Investigating relationships between Twitter latent topics and political orientation

Cognitive, Behavioral and Social Data

Authors: Natascia Caria, María Emilia Charnelli,
Francesco Ferretto, Matteo Lavina,
Andrea Sinigaglia

University of Padova
Department of Mathematics
Master Degree in *Data Science*

A.Y. 2019-2020

Contents

1. Introduction

Sharing political ideas through social media is increasingly popular, both among politicians and citizens. For this reason, social platforms may be considered as the main data resources for the investigation of people's political orientation. In this document we are going to discover which are the latent topics of the tweets related to different Italian political parties, through machine learning techniques.
The work has been divided into four main phases, data collection, data preprocessing, data analysis and topic modelling. In the following sections, we discuss about the way in which we collected the data and we describe the techniques and the algorithms used, showing the results and pointing out the pros and cons of our approach.

2. Data collection

The first phase of the work is data collection that, in our context, is to create a dataset of a sampling of twitter users with their associated political orientation. For this purpose, we have decided to collect data by going through profile on Twitter randomly.

We have written a Python script using the Tweepy library for accessing the Twitter API [6] and the random functions of the Numpy library [1].
In order to collect users, the program first takes an initial random list of $n = 100$ users from Italy meeting the following conditions:

- Minimun number of tweets: 10;
- Minimun number of following users: 15;
- Language of the last tweet: Italian;
- Last activity of the user: after 1/6/2019.

The initial list of users is obtained through the *search* method of Tweepy which returns a list of t tweets by geolocation, in this case from Italy. From this list of tweets, the program selects the users who wrote them and the ones meeting the conditions above. Those users are added to a so called *Selected users list* from which the program starts an iterative process. It chooses 10 users randomly from the *Selected users list* and from each one of them it takes 10 of their following users, randomly again. If those following users meet the previous conditions, they are added to the *Selected users list*. The random sample of users is obtained using the function *choice* of the random package in the Numpy library, with a parameter that allows choosing users in a list without replacement, so that we get different users with each iteration. This process is repeated several times until 1000 users are achieved.
Why going on selecting following users among the *Selected users list* instead of taking directly 1000 random users? The problem with choosing users at random is that it is very likely to catch bot users. The idea is that if the starting user is not a bot, then it is very unlikely that he follows a bot user, since it is supposed that he knows or likes to follow his friends.

The information related to the users selected with the Python script are finally stored in a Excel file. In particular, for each user we have registered the Twitter ID, the User Name and a link which has allowed us to directly access to the user Twitter profile.

2.1 Strategy to detect the party of the users collected

In order to record users political orientation, each person of the working group is required to detect which political party the users belong to. More in details, each member of the group assigns a label from 0 to 5 representative of a party according to the following legend:

0: Unidentified

1: Lega

2: Partito democratico

3: Fratelli d'italia

4: Movimento 5 stelle

5: Forza italia

The value 0 is assigned to the users whose political orientation was not identifiable or was none of the political parties above. We have established with the other working groups a temporal window in which analyze users' activity on Twitter; the period we have set goes from 1 June 2019 to 30 September 2019. The mark given by each member of the group is mainly based on the content of the users' tweets, the number of retweets related to that party and the content of the profile description.

Once all groups have collected the data, these have been aggregated into a unique dataset recording the following variables per user:

- Twitter ID: User's Twitter ID
- UserName: User's nickname
- Sex: Gender
- SID1, SID2, SID3, SID4, SID5: Political Party Labels
- Other info: Notes
- Class frequencies: Frequencies of labels
- TASK: Data collection method identifier

2.2 Tweets collection

Once users' matches were downloaded and tagged, users downloaded by all groups were unified to have a large dataset. Then, the task of downloading the tweets of previously collected users was divided between the groups.

The data collected by tweet was:

- `id`: id of the tweet
- `created_at`: Time of creation
- `favorites`: Number of favorites
- `retweets`: Number of retweets
- `source`: If the tweet was created using the android/iphone application or via web
- `full_text`: The content of tweet

3. Data Preprocessing

Data Preprocessing is an important step before apply a Machine Learning algortihm. It is a Data Mining technique which is used to transform the raw data in a useful and efficient format.

3.1 Users

In the User's dataset the variable gender was considered to have 10 different classes, and not 3 as expected: "M", "F" and "U" respectively standing for male, female and unidentified. Then we mapped each value to either "M", "F" or "U". Another thing that we noticed was that there were a lot of NaN values, in particular related to the variable SID5, however this was reasonable since there were groups with four participants instead of five.

But the main problem was that there are a lot of duplicated accounts. At the beginning, it seemed that 94.0% of the accounts had a unique Twitter ID, but then we noticed that there were groups that filled the column related to the variable Twitter ID with the symbol $'@'$ preceding the real Twitter ID and other not. Removing the symbol we discovered that actually the percentage of unique account was 88.5%. Then we removed the duplicates, taking into account only the first occurrence. Finally, we selected the accounts having a correspondent file with their own tweets in the folder were they were grouped.

3.2 Tweets

In the Tweets's datasets we only focus in the full_text variable that has the content of the tweets.

When operating with textual information, it is necessary to use Text Mining techniques to clean the unstructured data. This was achieved through a process comprising many stages. First of all, we put the words in lowercase and then we applied a stopwords filter, which filters the words that match any indicated stopword. We used stopwords of the Italian and the English languages like articles and pronouns, also we removed special characters, puntuations, stopwords of the social networks context such as smileys, greetings, etc; general words from the context of the Italy; words that refer to web page addresses; words that are between symbols, etc. Also we removed words with length less than 3 except for abbreviations of party names.

Once we have the tweets cleaned, the frequency of occurrence was calculated for each words, and words that appeared more than once were chosen.

4. Data Analysis

Data analysis is a process that involves inspecting data in order to highlight useful information, to suggest conclusions and support in decision making.

The cleaned dataset is made up of 6570 users. Setting a threshold of 100% of agreement, there are 2907 users labelled with a party different from 0; while lowering the threshold to 75% of agreement, those users are 3360.

Lega	PD	FdI	M5S	FI
1025	1069	408	776	82

(a) 75% Agreement

Lega	PD	FdI	M5S	FI
868	1002	312	660	65

(b) 100% Agreement

Table 4.1: Number of users per party

Table 4.1 shows the number of users per party after setting the two different thresholds. We can notice that the dataset is very unbalanced: the sample size is very small for party 5 (FI) compared to the others. Moreover there are not so many differences between the two tables, so we don't expect there to be large differences between the results obtained using the two different thresholds.

Figure 4.1 shows the number of users per party by gender variable, setting the agreement to 100% percentage. We can notice that for each party, there are more males than females.

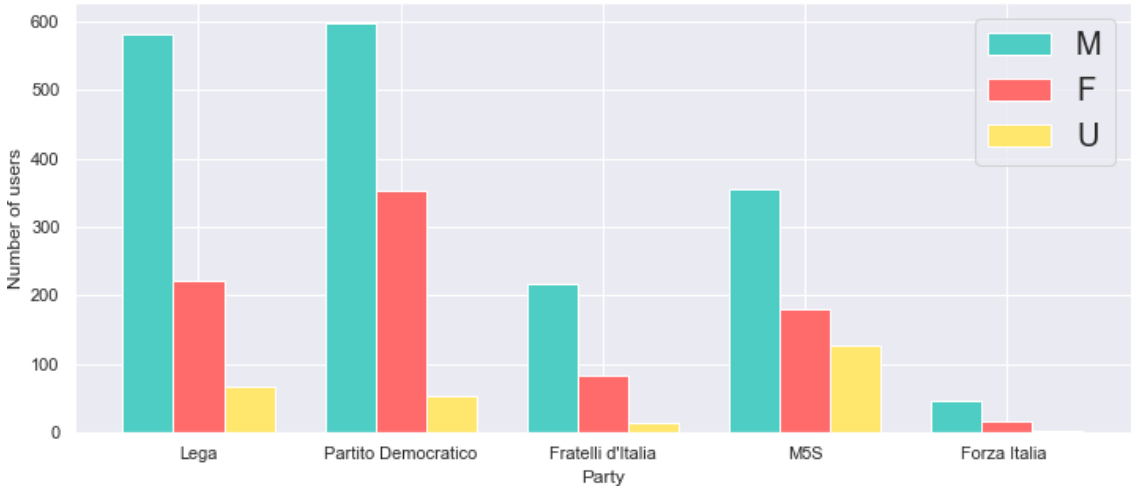

Figure 4.1: Number of users per party divided by gender

Finally, table 4.3 shows the number of tweets per party after the preprocessing phase. As we might expect, Party 3 (FdI) and Party 5 (FI) have fewer tweets compared to the other parties; instead Party 3 (M5S) has the higher number of tweets even if it is not the party with the higher number of users.

Lega	PD	FdI	M5S	FI
1,026,915	883,203	220,904	1,241,890	74,852

Table 4.3: Number of tweets per party

5. Topic Modelling

In many Natural Language Processing (NLP) applications, we face the problem of classifying documents according to the topics they speak about. Typically, topics are identified by finding the special words that characterize documents about that topic.

Topic modeling is a form of Text Mining, employing unsupervised and semi-supervised statistical Machine Learning (ML) techniques to identify patterns in a large amount of unstructured text.

In order to use a ML algorithm we need a way to represent the text in a numerical representation, identifyng characteristics and relations between the words. For the project we used the approaches TF-IDF and Doc2Vec to represent the tweets. Once we have those representations for every tweet we need an algorithm to do topic extraction. In this project we chose the K-means clustering algorithm.

5.1 TF-IDF

Articles that talk about baseball would tend to have many occurrences of words like "ball", "bat", "pitch" and so on. Once we have classified documents, to determine they are about baseball, it is not hard to notice that words such as these appear unusually frequently. However, until we have made the classification, it is not possible to identify these words as characteristic. Our first guess might be that the words appearing most frequently in a document are the most significant. However, that intuition is exactly the opposite of the truth. In fact, the several hundred most common words in any language (called *stopwords*) are often removed from documents before any attempt to classify them. In fact, *the indicators of the topic are relatively* ***rare*** *words.* The formal measure of how concentrated into relatively few documents are the occurrences of a given word is called TF-IDF (Term Frequency times In-verse Document Frequency), briefly, TF-IDF is an index that evaluates words that appear in the document but in general, are quite rare. It is normally computed as follows.

Suppose we have a collection of N documents. Define f_{ij} to be the frequency (number of occurrences) of term (word) i in document j. Then, define the term frequency TF_{ij} to be:

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

That is, the term frequency of term i in document j is f_{ij} normalized by dividing it by the maximum number of occurrences of any term in the same document.

Thus, the most frequent term in document j gets a TF of 1, and other terms get fractions as their term frequency for this document.

The IDF (Inverse Document Frequency) for a word is a measure of how much information the token or word in our case, provides. Suppose term i appears in n_i of the N documents (*tweets per party*) in the collection. Then

$$\mathrm{IDF}_i = log_2(N/n_i)$$

The TF-IDF score [4] for term i in document j is then defined to be $\mathbf{TF}_{ij} \times \mathbf{IDF}_i$.

The terms with the highest TF-IDF score are often the terms that best characterize the topic of the document.

In the figure 5.1, we can see an example of the TF-IDF Matrix obtained for 3 sentences.

	lot	love	program	python
i program in python	0.000000	0.000000	0.789807	0.613356
i love python	0.000000	0.861037	0.000000	0.508542
I program a lot in python	0.720333	0.000000	0.547832	0.425441

Figure 5.1: Example of TF-IDF text representation

5.2 Doc2Vec

In general, when you like to build some model using words, techniques like TF-IDF are acceptable approaches. However, when using such encodings, the words lose their meaning. Doc2Vec is an unsupervised method that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents.

Doc2Vec's learning strategy exploits the idea that the prediction of neighboring words for a given word strongly relies on the document also. For example, if the word "catch" is frequent in the corpus and we know that the topic of a document is about programming, we can expect words such as bug or exception after the word "catch" but if the topic of the document is about sports, then we can expect ball after catch.

The task is to predict a context given a word. In this framework, every word is mapped to a unique vector, represented by a column in a matrix W. The column is indexed by the position of the word in the vocabulary. The concatenation or sum of the vectors is then used as features for the prediction of the next word in a sentence.

Doc2Vec uses The Distributed Bag of Words version of Paragraph Vect (PV-BOW), presented in [2], this algorithm forces the model to predict words randomly sampled from the paragraph in the output. In reality, what this means is that at each iteration of stochastic gradient descent, we sample a text window, then sample a random word from the text window and form a classification task given the Paragraph Vector. The strategy of PV-BOW is shown in Figure 5.2.

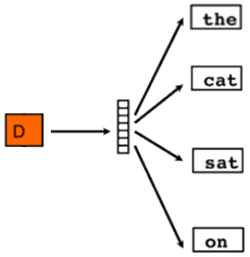

Figure 5.2: Distributed Bag of Words version of paragraph vectors

The algorithm of Doc2Vec is a neural network whose architecture is:

$$doc2vec(x) = (W_2 D_1 \mathbf{x})$$

D_1 is the document embedding matrix of $m \times d$ where m is the embedding size and d is the total number of documents. W_2 is the matrix that performs a linear mapping between the document embedding space to the word space. Finally the softmax function is applied to obtain a multinomial probability distribution of words. The Doc2Vec algorithm minimize the cross entropy loss function between `CrossEntropy`(docvec(x), y) where y is the sampled words for a given document. The weight matrix are learned using stochastic gradient descent.

After the training converges, the word vectors are mapped into a vector space such that semantically similar words have similar vector representations (e.g., "strong" is close to "powerful"). Also, words that are related, like "engine" and "transmission", would probably have similar contexts as well.

5.3 *K*-means

K-means clustering is a simple and elegant approach for *partitioning a dataset into K distinct, non-overlapping clusters.*

To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters. The K-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation.

Let $C_1, ..., C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup ... \cup C_K = \{1, ..., n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_i \cap C_j = \varnothing$ for all $i \neq j$. In other words, the clusters are non overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$. The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible.

The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other. Hence we want to solve the problem:

$$\underset{C_1,...,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W\left(C_k\right) \right\}$$

In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

Solving the above mathematical problem seems like a reasonable idea, but in order to make it actionable we need to define the within-cluster variation. There are many possible ways to define this concept, but by far the most common choice involves *squared Euclidean distance.* That is, we define:

$$W\left(C_k\right) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

where $|C_k|$ denotes the number of observations in the kth cluster. In other words, the within-cluster variation for the kth cluster is the sum of all of the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations in the kth cluster.

Now, we would like to find an algorithm to solve the optimization problem of finding the partitioning of the observations into K clusters such that the optimization problem combined with the squared Euclidean squared is minimized. This is in fact a very difficult problem to

solve precisely, since there are almost K^n ways to partition n observations into K clusters. This is a huge number unless K and n are tiny. Fortunately, a very simple algorithm can be shown to provide a local optimum (a pretty good solution) to the K-means optimization problem that follows:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

This approach is laid out in the following Algorithm:

Algorithm K-means

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster centroid. The observations in the kth cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).

5.4 Choosing the best number of topics

K-means requires the number of clusters as a parameter. One simple approach to calculate the optimal number is to run the algorithm with different cluster sizes and compute the sum of squared errors (SSE) for every model obtained and then choose the best parameter using the elbow strategy.

The idea is to plot the SSE obtained with the different cluster sizes and if the line graph looks like an arm, the "elbow" on the arm is the optimal value of the number of clusters. The goal is to minimize the SSE choosing a small value of cluster size where the error metric begins to level off.

SSE tends to decrease toward 0 as we increase the cluster size, SSE is 0 when the cluster size is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster.

$$SSE = \sum_{i=0}^{n} \min_{\mu_j \in C} (\|x_i - \mu_j\|^2) \tag{5.1}$$

6. Results

Before exposing the best results obtained per method we're going to make some prior observations. Firstly, in order to get the topics we've decided to do topic extraction by party for many reasons:

- The initial dataset is *unbalanced* and it is not convenient to balance it because there would lose a lot of information;
- With the application of one model on the total dataset, we would obtain topics with too much general meaning.

Another aspect in which we focus is that running the algorithms considering 75% and 100% percentage of agreement led to similar results. For this reason, we are going to show only the results of 100% of agreement. In order to get the best number of topics, we run K-means for both approaches (TF-IDF and Doc2Vec) setting a minimum of 2 and a maximum of 20 topics. Once we get the optimal number of clusters we run again the algorithm which gives us the topic associated with each tweet per party.

6.1 TF-IDF and K-means approach

The first approach was to apply TF-IDF and K-means algorithm, which involved the use of scikit-learn library [3] in Python. The number of clusters per party is displayed below:

LEGA[1]	PD[2]	FDI[3]	M5S[4]	FI[5]
14	14	16	12	14

Table 6.1: Topics per party obtained

For instance, figure 6.1 shows that the best number of topics considering SSE error metric for party 5 is 14.

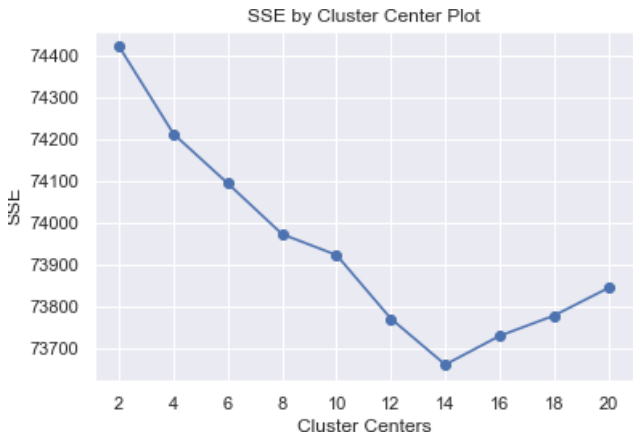

Figure 6.1: SSE obtained for party 5 using TF-IDF

Cluster	Top terms
Cluster 0	vita, paese, lega, mondo, politica, italiani, ministro, storia, europa, donne, tweet, partito, uomo, milano, legge
Cluster 1	renzi, matteo, salvini, governo, partito, zingaretti, enews, calenda, italiaviva, politica, sinistra, berlusconi, viva, leopolda, conte
Cluster 2	salvini, matteo, lega, ministro, meloni, milioni, italiani, savoini, soldi, migranti, dimaio, politica, berlusconi, interno, conte
Cluster 3	sardine, politico, piazza, fatti, modena, salvini, modenanonsilega, movimento, bologna, pagina, genova, piazze, palermo, facebook, parma
Cluster 4	leghista, solidarietà, segre, cittadinanza, reddito, liliana, sindaco, senatrice, onoraria, scorta, commissione, odio, biella, razzismo, meloni
Cluster 5	facciamorete, sardine, fbpe, siamotanti, salvini, facciamoinformazione, modenanonsilega, iostoconlesardine, nessunotocchilesardine, restiamoumani, facciamoeuropa, farlo, lega, salvinidimettiti, governo
Cluster 6	presidente, consiglio, mattarella, repubblica, conte, commissione, parlamento, governo, regione, salvini, eletto, europeo, quirinale, senato, ministro
Cluster 7	roma, raggi, piazza, città, rifiuti, manifestazione, sindaco, milano, marcia, capitale, ottobre, centro, comune, salvini, napoli
Cluster 8	porta, capitone, tranquillo, salvini, salva, sfiga, voti, aperta, milano, napoli, italiani, lega, renzi, piazza, vita
Cluster 9	ovviamente, squadra, parlando, finita, pacchia, napoli, juve, partita, calcio, giocatori, gioco, allenatore, campo, tifosi, partito
Cluster 10	figli, scelta, pensavo, gruppo, libri, genitori, vita, donne, storia, politica, famiglia, futuro, libertà, padri, nipoti
Cluster 11	twitter, facebook, tweet, amici, instagram, followers, ilmioanniversarioditwitter, mentions, facciamorete, follower, retweet, profilo, trump, retweets, salvini
Cluster 12	governo, salvini, lega, conte, crisi, cadere, paese, maio, cambiamento, opposizione, italiani, ministro, voto, parlamento, elezioni
Cluster 13	maio, conte, salvini, luigi, ministro, zingaretti, giuseppe, ilva, esteri, governo, lega, umbria, grillo, stelle, battista

Table 6.2: Top terms per cluster Party 2

For each party we get the clusters with the top terms associated with them. For instance, Table 6.2 gives the top-15 words for each cluster of Party 2 (PD). For most of them we are able to identify a topic. In particular:

- **Cluster 3**: Political Movement "Le sardine". It is a protest movement against right-wing politics (populism and sovereignty) that arose in November 2019.
- **Cluster 4**: The problem of the anti-Semitic attacks against Liliana Segre and the honorary citizenship not conferred to her by the mayor of Biella.
- **Cluster 5**: Opposition against right-wing politics, mainly represented by the hashtag #facciamorete which has spread to social media since January 2019, but also by the movement "Le sardine" again.
- **Cluster 7**: Rome, its administration by Virginia Raggi and the problem of rubbish.

- **Cluster 9**: Football.
- **Cluster 10**: Family.
- **Cluster 11**: Social.
- **Cluster 12**: The government crisis happened in August 2019.
- **Cluster 13**: The political discussion against Ilva.

There are a lot of clusters whose most frequent terms refer to politics in general, however it is difficult to assign them a specific topic. We are now going to analyse how the frequency of the topics changes over time. Since most of the detected clusters mainly refer to recent political events, we focus on the following time windows: December 2018 - March 2019 - June 2019 - September 2019 - December 2019. The distribution of the topics and their trend over time can be seen in Figure 6.2. We can notice that Topic 3 (dark green area) appears just before December 2019. This may be a confirmation of our correct interpretation because, as we have already said, Cluster 3 refers to the political movement "Le sardine" which arose in November 2019. Instead, Topic 5 (red area) appears more often between December 2018 and March 2019 probably due to the widespread use of the hashtag `#facciamorete` in this temporal window on social networks. Moreover, we can notice that Topic 0, which we are not able to detect what is about, is every time the topic with the highest frequency, probably because it refers to very general political arguments.

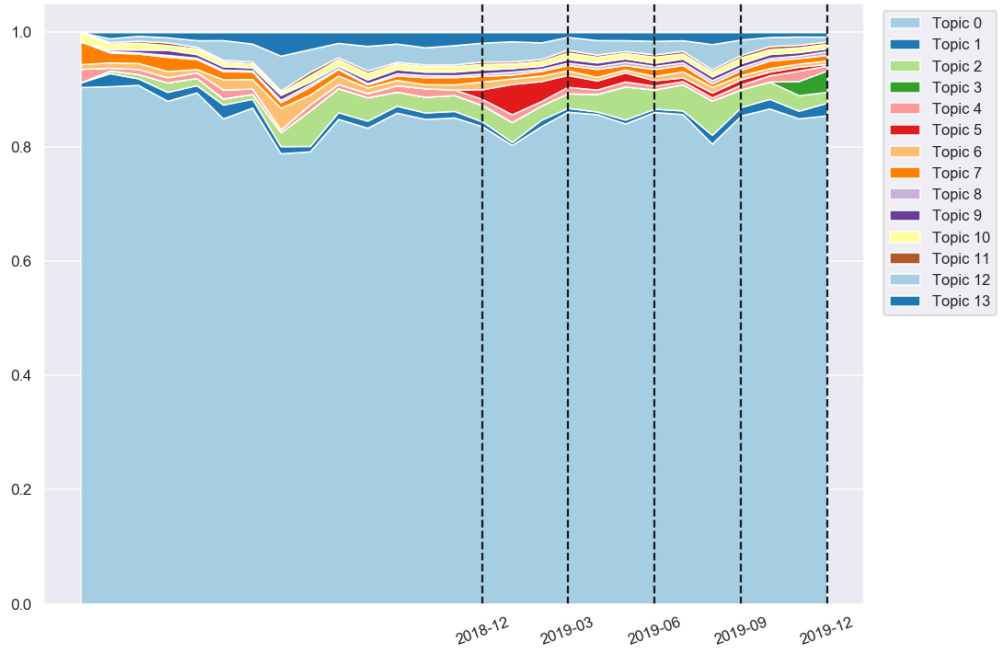

Figure 6.2: Analysis of the temporal trend of the topics of Party 2 (PD)

6.2 Doc2Vec and K-means Results

As a second approach we used Doc2Vec algorithm, involving the gensim [5] library in Python. Using Doc2Vec we obtained a small numbers of topics compared to TF-IDF. The table below displays the best number of clusters per party with the Doc2Vec application:

LEGA[1]	PD[2]	FDI[3]	M5S[4]	FI[5]
6	7	4	7	5

Table 6.3: Topics per party obtained

Cluster	Top terms
Cluster 0	germania, mentre, immigrati, port, africani, clandestini, milioni, francia, lampedusa, donne, vogliono, soldi, italia, milioni, nave
Cluster 1	elezioni, mattarella, presidente, soldi, italia, legge, stopmes, bisogna, votare, politica, tasse, voto, parlamento, partito, ministro
Cluster 2	segre, vita, destra, vero, odio, schifo, sardine, nessuno, qualcuno, basta, politica, bibbiano, nulla, marcogervasoni
Cluster 3	umbria, piazza, capitano, sera, footam, matteo, legasalvini, ministro, amici, stelle, diretta, ieri
Cluster 4	famiglia, buon, giorno, venezia, cuore, claudio, vita, napoli, mondo, massimo, bella, juventusfc, chiossiemanuela, petraromano
Cluster 5	matteorenzi, movstelle, porcipolitici, laveritaweb, giuseppeconteit, tramite, cazzo, vero, nominodearjack, chiossiemanuela, mtvema, detto, stopmes, piaciuto, albertobagnai

Table 6.4: Top terms per cluster Party 1

For each party we get the clusters with the top terms associated with them. For instance, Table 6.4 gives the top words for each cluster of Party 1 (Lega). For most of them we are able to identify a topic.

- **Cluster 0**: A main topic for this party is immigration and in this cluster we can observe that all the terms are related to this topic.
- **Cluster 1**: This cluster is about the topic of StopMes, a economic strategy sustained by the Lega party. The MES, defined as European Stability Mechanism (ESM), is an European fund, designed to be shared by only a specific subset of the European nations and discussed very frequently by the politicians of the Lega Party.
- **Cluster 2**: The interpretation for this cluster is that it expresses the common feeling of hate towards the left wing of the political Italian scenario.
- **Cluster 3**: This probably refers to the social activities and references to the advertising campaign of the leader of the Lega Party, Matteo Salvini.

We notice that in this example 2 topics could not receive a precise interpretation (last two ones), even if they seemed to speak about politics, but in a general way.

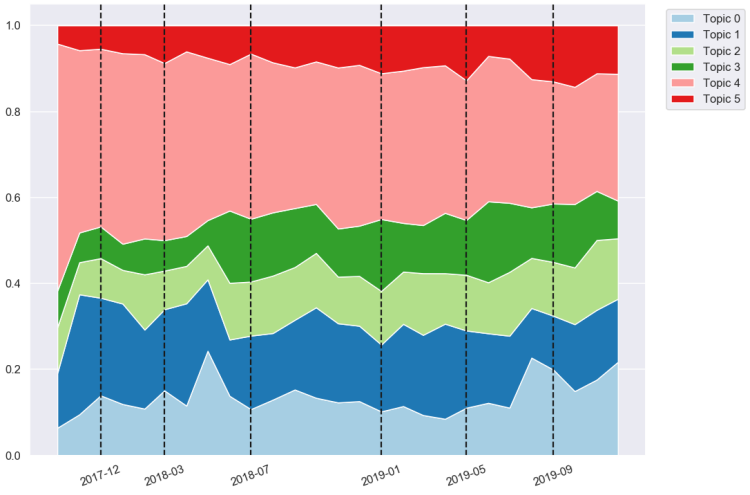

Figure 6.3: Analysis of the temporal trend of the topics of Party 1 (lega)

7. Conclusions

Before speaking about the conclusions made on the results, we want to introduce our judgement criteria concerning the topics, or briefly, a shared qualitative metric of evaluation of them.

In particular we thought that there could be at least two criteria:

- AN ABSOLUTE CRITERIA

  Emphasizing an higher level of specificity on the overall topics detected by the algorithm. So without having, necessarily, the same fraction of topics with the same level of specificity per party.

- A RELATIVE CRITERIA

  Emphasizing the obtainment of the same level of specificity among the parties, or briefly, a metric that takes into account the evaluation of well distributed level of specificity among the parties. A scale focused more on homogeneity of the results.

The second metric was introduced to evaluate some results that had an homogeneous level of specificity among the parties, but still not a considerable value of specificity.

- **TF-IDF** and **K-means approach**:

  Initially[1], with the application of the TF-IDF and the K-means algorithms we obtained a high number of topics extracted, even if the goodness of them, clearly, wasn't so good as in the *context-based* approach (`Doc2Vec`).

  This revealed the limitation of syntactic methods since the first approach, even setting a fitted stopwords' set, led to topics with a high degree of free interpretation[2], and we suppose that this is due to multiple reasons:

  - **Politics concerns about highly dynamic and specific topics**, and the application of the TF-IDF and K-means, we suppose, couldn't get all of them because of their *volatility* (*i.e.* short life the topics, other term to express the dynamism of the domain), and the algorithm could obtain just a collection of topics made up by the union of more specific ones, that together behaved well in average in the long term. This last statement is a shared aspect that can be viewed also by the `doc2vec` application, but with a different shade.
  - The **nature of the approach**. The intuition is that the syntax of the text that refers to a topic not always can reveal a precise semantic meaning, thinking about idiomatisms in Italian, that rarely can be interpreted precisely by the syntax.
  - The **context of the application**. The problem with the syntax of tweets is that they are informal and usually do not respect all grammar rules. In addition we have tweets that do not talk about politics that negatively affected the results or have topics that are mixed with similar tweets that are about politics and this generates noise in the results.
  - The **lack of any initial feedback on the goodness of the stopwords' set**, as a measure of its fitting at the domain of interest's application. It is also important to mention the need of a shared judgement on the choice of the stopwords by Italian native speakers, like the important words and the possible themes that can or cannot be avoided in politics.[3]
  - The **lenght of the observational time** of the topics in this domain.
  - The quality of user's classification on a party **depends on a subjective assessment** and a task that was performed manually.

[1] The following considerations are made on the first execution of the program

[2] With **degree of free interpretation** we meant the number of possible ways in which a topic could be fitted (contextualized) inside a more specific one. You can think it as a quantity analogous to the generality of the topic.

[3] Furthermore, with a feedback we were been able to see the impact of the elimination of some other stopwords on the performance of the two approaches.

We highlight that the fact the latter considerations are made on the basis of the first results obtained with this approach.

In a second run, looking at our analysis the only aspect that we could modify, with the aim of seeing any improvement, was to review the definition of the stopwords' set. The reason is explained by the presence of a considerable fraction of topics revealed that, had no sense at all (*politically speaking*) and contained some words that we missed filtering the potential stopwords from the corpus. So we updated it.

The results (shown in the previous Chapter) improved, showing a ***positive relationship*** between the *efficacy* of the syntactic method's analysis in the politics' domain and the *robustness of a correct filter* applied onto the corpus, respect to the second criteria of evaluation. Indeed the number of topics found by this approach didn't change so much after the second pre-processing of the datasets, but the quality and the interpretation of them got better, showing a significant growth of the number of specific topics per party (*e.g.* at least $\frac{8}{13}$ of the topics for the party 2 could have been interpreted) .

- **Doc2Vec** and **K-means approach**

  In a first analysis, we obtained a smaller number of topics respect to the *syntax-based* algorithm. What we found surprising was the strong difference between the quality of the results respect to the first approach (at least in the first attempt).[4]

  Secondly, after the second pre-processing, we encountered an unexpected result. The algorithm, performed better respect to the second criteria (homogeneity of the results among the parties) but the maximum level of specificity decreased in a noticeable way. We mention that in the second pre-processing phase we removed words that were twitter ids, common words and not relevant words (at least in our opinion) for the political domain. The impact of this kind of removal suggested a possible ***negative relationship*** between the *efficacy* of the context-based method's analysis in the politics' domain and the *robustness of a correct filter* applied onto the corpus, respect to the first criteria of evaluation.

7.1 Final reflections

Finally, we share the belief that the observational time was too long for our application in the domain of interest (for both the methods' types), and considering a small interval of time might have been essential in order to detect topics with a precise meaning (looking for some matches with real events) in order to see the limitations of each of the methods in this kind of domain.

All the considerations that were done are not supported by any statistical analysis of the results, but just some shared belief about them. Showing some paths to deeper analysis.

Finally, other technical issues that could have been considered to obtain the topics are:

- There are a lot of parameters that we can use to run Doc2Vec, maybe exists a possible model that offers better results.
- Another strategy is taking into consideration the numbers of retweets that each tweet has and use the most important tweets to do topic extraction.

[4]The results of the application of the mentioned methods are displayed in the previous Chapter.

- Also, there are other metrics that could have been used in the project to obtain and validate the topics.

Bibliography

[1] NumPy developers. Numpy Library. `https://numpy.org/`, Copyright 2019. [Online; accessed 01-October-2019].

[2] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Anand Rajaraman, Jure Leskovec, and Jeffrey D. Ullman. Mining massive datasets, 2014.

[5] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[6] Joshua Roesslein Revision. Tweepy Library. `https://www.tweepy.org/`, 2009-2019. [Online; accessed 01-October-2019].