Social and Behavioral Networks - Italian Referendum 2016

Pedro Magalhães Bernardo 1794547

Amir Abbasinejad 1793613

Introduction

In 2016 the then Prime Minister of Italy, Matteo Renzi, called a constitutional referendum to reform the composition and powers of the Parliament of Italy, as well as the division of powers between the State, the regions, and administrative entities. The vote was held on 4th December of 2016. Voters were given two options, yes or no. This project aims to analyse the Twitter interaction between the supporters of the two groups on the days that led to the referendum vote.

This analysis is comprised of three steps. First we look at the temporal interaction between yes and no supporters. To do that we look at the tweets produced in Italy from 26th November of 2016 to 2nd December of 2016. We identify the main influencers (politicians and journalists) of the two groups and analyse the tweets produced by then along this timeframe. The goal of this step is to identify the temporal behaviour of the main themes discussed by the two groups as well as possibly identify some kind of action-reaction interaction that happens between them.

The second step tries to idenfity the supporters of each group among all the Twitter users that produced content in the platform through the timeframe. To do that we look at users that mentioned influencers of each group, as well as looking at users that produced tweets with similar words of those used by the influences of each group. Besides that we use the given graph, which shows users that follow each other, to look for the main authorities, hubs and key players among the supporters for each group.

Finally, using the given graph we try to identify the spread of influence of each group.

The Twitter data and the mentioned graph were given as input of this project and are not a product of the analysis itself.

0. Data Pre Processing and Graph

0.1 Data Pre Processing

Before starting the analysis we indexed all the data available into a Lucene index. Besides indexing the information available we also processed the text to save the terms vector for each tweet. This is done to normalize the data and make it more easier to analyse. In order to do that the following steps are done:

1. All url links are removed from the text.
2. Every word with 2 or less chars are removed.
3. Every special char, number and accent is removed.
4. Italian stop words are removed.

This process is done only to save the terms vectors for each tweet, so the original text it is still saved, indexed and searchable. However, everytime in the project that text analysis is done (such as constructing SAX strings for terms, word frequency and so on) it is done on top of the processed terms vectors. But, whenever the original text is necessary (such as looking for users mention) it is used.

0.2 Graph

The graph made available is a user-user representation of the Twitter network. This means one edge coming from user U and arriving on user V represents that user U follows user V on the platform.

However, the way the graph is presented leaves to interpretation the direction of the edge. With that in mind we did a preliminary analysis of the graph to determine this direction. Since, usually, users on twitter follow less users than they have following them, and also after checking some specific users of interest (such as politicians) who in fact have much more followers than people who they follow, we decide to interpret the source on the original graph (first user id on the line) as the user who is being followed by the destination (second user id on the line).

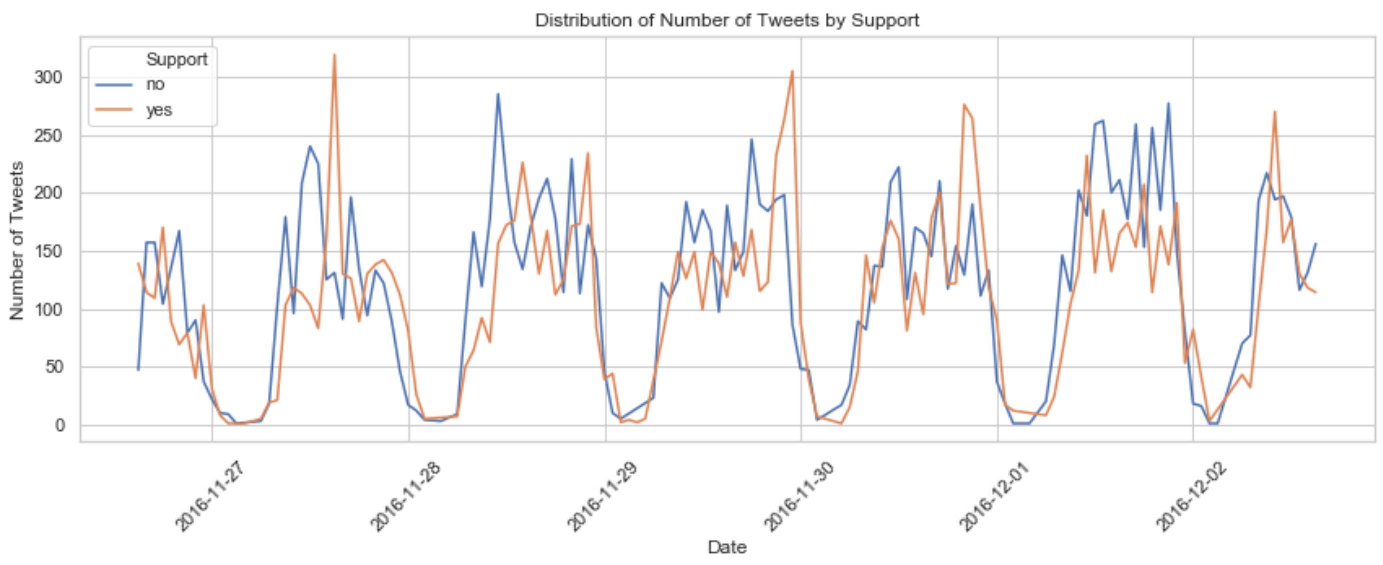
1. Temporal Analysis

1.1 Influencers Support

We start by manually identifying the main influencers for each group, yes and no. To do that we looked online on different sources to determine who were the main politicians and journalists, that were also Twitter users, that made their position public. With the goal to keep a balanced analysis we chose 120 influencers that supported NO, and 120 influences that supported YES. However, when we look at the number of tweets we can see that the NO supporters are more active, we get 16448 tweets by NO supporters and 14827 tweets for YES supporters, the table below shows a summary.

Support	Number of Users	Number of Tweets
Yes	120	14827
No	120	16448

The graph below shows how is the distribution of number of tweets by support throughout the days.



We can see a a similar temporal behaviour between both groups when it comes to the number of tweets produced on the days that led to the referendum day.

1.2 Cluster of Words

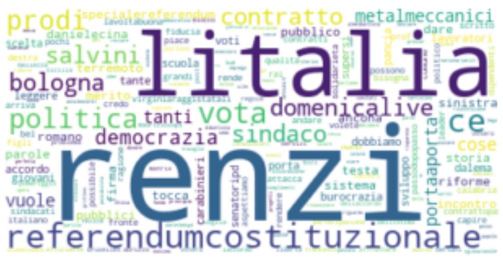
Now we take look at the actual text of the tweets produced by each group. To do that we retrieve the vector of terms from the Lucene index for each tweet of each user identified previously. After that we produce the SAX strings for each of the top 1000 words used by each group. To do that we use the JMotif Java library that implements SAX algorithms. With the SAX strings we are able to run a K-Means algorithm to identify clusters of words based on their temporal behaviour. The algorithm works as follows:

1. Set a number of clusters K (for this case we identified, empiracally, that K = 3 was a good option).
2. Choose the K initial centroids (random SAX strings).
3. Calculate the distance between each word and each centroid and assign the cluster according to the closest centroid.
4. Define new centroids (SAX strings) using the average (ASCII values) of each char of the SAX strings that comprise each cluster.
5. Go back to 3 and repeat until the clusters don't change anymore or it reaches a limit of iteratinos (determined to be 100 in our case).

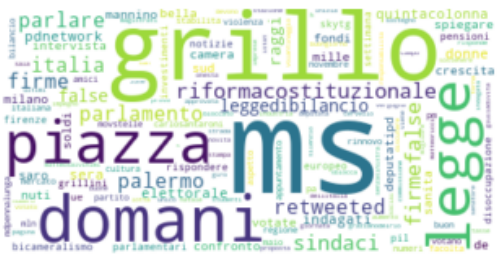
The distance used in 3 is the *mindist* proposed for SAX strings. The distance is defined on the original paper that proposes the SAX representation and it is also implemented in the JMotif library.

For the YES supporters we have the following word clouds for each cluster.

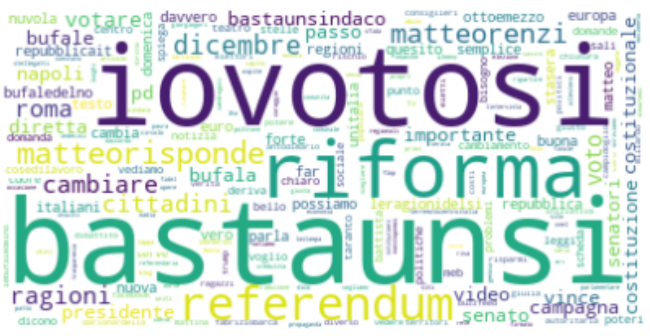
Cluster 1



Cluster 2



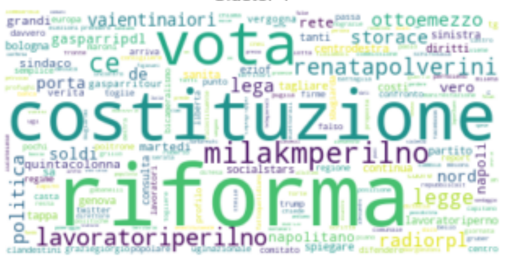
Cluster 3



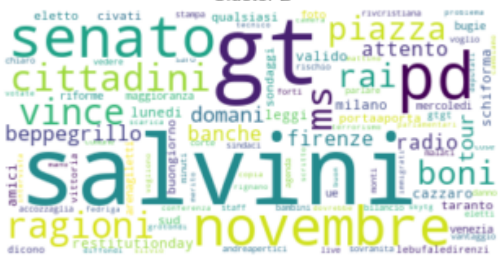
For cluster 1 we can see that the main words (by frequency) are "renzi", "litalia" and "referendumcostituzionale", which refers to then PM of Italy Matteo Renzi, who was responsible for calling the referendum and was also the main supporter for the Yes group. The main words for cluster 2 are "grillo", "ms", "firmefalse" among others, this refers to the political party "Movimento 5 Stelle" (M5S) and their leader, Beppe Grillo. The "firmefalse" refers to an investigation against the party in an electoral tampering case for faking and cloning citizen signatures in 2012 local elections. The case got a spotlight on the days before the referendum since the M5S was one of the main supporters for the No group. This shows that Yes supporters were trying to bring the issue to light in an effort to discredit their opposition. Finally, the main words for cluster 3 are "referendum", "iovotosi" and "bastaunsi", the last two were the main hashtags shared by the Yes supporters on the days that led to the referendum.

For the NO supporters we have the following word clouds for each cluster.

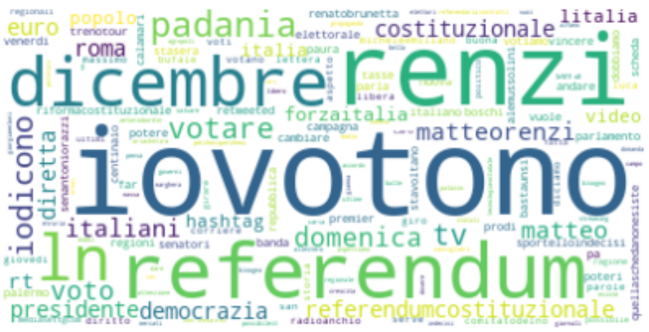
Cluster 1



Cluster 2



Cluster 3

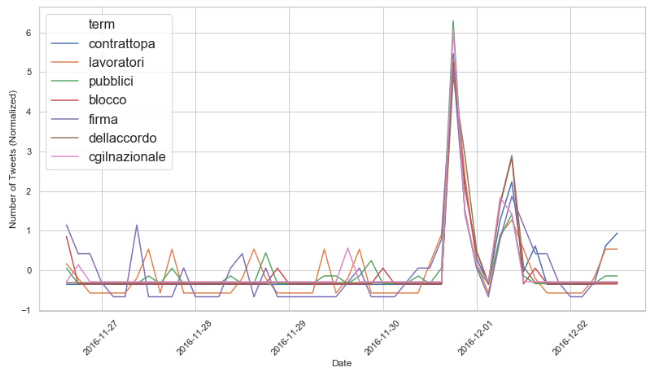


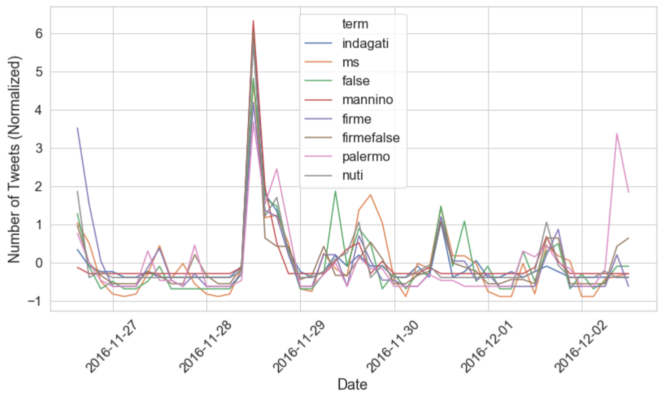
For cluster 1 we can see that the main words (by frequency) are "riforma", "costituzione" and besides that we can also see the words "milakmperilno" and "lavoratoriperilno" which were two big hashtags used by the group "Lavoratori Per Il No" which was a coordinated group supporting NO. The main word for cluster 2 is "salvini" which refers to the member of the Lega party, Matteo Salvini, he is an active Twitter user that supported the NO. Finally, on cluster 3 we see the main hashtags used by the No supporters, such as "iovotono" and "iodicono", besides that we also see the word "renzi", which shows that No supporters were also making their opinion about Matteo Renzi known.

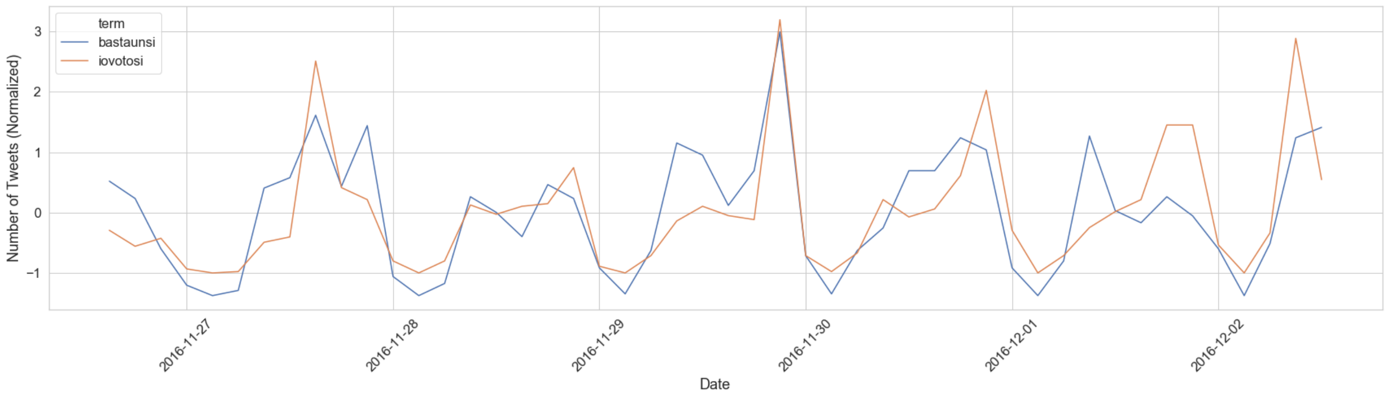
1.3 Graph Analysis

For each cluster identified we build a co-occurence graph. In this graph two words are connected if they appear in the same tweet, and the weight of the edge between them is the number of tweets that they appear together. After building the graph we extract the largest connected component and the innermost core (k-core) of each one. Since most of the words are connected, the largest connected component most of the times is the full graph, so this doesn't give us any new information. However, if we look at the k-core we can see the words that are more likely to show together in a tweet for each cluster.

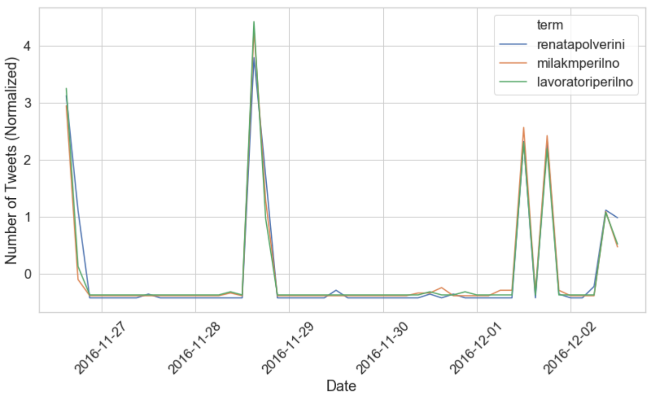
Using the words present in the k-core for each cluster we can see how they behave througout the time. The graph below shows the words in the k-core for each cluster for the YES supporters (for this and the next graphs in this section the frequencies of words are normalized).

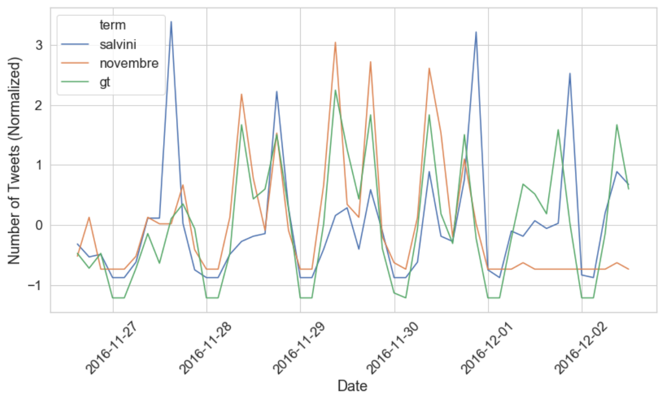


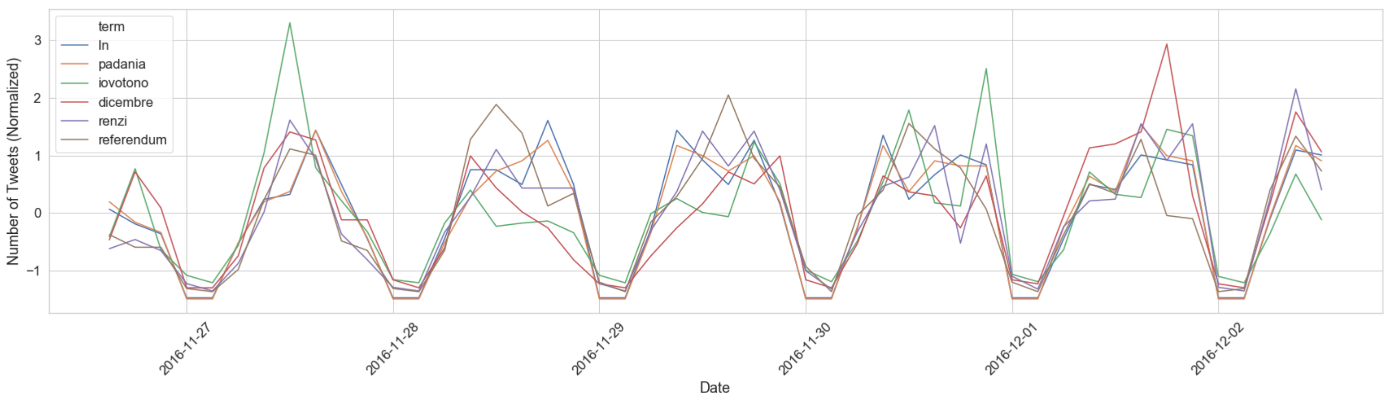




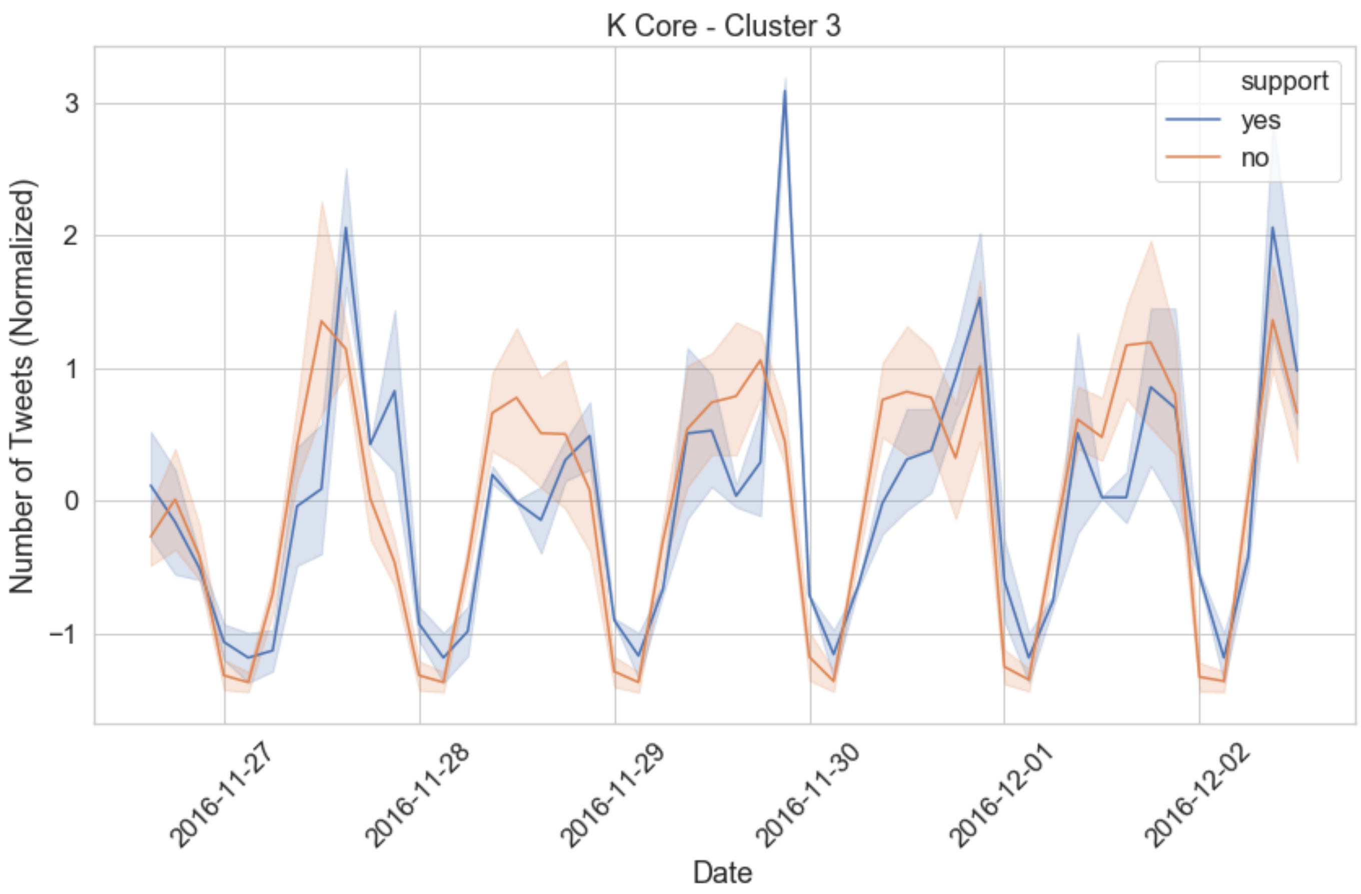
The graph below shows the words in the k-core for each cluster for the NO supporters.







We can see that the cluster 3 for each case seems to have similar behaviours, so lets compare them.



The graph shows the aggregate for every word (with a confidence interval) in the k-core of cluster 3 for YES supporters and NO supporters. We can clearly see an action-reaction happening here. When NO supporters start to show they support there are massive responses from YES supporters, this is specially highlighted between the days 29 and 30.

2. Identifying Yes/No Supporters

2.1 Identifying Supporters

Now we want to take a look at all the supporters, not only influencers, for each group. To do that we retrieve every tweet that mentioned one of the influencers as well as every tweet that had one of the main hashtags used by each group, for the YES supporters we use the hashtags "iovotosi", "iodicosi" and "bastaunsi", for NO supporters we use the hashtags "iovotono", "iodicono", "bastaunno". Finally, based on these tweets we give a support score for each user using the following formula:

$$S = (M_y + 3H_y) - (M_n + 3H_n)$$

- M_y is the number of mentions of users that are yes supporters.
- M_n is the number of mentions of users that are no supporters.
- H_y is the number of times that YES hashtags are used.
- H_n is the number of times that NO hashtags are used.

We give a higher weight to the use of hashtags since mentioning an influencer not always indicate support for what the influencer supports. Finally, we classify each user using the following formula:

$$C = \begin{cases} Yes & S > 10 \\ No & S < -10 \\ Neutral & -10 \leq S \leq 10 \end{cases}$$

We were conservative when assigning the support to only keep the true supporters for each group.

After setting the support for each user collected we have the following numbers:

Support	Number of Users	Number of Tweets
Yes	3564	126698
No	7532	215501
Neutral	40531	89381

We can see that although most of the users are neutral, most of tweets are still concentrated between the YES and NO supporters. However, the number of actual NO supporters is almot double as the number of YES supporters.

2.2 Authorities, Hubs and Key Players

After identifying the users that support each option we want to highlight who are the authorities, hubs and key players for each group in the network. To do that we use the provided graph and the G library.

Using the users retrieved we generate the induced subgraph of the original graph and then we compute HITS on the induced sugraph. To identify the authorities we simply look at the authority score given by the algorithm, for the hubs we use a combine metric between the hub score given by the algorithm and the support score S defined aboved. This new score, that we call centrality score (CS) is given by the following formula:

$$CS = \frac{2H + S}{3}$$

- H normalized (between 0 and 1) hub score.
- S normalized (between 0 and 1) support score.

This represents a weighted average between the two scores giving a higher weight for the hub score. The decision to give a higher weight to the hub score stands on the fact that although a user can demonstrate a lot of support for one option (high S score), this does not necessarily show that he is well connected to the network. By using this weighted average we capture the users that are central to the network but that are also highly supportive of their group.

Authorities in this case are the users who have a lot of followers and are more likely to drive the discusion among the supporters of the group. We expect the top authorities to be among the influencers identified in part 1 of this analysis. Hubs, on the other hand, should be users that are active on Twitter, heavily support their group and follow a lot of the authorities of the group.

Let's have a look at the top 10 authorities for each group:

No Supporters		Yes Supporters	
Rank	**User**	**Rank**	**User**
1	fattoquotidiano	1	matteorenzi
2	beppe_grillo	2	repubblicait
3	civati	3	lauraboldrini
4	La7tv	4	Montecitorio
5	lucatelese	5	graziano_delrio
6	PiazzapulitaLA7	6	angealfa
7	ultimenotizie	7	pdnetwork
8	GiorgiaMeloni	8	serracchiani
9	Mov5Stelle	9	nomfup
10	NichiVendola	10	nzingaretti

We can clearly see that for both cases we mainly have politicians (beppe_grillo, GiorgiaMeloni, matteorenzi, lauraboldrini) and journalists (fattoquotidiano, La7tv, repubblicait), who were heavy supporters for their groups, which was expected as per the explanation above.

Now lets have a look at the top 10 hubs for each group:

No Supporters		Yes Supporters	
Rank	**User**	**Rank**	**User**
1	domenicoformic3	1	angelinascanu
2	italse	2	57_mimmo
3	GASPARECARLINI	3	AntonellaGramig
4	Didi1648	4	AS3691
5	AurelioToro	5	paolocaccamo68
6	babylonboss	6	danielecina
7	pippodido1	7	siperilsud1
8	soniabetz1	8	gioicaro
9	Giangiagainst	9	SteAlbamonte
10	LemGiuseppe	10	martinorosario2

In this case is harder to identify each user since they might not be known personalities. What we can see are user who were heavy supporters of their respective group, who follow a lot of politicians from the parties that they support, and that are mostly engaged in political discussions on Twitter.

Finally, we want to identify the key players in the network. To do that we use the KPP-NEG algorithm, which identifies nodes that if removed from the network are more likely to disrupt or fragment the network.

Let's take a look at the top 10 key players for each group:

No Supporters		Yes Supporters	
Rank	**User**	**Rank**	**User**
1	senantoniorazzi	1	matteorenzi
2	flavia_marzano	2	Montecitorio
3	oinot49	3	angelerrimo
4	QuagliarielloG	4	flavagno
5	iLGae75	5	claudioliga63
6	mauroev	6	nerodiseppia56
7	matteosalvinimi	7	VittorioBotti
8	alegio957	8	AntonioRepossi
9	ConversioneOrg	9	POAxSI
10	molotov00	10	andrearossi76

Looking at the key players we can see that most of the ones on top are politicians who represented a political party that supported either YES (matteorenzi, flavagno) or NO (matteosalvinimi, senantoniorazzi). Besides that they are also the ones with the highest number of followers, Matteo Renzi (matteorenzi), for example, has 3.4 million followers, whilst Matteo Salvini (matteosalvinimi) has 1.4 million followers. They not only represent opposite political positions, what is reflected on their support on the referendum, but they are also well connected on the network. Besides that, Matteo Renzi was the one responsible for calling the referendum, so it makes sense that he is one of, if not the, biggest key player in the network, as he attracted attention of both groups. This shows that removing these users from the network could lead to a break on the flow of information, as they are responsible for connecting the other nodes of the network.

3. Spread of Influence

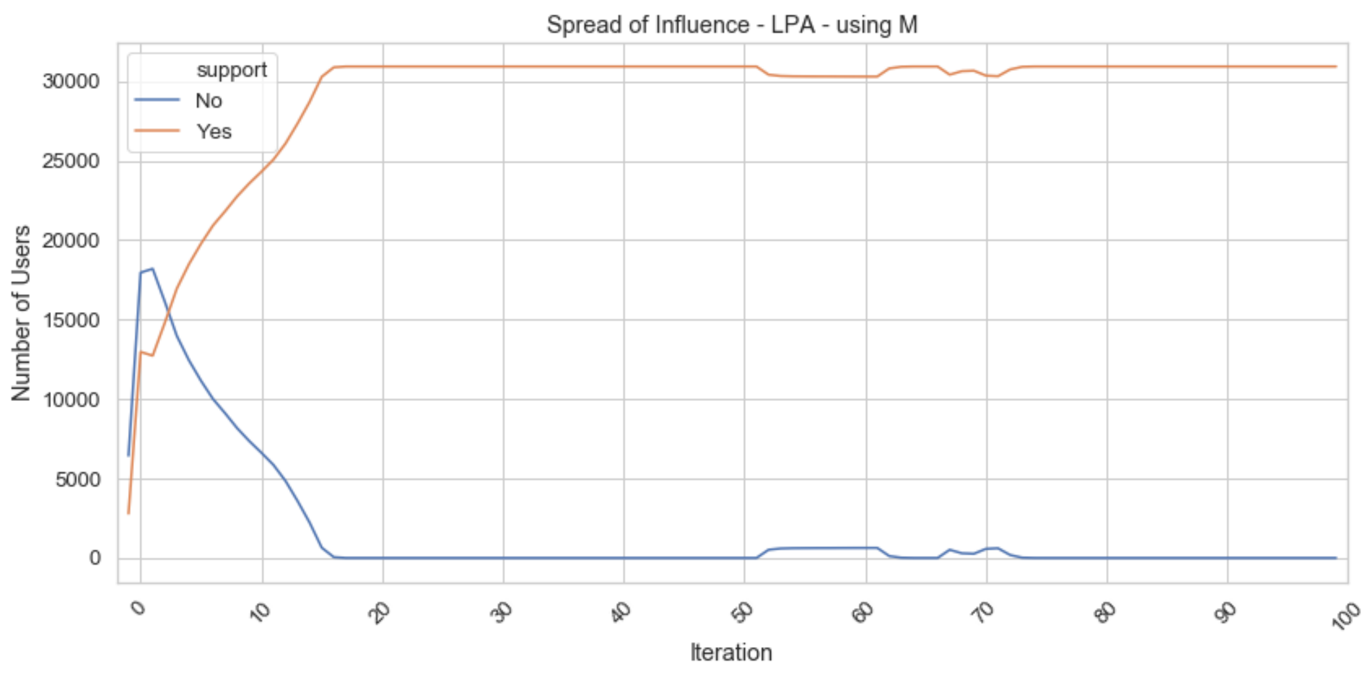
3.1 Spread of Influence using Label Propagation

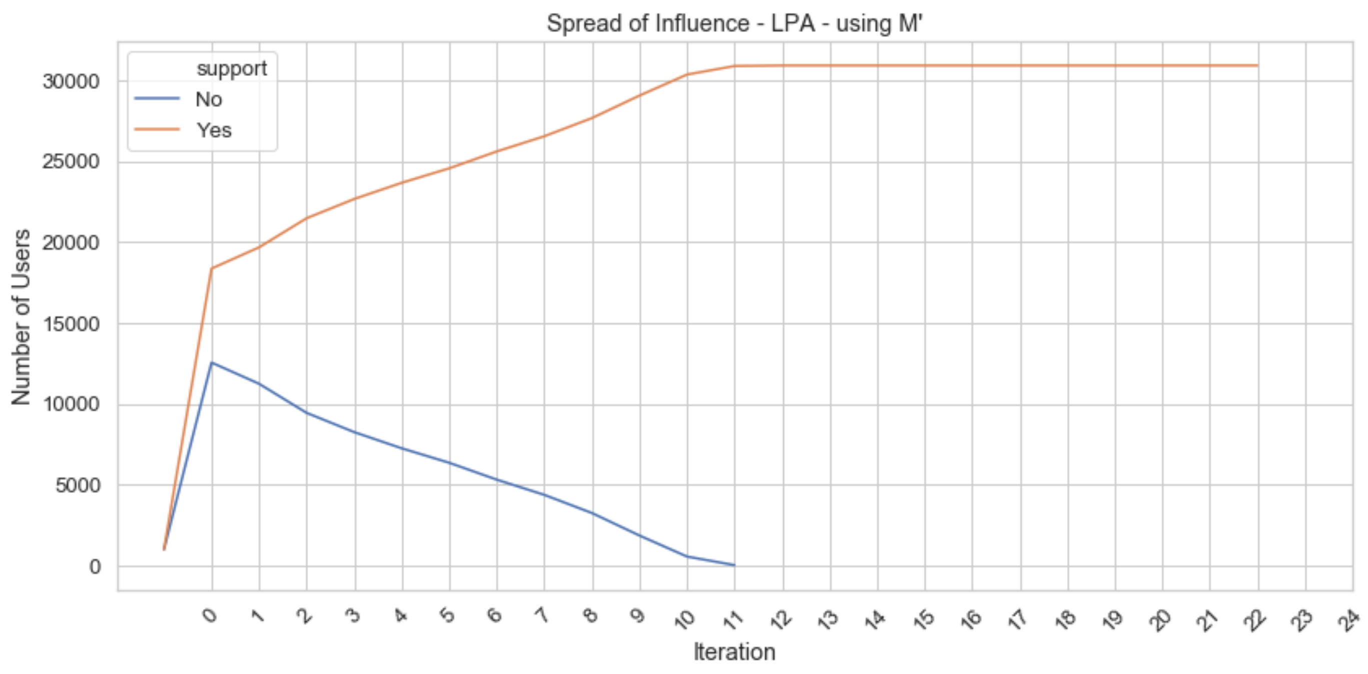
For the last step of our analysis we want to take a look at how is the spread of influence of each group. This can be seen as which group is more likely to influence the network through information sharing, in a way we can see this as which group is more likely to change the opinion of the other. To do that we developed two approaches, both of them are based on the label propagation algorithm.

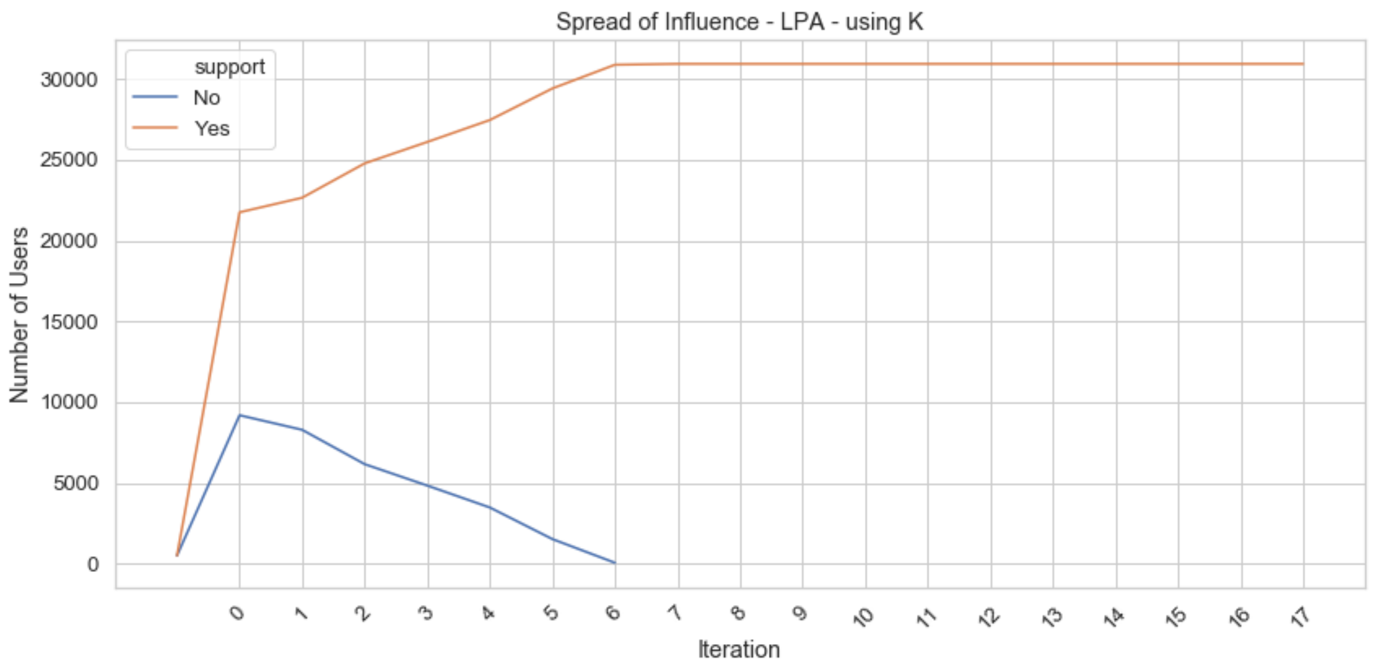
The first one, discussed in this section is the actual label propagation algorithm. We run the algorithm using three different seeds (initial labels). First using all the users identified as yes or no supporters (M - identified on section 2.1), secondly using the top 1000 hub users for each group (M' - identified on section 2.2), and finally using the top 500 key players for each group (K - identified on section 2.2).

The classical label propagation algorithm was proposed for undirected graphs, in our case we have a directed graph, so when deciding which label to assign to a user we look at the users that he follows and their current label, so if a user follows more users that are assigned a YES label he would be assigned a YES label, and so on. For the first iteration of the algorithm for all the cases we would have "Neutral" users, however this label is not propagated througout the network.

The following graphs show the distribution of labels throughout the iterations of the algorithm for each case. The algorithm would stop if no lables would change or if a maximum of 100 iterations was reached.







We can see on the first graph that since the number of seed labels for the NO supporters are higher they start by increasing their influence in the network. However at around the 3rd iteration the YES supporters start increasing their influence whilst the NO supporters start to decline. If we look at the other graphs, when we start with a balanced number of seed labels the YES supporters spread their influence much faster, in all cases, eventually the NO supporters almost disappear. The difference between M' and K is that using K the convergence is faster. This can be explained because a lot of the influencers who support YES are highly connected to the network and tend to influence more the spread then the influencers that support NO.

3.2 Spread of Influence using Modified Label Propagation

Finally, we propose our own version of the Label Propagation algorithm. The proposed algorithm work as follows:

1. Assign the number of clusters (in this case 2, Yes or No).
2. Define the initial clusters by using different seeds (as explained on 3.1)
3. Define initial centroids for each cluster (randomly select one node for the Yes supporters and one for the No supporters).
4. Update the clusters based on a metric Y and N. Assing the cluster Yes when Y > N, the cluster No when N > Y, and randomly choose one cluster when Y = N.
5. Update the centroids.
6. Go back to 4 or stop the iterations when no label change or if a maximum of 100 iterations is reached.

The metrics are defined as:

$$Y = 1.5N_y + 0.8Nn_y + 0.5C_y$$
$$N = 1.5N_n + 0.8Nn_n + 0.5C_n$$

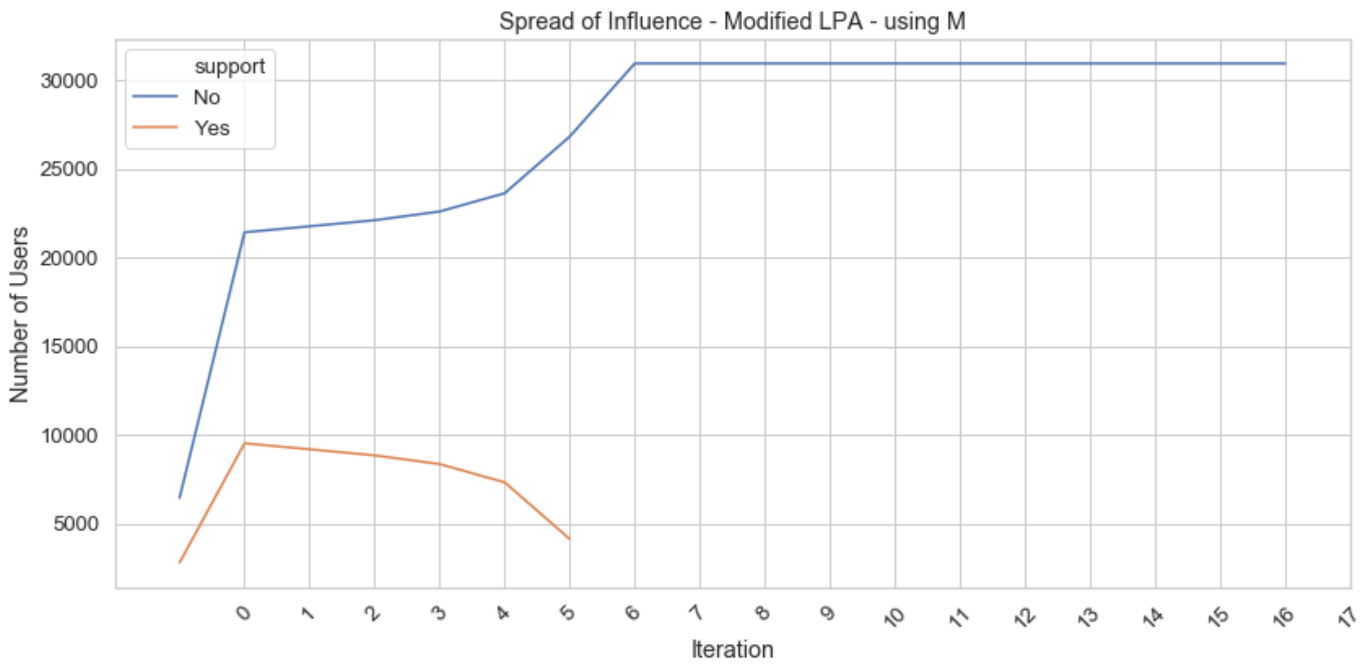
- N_y and N_n represent the number of neighbors with labels YES and NO respectively.
- Nn_y and Nn_y represent sum of the number of neighbors of each direct neighbor with labels YES and NO respectively.
- C_y and C_n represent the number of neighbors of the YES centroid and NO centroid with labeles

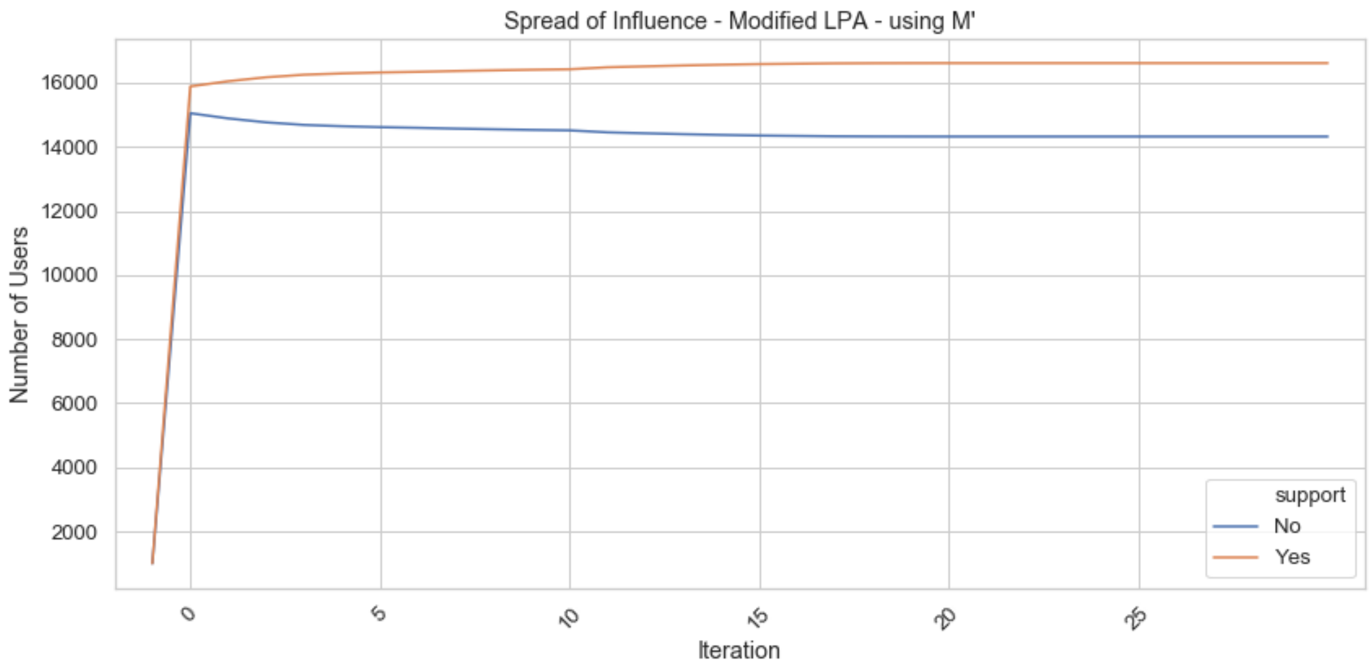
YES and NO respectively.

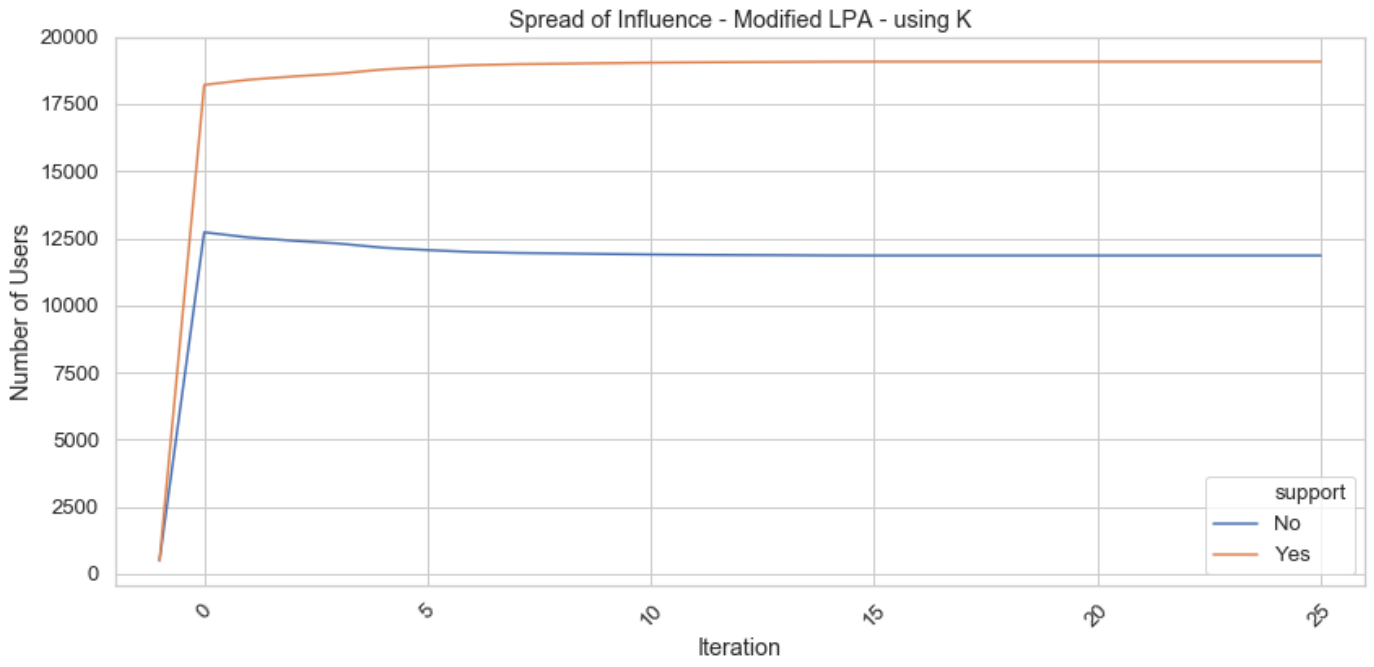
The update of the centroids is done by choosing the member of the cluster with the highest number of neighbors with the same label as him.

We can see this algorithm as a variation of the label propagation where besides looking at direct neighbors we also look at level 2 neighbors as well as to the node of the group who has more labels that coincide with his. The idea is to give a higher scores when the group is more connected between its nodes and the node is more connected to the group. We also decay the multiplying constant the further we look from the original node, thus giving a smaller weight to labels that are more distant from the node.

The following graphs show the distribution of labels throughout the iterations of the algorithm for each case proposed on section 3.2. The algorithm would stop if no lables would change or if a maximum of 100 iterations was reached.

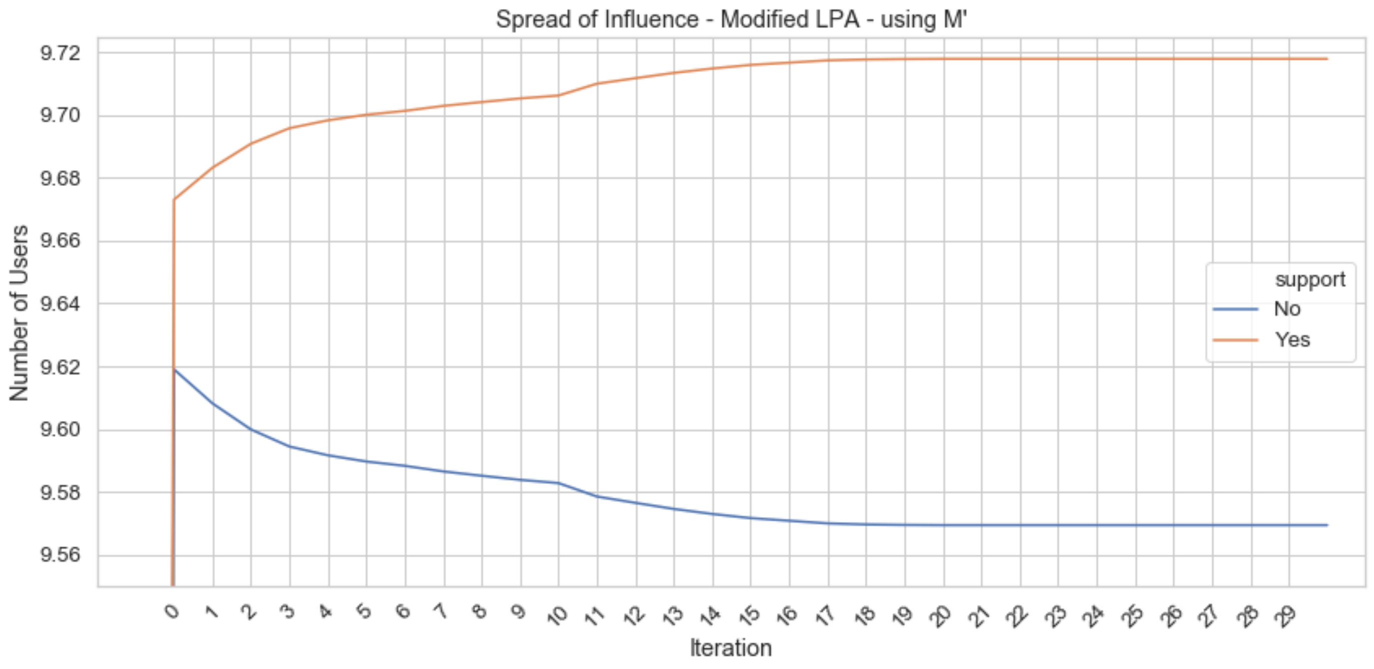


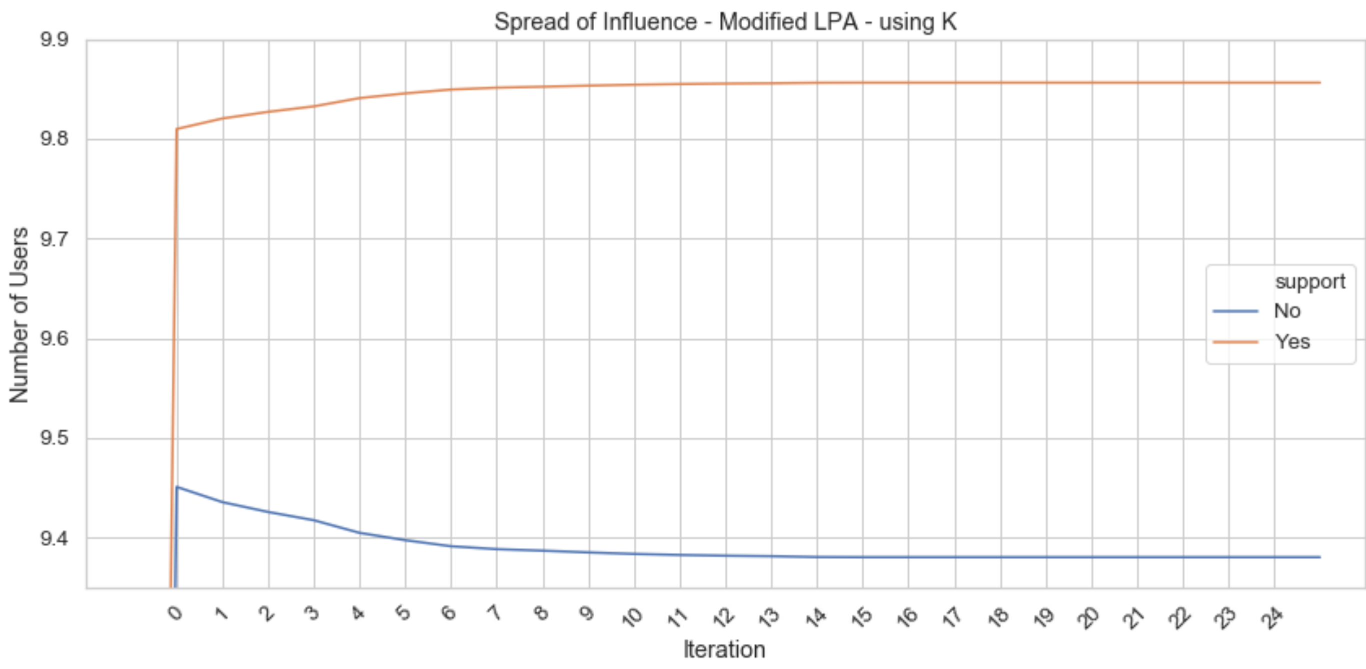




We can see that for the first case, the algorithm heavily benefits the cluster that initially has more members, therefore the NO cluster rapidly spreads to the network, and the YES cluster disappears after 5 iterations.

However, when we have a balanced seed the algorithm is much more conservative. For the M' and K case we can see that the YES cluster spreads more than the NO cluster, just like the original label propagation algorithm. If we take a closer look (graphs below), using the log of the number of members per cluster we can see that the algorithm converges at the 20th iteration for M', and at the 15th iteration for K. They are also similar to the original LPA, in the sense that it converges faster when using K. Also, for K, the number of users in the YES group is higher than on M' after convergence.





The proposed algorithm seems to represent a more realistic view of the spread of both groups, as both groups keep existing, but the one that have users more central to the network as a whole (YES group) are more spread than the other. Besides that, the algorithm also keeps the (nice) property of converging faster when we have a better seed.