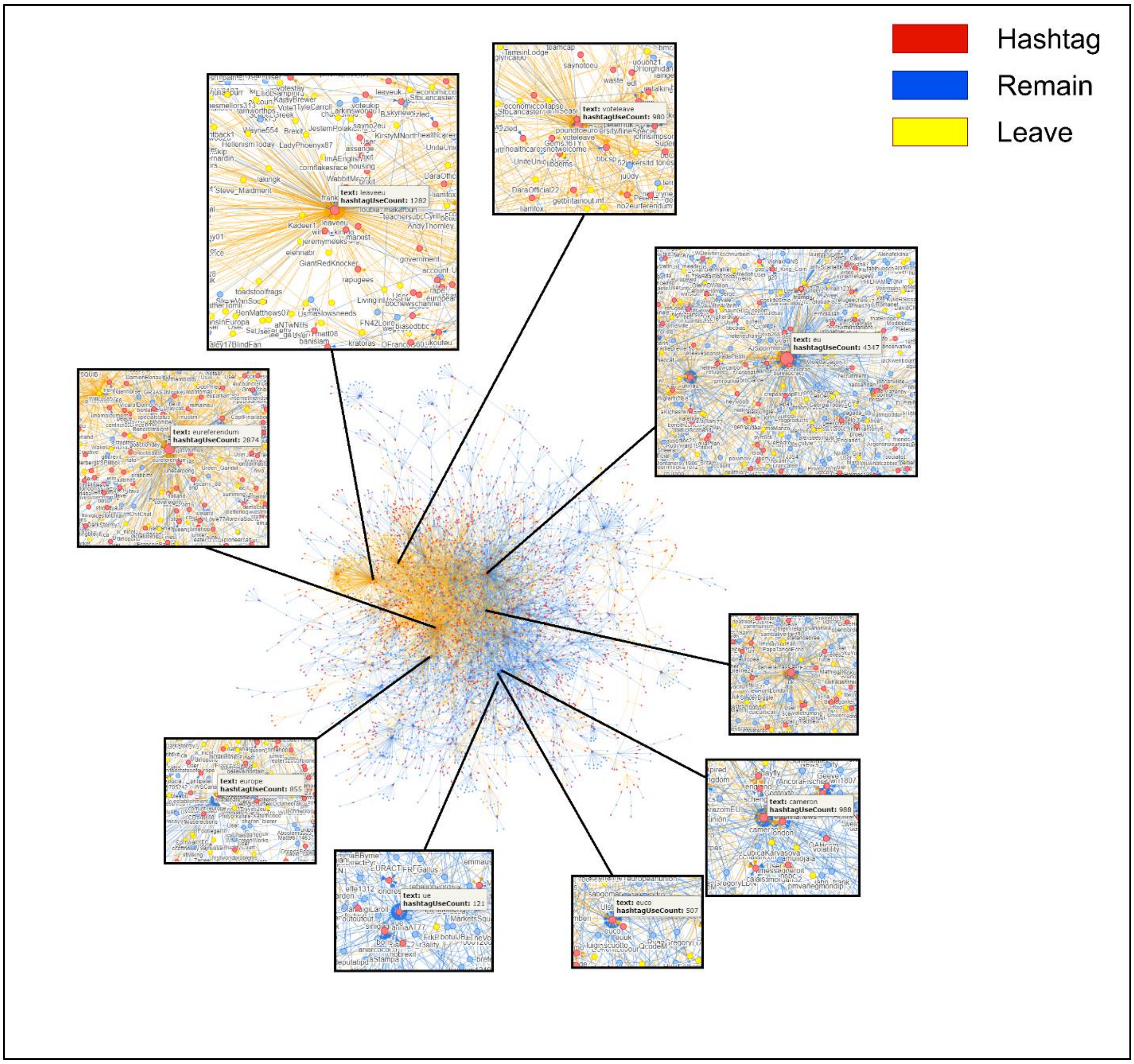
To reduce clutter, hashtag “Brexit” and all its upper and lower case variants were removed as it was too prominent and not significative for the analysis. Size of the nodes represent how many users refer to that specific user in the database, this statistic includes also nodes not visible in the graph (In-degree of relationship “RefersTo”).

Color of edges represents their class, and within the same class, node color difference is used to demark between stances.

The query was limited to hashtags that had at least two users connected to them to restrict the result to the connected component of the subgraph (removing stance nodes but including all users).

The Cypher query used to retrieve the data was:

```
MATCH (a:Hashtag)<-[r:UsesHashtag]-(:User)
WITH a,count(r) as cnt
WHERE cnt > 1 and a.text<>'brexit'
MATCH (a:Hashtag)<-[r:UsesHashtag]-(u:User)
WHERE u.stance = 'remain' or u.stance = 'leave'
MATCH (u)-[h:HasStance]->(m:Stance)
RETURN a,r,u,h,m
```



Results

From the first visualization, we can see how the two communities have high internal cohesion and high external separation. There are very few edges that cross the boundary between communities, and user nodes at the boundary are typically hubs in the network, such as politicians.

An interesting feature to note, that marks a difference between the “Leave” and the “Remain” communities is that in the latter we can find stronger hubs. The “Leave” community shows a lower number of hubs and a more uniform degree distribution. However, this could just be the result of the choice of sample and should be taken with a grain of salt. If confirmed by more experiments, however, this could verify the differences that are normally regarded as stereotypes that demark the political left from the right.

From the second image we see that even though we can still define the two communities, these are less compact. This is possibly due to the large number of hashtags that are available in the dataset; this skews the results as hashtags could relate to the same topic but have different spelling hence not being grouped together.

The most common hashtags refer to generic terms and are used by members of both communities. Some minor hashtags, however, regard topics that strictly relate to one of the communities; as we can see in the visualization, for example, with the tag “voteleave”.

Discussion and future work

This work was heavily limited by the computation power necessary to scrape tweets, to construct the database and to render the visualizations; in fact, only 500’000 tweets were used from the more than 20 million tweets present in the dataset. In future works, a combination of more powerful machines and tools (for example using Gephi instead of NeoVis) could allow for a much richer analysis of this set and could help understand more of the data.

We understand that the operations presented here are not user friendly and replicating the results could be difficult. In future works, a more streamlined workflow could be the key for replicability and extension of this study.

APPENDIX A – WORKFLOW

The workflow is here described in order of execution

Action	Description
Tweet scraping	Run “Scrape_WithTag_Multithread” notebook to scrape tweets from ids in dataset
User standardization	Run “ParseUsers” notebook to transform user file into standard format
Hashtag/Mentions	Run “FindHashtags” notebook to parse mentions and hashtags from retrieved tweets
User IDs	Run “AtTagToUserId” notebook to scrape user ids from tags found in mentions
User Stances	Run “UserStance” notebook to lookup stance and sentiment of users in dataset
Database creation	Run query set to form the database from file “Neo4JCommands”
Visualization	Open the two analysis files in the “Visualization” folder to see the results.