Twitter Analysis on Grande Fratello VIP(2016)

SBN final project

Name: Vamsi Krishna Varma Gunturi
Metricola ID: 1794653
Course: Data science

Introduction:

This report encompasses social and behavioral analysis about the Grande Fratello (Big brother) show 2016 edition. The analysis is carried out based on the tweets of the main contestants and analysis of the spread of influence based on the tweets of individual participants. The main source of data for this analysis has been from twitter, where we analyzed tweets of show participants. Mainly, 5 participants namely **Alessia Macari**, **Valeria Marini**, **Elenoire Casalegno**, **Andrea Damante** and **Pamela Prati** (who are active on twitter) are
considered.

We also had access to a dataset containing all the tweets across 4th of December, worth of 10 Gigs of compressed tweets. So, the volume of dataset was enormous. This dataset was used to collect the tweets of the list of participants mentioned above. We were also provided a graph of sampled network having nodes as twitter ids and edges between nodes representing if two nodes are related or not.

This project involves the fundamentals of Temporal Analysis and Graph analysis and some algorithms like HITS (for finding authorities and hubs), k-means (for clustering) and Label propagation algorithm (to analyze the spread of influence). The whole project has been implemented in Java and the libraries used were Twitter4j, Maven, Lucene, SAX, G Stilo Library.

Temporal Analysis:

Theoretical background:

Temporal statistical analysis enables you to examine and model the behavior of a variable in a data set over time (e.g., to determine whether and how concentrations are changing over time.). In our case we do temporal analysis to analyze the change and spread of opinion about the participants (based on their tweets and other variables) over the period of 10 days through the tweets data and list of participants as a starting point.

Analysis:

1.1) I started the analysis from the list of participants mentioned in the Wikipedia page. Initially there are 14 participants from the 2016 edition but only 5 people mentioned above are active on twitter, so I started with them for my analysis. From each of these 5 participants (Alessia Macari, Valeria Marini, Elenoire Casalegno, Andrea Damante and Pamela Prati) I did a query on the main tweets dataset to create separate indexes for each participant.

1.2) We then analyze the words mentioned in these 5 tweets sets and extract top 1000 words (of each participant) based on their frequency. We also build a SAX string for every term with grain = 12h for doing temporal analysis. After this, we cluster these words using k-means algorithm I put together to do a clustering of the words which expose similar temporal behavior. This clustering is done on the 5 sets of top 1000 words from 5 participants.

1.3) For every cluster obtained in the previous step we built a co-occurrence graph based on the number of documents(tweets) a pair of terms occur together and how similar are their SAX strings (temporal occurrence) divided by respective term frequency which gives us 2 scores, we consider the maximum of these 2 scores as edge weight between these 2 words or terms in the graph. We build a co-occurrence graphs like this for each cluster of top-1000 terms for each participant (or words) that we got in the previous step. We then find the connected component and k-core for each of these graphs to obtain sub-set of words for each cluster of words from the previous step. so, we focus our attention towards the words that really matter for our analysis of referendum. so, we defined a score function to determine the edge weights in the graphs that we created for the co-occurrence graph

With this we conclude the temporal analysis of the available tweet data superimposed with the list of participants that we started with. Below are some results that I obtained in this analysis (in the results section)

Code analysis:

We used the following files to create indexes and plots,
For plots I made use of Python and code is present in **spread_of_tweets.py** and **term_worcloud.py** files inside plots folder.

TweetIndexManager.java (extended from base abstract class IndexManager.java) and helper classes TweetIndexBuilder.java (extended from base abstract class IndexBuilder.java) to actually build the tweet indexes. - this is for building all the tweet related indexes like core tweets stream, all of the 5 participant tweet indexes. This sort of hierarchical code division aided me in clear separation of concerns and better code reusability and code maintenance for further edits.

ClusterGraph.java which makes use of quite a few libraries from G library is used to build co-occurrence graphs from the clustered term sets. ClusterGraphFactory.java is a mediator class to manage group of Cluster graphs easily without any code duplication.
TweetTerm.java is used to hold the SAX and binary representation for the frequent terms from the tweet indexes for temporal analysis and better code separation and object manipulation.

Results:

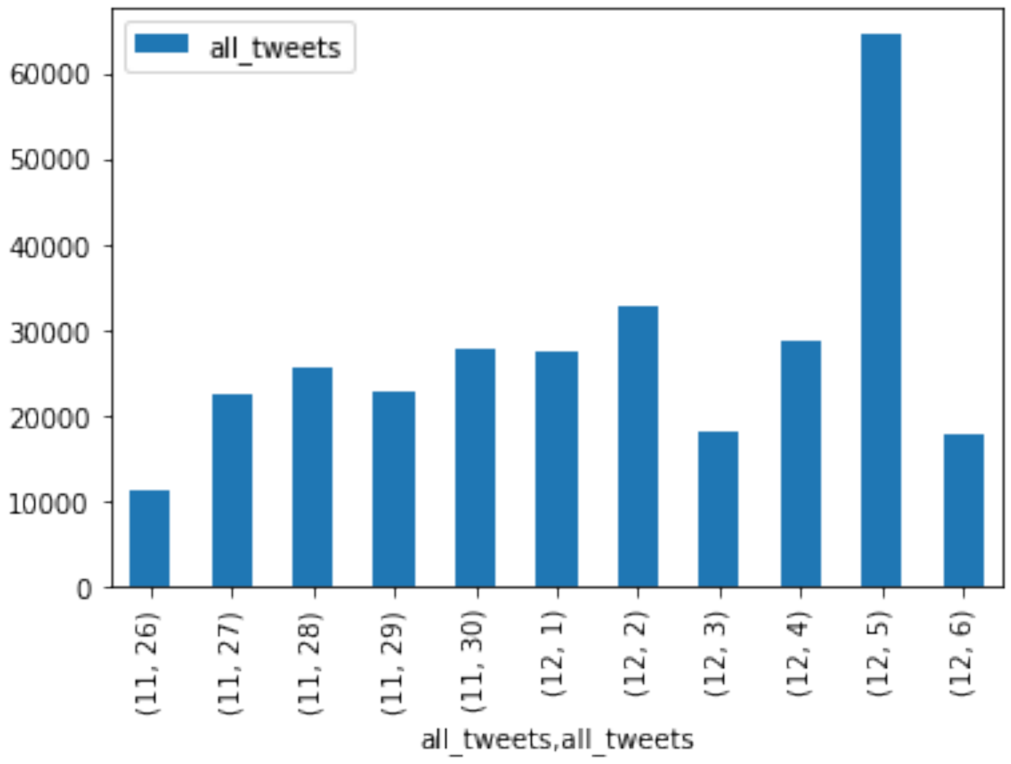
1.1) Initially based on the twitter stream data (from 26/11/2016 to 6/12/2016) that was provided, I created indexes for each tweet using Lucene. In total **18850000** tweets were indexed. Following information is stored for every tweet index, userId, date, name, screenName(this is the twitter id which is unique for every user), tweetText, hashtags, mentioned, followers, rtScreenName, rtUserId(this holds re-tweet information).

Then the tweet indexes are created for each of 5 participants and this yielded tweets as follows,

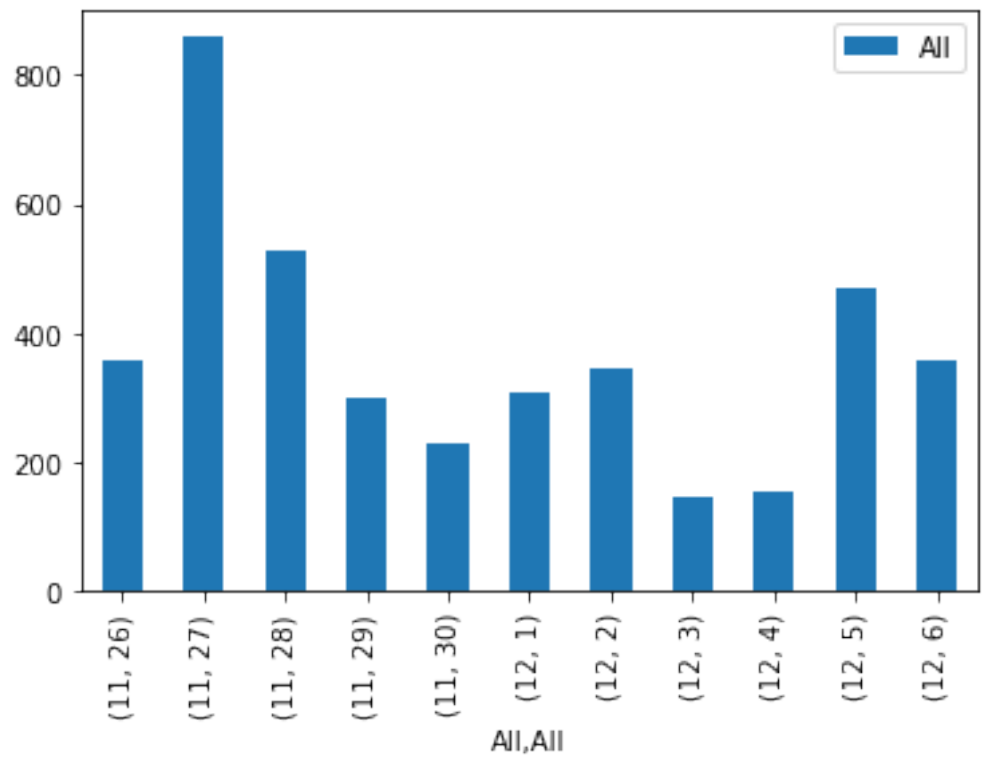
Alessia Macari - AlessiaMacari1 – 1385 tweets
Valeria Marini - ValeriaMariniVM – 1200 tweets
Elenoire Casalegno - elenoirec – 1094 tweets
Andrea Damante - AndreaDamante – 262 tweets
Pamela Prati - PamelaPrati – 119 tweets

So the tweets sample is considerably small from the tweets dataset we have (from 26/11/2016 to 6/12/2016). Below are some plots for tweet distribution of entire tweets dataset and all 5 participants tweets

ALL TWEETS



ALL PARTICIPANTS TWEETS:

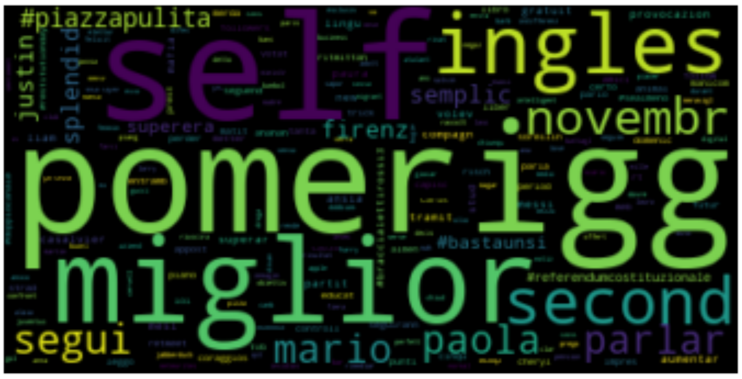


NOTE: Please check Appendix 1 for tweet distribution of individual participants.

1.2) We analyzed each participant tweets and extracted top 1000 words from both YES and NO tweets (text is extracted from 2 fields, tweetText and hashtags) based on their frequency. Then from every word of these 2000 thousand words we generated a SAX string based on its frequency over the 10 days period under consideration with the grain value of 12 hours so as to know the temporal behavior of each word over the period. We use this SAX representation to cluster similar words using trivial k-means algorithm. In other words, we group words in to clusters based on their temporal behavior. Top 1000 each participant s and their respective frequencies can be found in **output** folder.

Note: I ensured that there are no STOP words in the top 1000 words from each of participants by the comparing each term with stop words updated in **input/stopwords.txt**

Alessia Macari (words) word cloud -



Note: Appendix 2 contains remaining participants word clouds.

For k-means, with the elbow method we arrived at **clusters=10** to be the optimal cluster terms for our list of words. We considered number of **epochs/iterations to be 1000** for the clustering so if the centroid calculation reaches saturation then the algorithm terminates without any further centroid readjustment.

1.3) From the clusters of words (20 clusters, 10 each for YES and NO words) obtained in the previous step, we built co-occurrence graph for each cluster of words and generated the graph through a threshold from a score function we put together.

The score function goes like this,

```
double div1 = intersection / uFreq;
double div2 = intersection / vFreq;

// Get the max of them
float maxRelFreq = max((float) div1, (float) div2);

// If this quantity is higher than a threshold add the edge between the nodes
if (maxRelFreq > 0.0001) {
    g.add(i, j, 1);
}
```

So, if the score of a pair of words is greater than 0.001 we add the edge to the co-occurrence graph otherwise we skip the connection between 2 terms.

Once the co-occurrence graphs are built for every cluster then we generate Connected Components and k-core for every cluster to extract the sub-set of words which are highly connected (and there by very related) for every cluster so as to deep dive in to our analysis on the words that really matter for the referendum analysis.

Results for Connected Components of all the clusters can be found in:
output/relComps.json (with separate arrays for clusters from YES and NO words)
output/ccWords.json - contains all the subset of YES and NO words from CC components

similarly, we updated the results of k-core in **output/relCores.json** and **output/coreWords.json**

Note: After all the supporter analysis and Spread of influence we found that K-Core yields better results in terms of number of hubs and authorities created for our specific analysis than Connected components results.

Identify mentions of participants or supporter analysis:

Analysis:

2.1) Initially to create the supporters indexes I created a new java object with name Supporter.java with properties like id, name, pMentioned(participant mentioned), cUsed(words used), eUsed(expressions used) So, a lot of metadata has been added to each supporter object for ease of analysis and correlation. We then query through each supporter object to decide the vote of each supporter (one of the 5 participants) based on a score function(more details below) and create Lucene index with following properties name, id and vote for each supporter and store it for further analysis

Next, we made use of pre-defined expressions for each participant to further segregate supporters list. Below are the expressions we used,

Alessia Macari Expressions:
#cioco", "#macari", "#alessia", "#frosinoneculone", "#laciociaria", "#donne", "#malessia"

Valeria Marini Expressions:
"#valeria", "#marini", "#donne", "#mvaleria"

Elenoire Casalegno Expressions:
"#andrea", "#damante", "#damellis", "#giuliadelellis", "#andreaiannone", "#uomo", "#dandrea"

Andrea Damante Expressions:
"#elenoire", "#casalegno", "#donne", "#celenoire"

Pamela Prati Expressions:
"#pamela", "#prati", "#donne", "#pratiful"

Then we implemented a score function to update the vote of each supporter as mentioned in the following pseudo code,

```
for(int i=0; i < p.pList.size(); i++) {
   float curScore = (float) ( mList.get(i) + 1 * cUsed.get(i) + 3 * eUsed.get(i)) ;
   scoreList.add(curScore);
}
```

2.2) In this step, Using the provided Graph and the library G, we generated the subgraph induced by users S(M) from the previous step (here users = supporters created before). Then we found the largest connected component CC and computed HITS on this subgraph of M users.

2.3) By performing HITS on the subgraph created on the previous step, we could compute the authority and hub scores of each node sorted in descending order. Now we basically extract the node ids from the graph and get their support (one of the 5 participants) already computed in the step 2.1 (using the score function). We extract top 1000 (for all the participants) Authorities and HUBS from the HITS result using functions provided in G library for Hubs and Authorities (using HubnessAuthority algorithm). All the results are mentioned in the Results sub section below.

Code analysis:

The workflow for Supporter Index creation is more or less same as both Tweets indexes with some additional utilities added for score functions and other requirements. **SupporterIndexManager.java** (extended from base abstract class **IndexManager.java**) and helper classes **SupporterIndexBuilder.java** (extended from base abstract class IndexBuilder.java) to actually build the supporter indexes. All the supporter indexes have format mentioned in **Supporter.java**, we use this class to manipulate supporter indexes once we retrieve them.

Results:

2.1) From the analysis we did, we obtained **48071** supporters.

Out of the **48071** supporters,
971 - from mentioned field in the tweet stream
43588 - from partcipant expressions mentioned above
3507 - From list of words from the k-core algorithm

2.2) Provided graph file has been correlated with above **48071** supporters and the resultant subgraph has been saved to **output/ccsg.txt** file.

We could obtain this in the created subgraph,
Pamela Prati has **401** supporters
Valeria Marini has **1709** supporters
Andrea Damante has **5286** supporters
Alessia Macari has **5849** supporters
Elenoire Casalegno has **5911** supporters
Supporters not found: **35844**

2.3) We then computed HITS on the above generated sub-graph. Below are the results we obtained authorities of each participant as follows,

Pamela Prati has 199 authorities
Valeria Marini has 754 authorities
Andrea Damante has 1000 authorities
Alessia Macari has 1000 authorities
Elenoire Casalegno has 1000 authorities

and hubs –

Pamela Prati has 401 hubs
Valeria Marini has 1000 hubs
Andrea Damante has 1000 hubs
Alessia Macari has 1000 hubs
Elenoire Casalegno has 1000 hubs

Above results are saved to following files,

output/Alessia_Macari_supporters.txt – all the list of supporters or users for Alessia Macari and similarly, for other participants
output/authorities.txt – contains all the authority scores of all the supporters (48071**)**
output/Alessia_Macari_authorities.txt – contains top 1000 authorities of Alessia Macari sorted by authority scores
and similarly, for other participants
output/unclassifiedAuthorities.txt – contains the authority scores of unclassified supporters
output/hubs.txt - contains all the hub scores of all the supporters (48071)
output/Alessia_Macari_hubs.txt – contains top 1000 hubs of Alessia Macari sorted by hub scores

Spread of Influence:

Analysis:
To do the analysis for spread of influence over the period of 10 days we created a wrapper for Label propagation algorithm which not only does the clustering of all the nodes of the graph provided but also adds a label(from 1 to 5 for 5 participants) on top of the clustering, so we can analyze how the spread of influence about the opinion (among the participants) throughout the network in a nutshell. This analysis kind of gives us an additional view point so that we can be very clear about the actual opinion of the supporters and all the nodes in the network in one place.

3.1) For the list of supporters(for all the 5 participants) we gathered in the previous step (from files output/Alessia_Macari_supporters.txt etc..,) we ran a customized version of LPA we prepared and while initializing the labels for the algorithm we pushed
1 for Alessia Macari supporters,
2 for Valeria Marini supporters,
3 for Elenoire Casalegno supporters,
4 for Andrea Damante supporters,
5 Pamela Prati supporters..

3.2) We did a similar analysis for the nodes in output/Alessia_ Macari_authorities.txt (and similarly for other participant authorities) as well as output/ Alessia_ Macari_hubs.txt (and similarly for other participant hubs) so that we can have a holistic analysis of spread of influence for both authorities and hubs there by removing any inconsistencies in the intermediate analysis.

Code analysis:
We created a customized version of Label propagation algorithm to account for additional labels on the individual nodes. The code is present in the CommunityLPA.java. We initialize all the nodes in the graph with zero and then add labels to the corresponding supporters extracted from the files mentioned above (all the files are in the output folder)

Results:

We applied LPA algorithm to 3 sets of data, All participant supporters, All participant authorities, All participant hubs. Below are the results obtained,

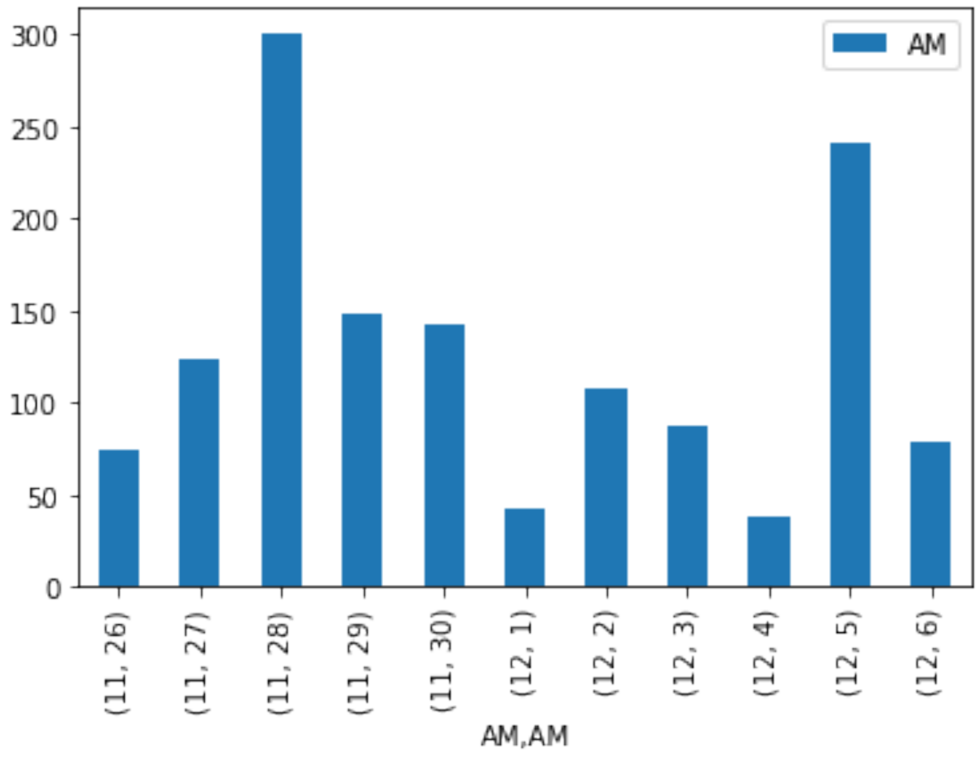
Supporters:
Alessia Macari has 5818 supporters
Valeria Marini has 8736 supporters
Elenoire Casalegno has 366790 supporters
Andrea Damante has 8429 supporters
Pamela Prati has 502 supporters
UNCLASSIFIED supporters: 59919

Hubs:
Alessia Macari has 1001 hubs
Valeria Marini has 380976 hubs
Elenoire Casalegno has 5996 hubs
Andrea Damante has 3017 hubs
Pamela Prati has 575 hubs
UNCLASSIFIED hubs: 58629

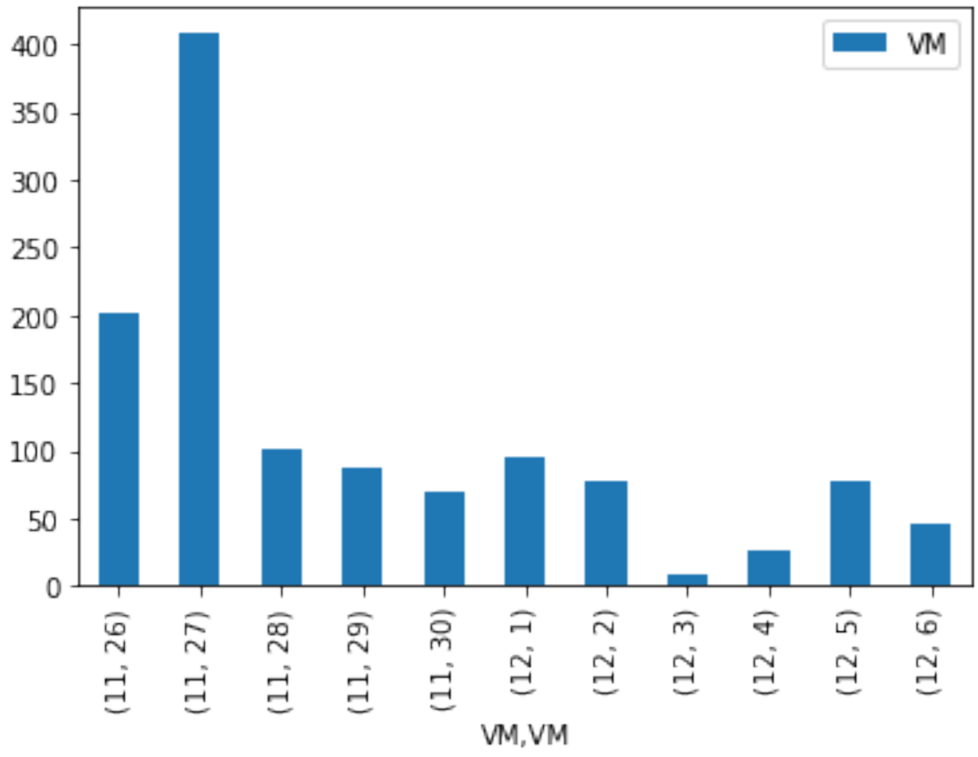
Authorities:
Alessia Macari has 998 authorities
Valeria Marini has 18930 authorities
Elenoire Casalegno has 366787 authorities
Andrea Damante has 4561 authorities
Pamela Prati has 263 authorities
UNCLASSIFIED authorities: 58655

APPENDIX 1:

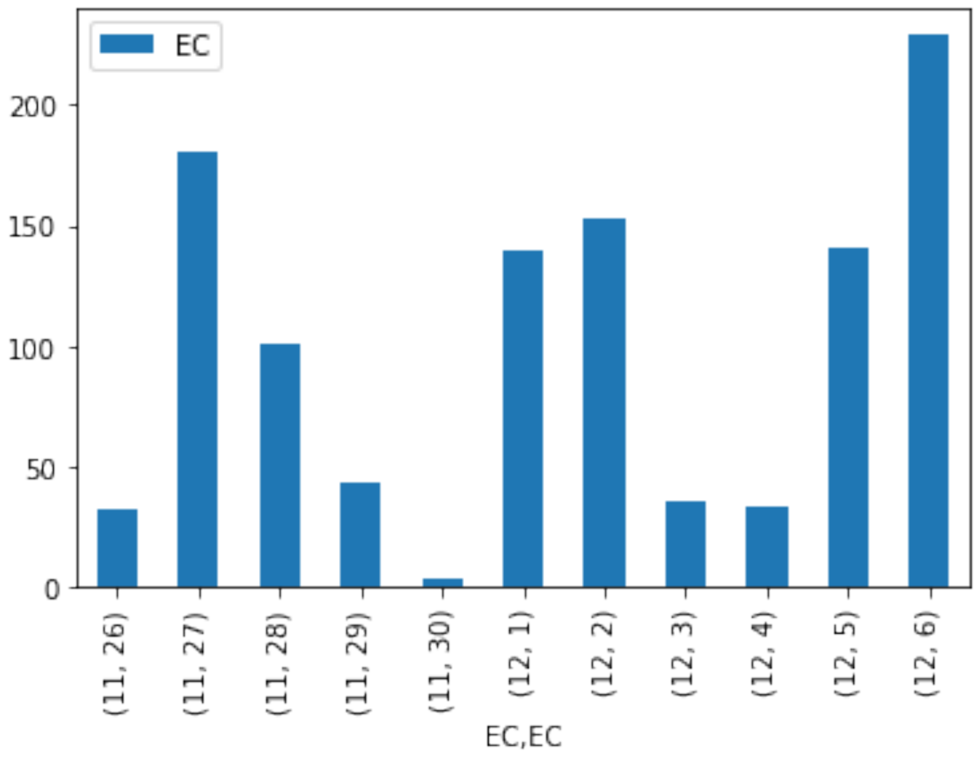
ALESSIA MACARI TWEETS:



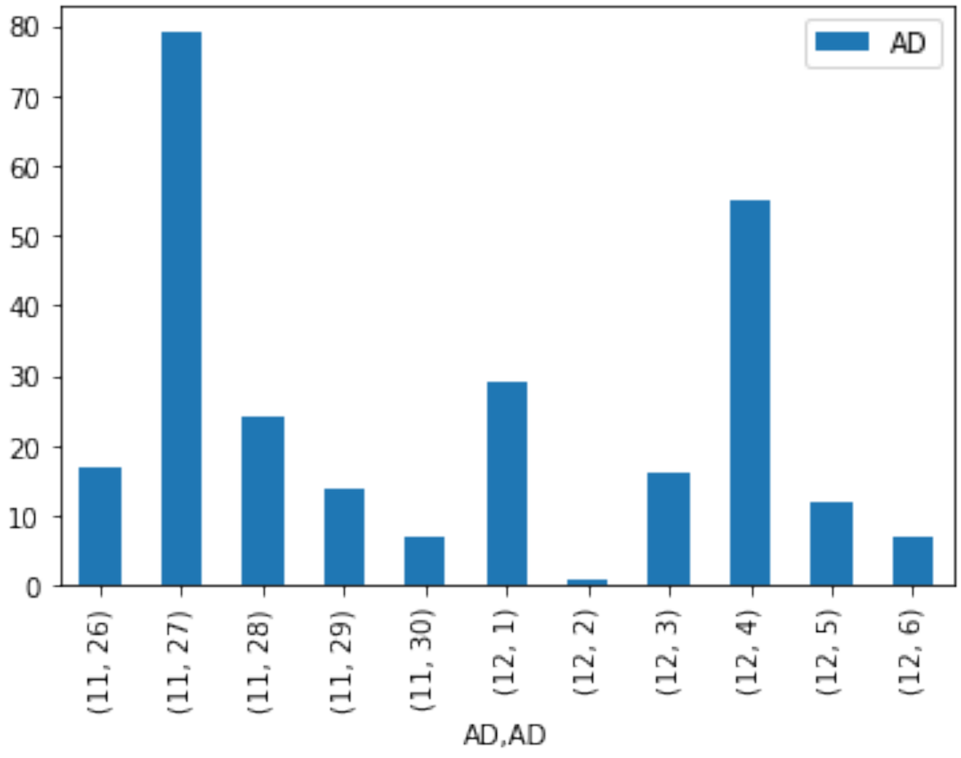
VALERIA MARINI TWEETS:



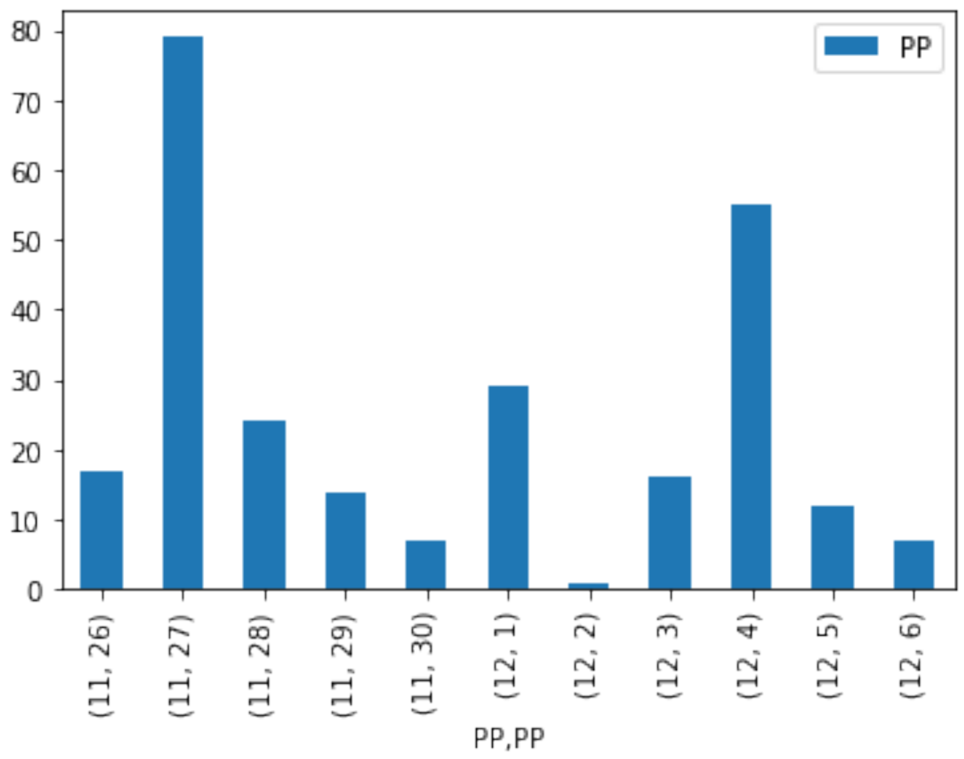
ELENOIRE CASALEGNO TWEETS:



ANDREA DAMANTE TWEETS:

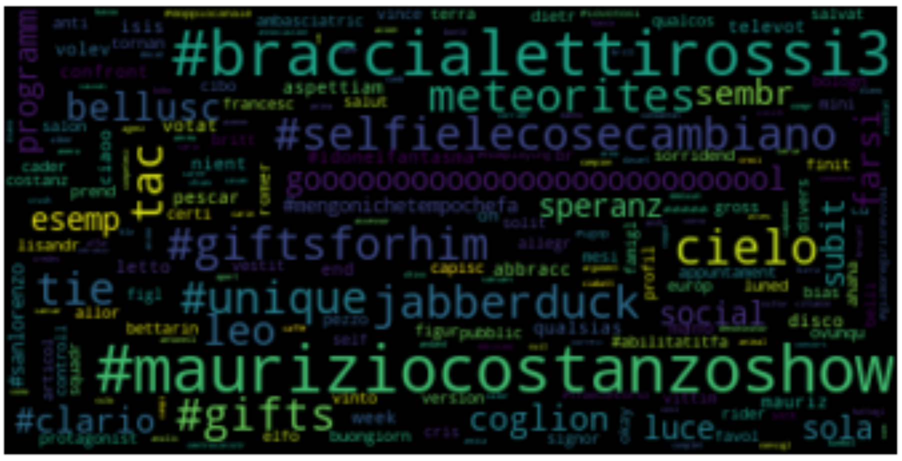


PAMELA PRATI TWEETS:

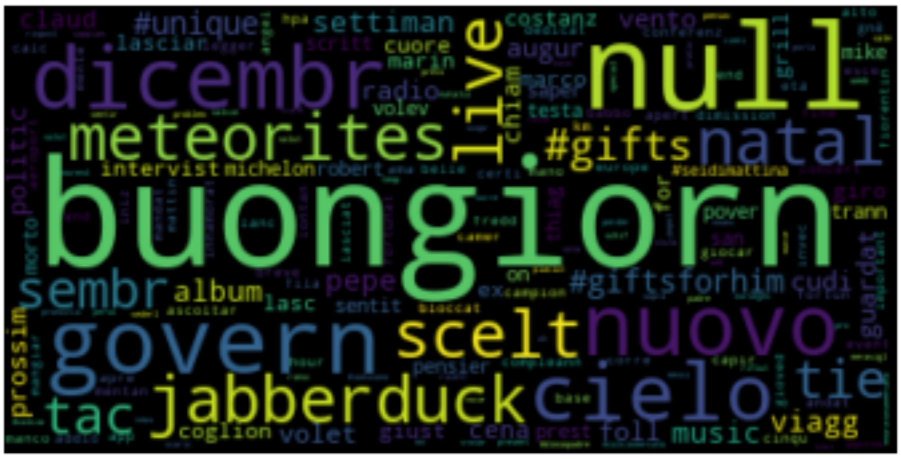


Appendix 2:

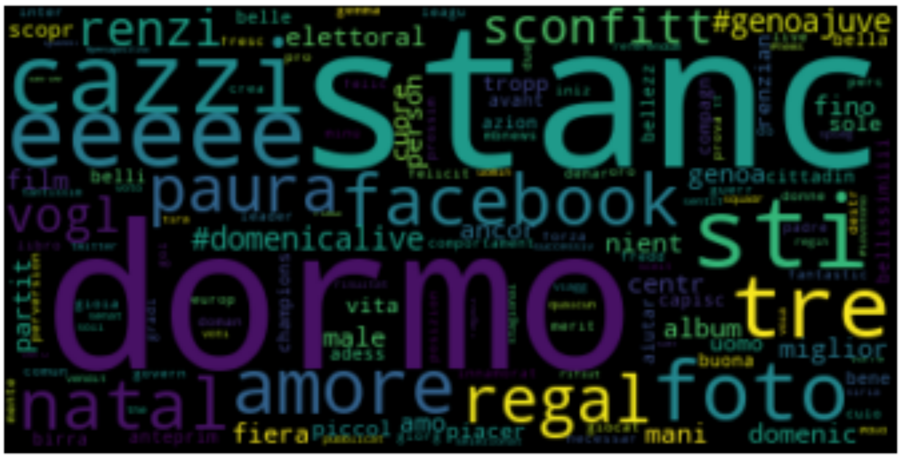
VALERIA MARINI word cloud:



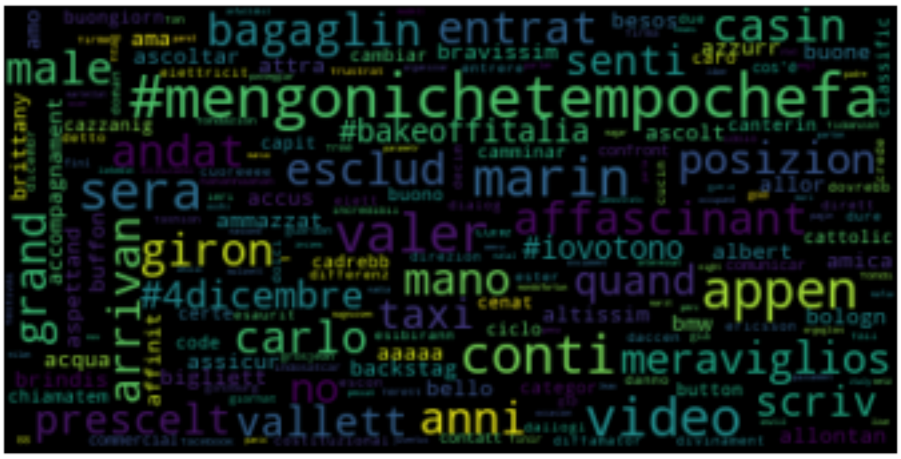
ELENOIRE CASALEGNO word cloud:



ANDREA DAMANTE word cloud:



PAMELA PRATI word cloud:


#mengonichetempochefa
bagaglin
entrat
casin
male
senti
#bakeoffitalia
andat
esclud
marin
posizion
sera
affascinant
valer
giron
#iovotono
#4dicembre
mano
quand
appen
taxi
carlo
conti
meraviglios
no
bello
prescelt
vallett
anni
video
scriv
grand
arrivan
acqua
brindis
buffon
brittany
accompagnament
ascoltar
bravissim
azzurr
buone
ascolt
cazzanig
ammazzat
cadrebb
altissim
bologn
backstag
ciclo
cattolic
amica
albert
besos
buongiorn