Twitter Sentiment Analysis and Topic Discovery

Mathieu Pont, Lucas Rodrigues Pereira
Anass Lahrech and Vanna Boungnalith

April 2019

Abstract

In recent years, social networks have enabled people to express themselves and share their opinions regarding different topics. Through short messages, we can detect sentiments on a specific topic. But also, we can discover different aspects of the topic and which feelings are expressed towards this aspect. The data provided by social networks can be analyzed with sentiment analysis (or opinion mining) which is a Natural Language Processing (NLP) method aiming to find the sentiment expressed in many ways, such as in a speech or a written text. In this study we want to analyze, through sentiment analysis and topic modeling, the public opinion related to "abortion".

1 Introduction

Social networks can be a huge database when the data are free to use. This is the case of Twitter that provides us an API to easily collect tweets related to certain keywords.

In the simpler form, the sentiment found after a sentiment analysis is rather positive or negative (or even neutral), but it can be more complex like a rating (from 1 to 5 with 1 being the most negative and 5 the most positive, for example).

There are two main approaches to do sentiment analysis, the learning-based and the lexicon-based. The first one uses machine learning algorithm as classifiers. The algorithm learns directly from the data and tries to separate the positive and the negative documents (if we are in this form of sentiment analysis). The other one, the lexicon-based, uses an external and predefined dictionary to find positive and negative terms in the text and reveal as a result its global sentiment.

Another interesting method in NLP is topic modeling, which tries to find the topic of a document. Actually, it separates a collection of documents in groups, and the documents of a same group should talk about the same topic.

These methods can then use unsupervised learning (or clustering), the process of partitioning a dataset in different groups having similarities, without the help of an oracle (a "teacher" saying the answer, i.e. the real group of each data).

Through this work, we seek to compare the different sentiment analysis approaches (lexicon-based against learning-based). As we are in a unsupervised context we have the purpose to evaluate the different unsupervised algorithms in the context of the learning-based approach.

As a text needs to be encoded in a vector to be analyzed by such algorithms, we finally want to try many documents representation to compare them in our context.

2 Related Work

This work is mostly inspired by [Karami et al., 2018], which analyzed the United States' 2012 presidential election through mining people's opinions in Twitter. Using lexicon-based approach for sentiment analysis (classification as positive or negative opinion), the authors grouped tweets by topics by candidate. As a result, the authors were able to classify the overall public opinion regarding the five main economic issues of the election: Economy in General, Job, Budget Deficit, Healthcare, and Tax. Some methods were also used in this work and will be briefly presented in the following section.

[Unnisa et al., 2016] used unsupervised learning techniques to analyse approximately 2000 tweets on movie reviews. The texts were tokenized, to only keep relevant terms, based on a feature vector created by the author. The feature selection was performed based on the frequency of occurrence of words, if they were considered

to be a negation, etc. Machine learning algorithms were used to create the clusters, such as K-means, Hierarchical clustering and Spectral clustering.

Another interesting article regarding the survey on mining subjective data on the web is [Palpanas et al., 2011]. The authors presented an overview on methods of Subjectivity Analysis. Some state of the art methods described in the article were studied and used in the present work, such as the point-wise mutual information (PMI) criterion for statistical dependence.

In [Joshi et al., 2018] they classified sentiments from tweets using machine learning and Natural Language Processing algorithms. The data were preprocessed as it comes with mentions, hashtags, URLs and emojis. The features used to make classification are Unigrams and Bigrams. Only the top 15000 unigrams and the top 10000 bigrams were kept to avoid those which are noise. In this work, the Baseline algorithm, Naive Bayes Formula and Maximum Entropy Classifier model were used to classify sentiments of tweets.

Thanks to the work done in [Novak et al., 2015], each emoji is assigned a sentiment from all the tweets in which it occurs. The authors form a discrete probability distribution which denote the neutrality, negativity and positivity of the emoji. Thereafter, a sentiment lexicon of 751 most frequent emoji is created. On average, the positions of emojis were on the 2/3 length of a tweet. The positions helps to increase the negativity or the positivity of the tweets wherease it decreases the neutrality. The authors made a sentiment bar which is a way to visualize the sentiment attributed to an emoji.

In [Hu et al., 2013], the authors go beyond the emojis and use every *Emotional signals* which are any information that could be correlated with sentiment polarity of a document or the words in the document.

3 Methodology

3.1 Preprocessing documents

Some preprocessing operations are needed to correctly use the documents for sentiment analysis. The first step to be done is to remove the duplicated tweets, so they do not skew the results by giving them more importance.

In order to increase the quality of the results we want to remove words that are not important for the classification task.

We search words that doesn't have any semantic values like the word *"the"* or *"you"* for example, this kind of words are called *stop words*. We can also remove punctuation, numbers and put each word in lower case. Another preprocessing called *lemmatization* tries to change each word into it's word stem. For example the word *"gone"* will become *"go"* . We removed the words that appears less than 10 times across all the documents, indeed if a word is very rare then it's probably not important.

During preprocessing, hashtags and emojis are converted to token words, in order to keep its semantics in the final result. We also removed the *RT* on the beginning of the tweets if it is retweeted. As far as emojis are concerned, they are used to express what we feels about something in an easy way. To take benefit of that in our work, we convert emojis to their Unicode names using the *demojize* function from the *emoji* Python library so we can match it thereafter with the sentiment lexicon.

3.2 Encoding documents as vector

Bag of words. One of the simplest encoding is the *bag of words*, it's simply an histogram of the words present in each document. The output is a matrix $\mathbf{X} \in \mathbb{R}_+^{n \times t}$ where n is the number of documents and t the total number of words within all documents. x_{ij} is the number of times the word corresponding to the column j appears in the document i. We call this matrix the document-word matrix.

TF-IDF score. Another method, based on the *bag of words*, is the *TF-IDF score* that stands for *Term Frequency and Inverse Document Frequency*. This score is computed for each word in each document. It outputs a matrix $\mathbf{X}$ with the same properties than the *bag of words* except that we multiply the frequency of the word in the document by a factor depending on the frequency of the word across all the documents like in equation 1.

$$x_{ij} = tf_{ij} \times \log \frac{N}{df_j} \tag{1}$$

Where tf_{ij} is the frequency of word j in document i. N the total number of documents. df_j is the number of documents containing the word j. Multiplying the second factor to the frequency of the word allows us to give more importance to word that are rare across all documents. Indeed words that occurs frequently across documents will not help us very much to discriminate them.

Finally we can apply L2 normalization to make each vector unit length, i.e. its norm equals 1.

Doc2Vec. Doc2Vec, an unsupervised machine learning method presented by Mikilov and Lee in 2014, is an extension of the Word2Vec, and cannot be explained independently. Therefore, the latter should be briefly dis-

cussed before actually tackling the embedding of documents into a vector.

In a nutshell, Word2Vec is the representation of words as vectors in space, encapsulating their relations to each other, such as synonyms, antonyms or analogies. This representation is created using either Continuous Bag-of-Words (CBOW) or Skip-Gram. Both methods have different approach to the same issue: they create search to predict the missing word given a context (usually called window). As the model is trained using a neural network, the weights of the hidden layer encapsulate the relation between the context and the output word.

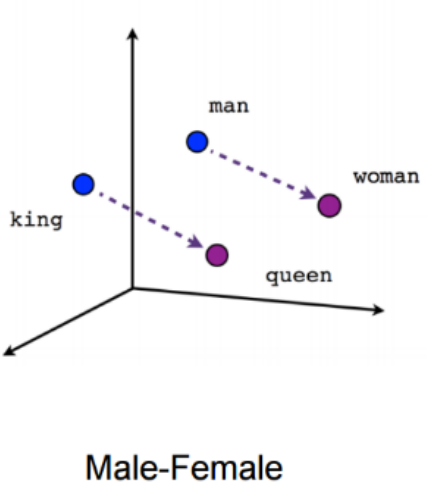

Figure 1: Example of relationship encapsulated in Word2Vec embedding.

As [Le et al., 2014] proposed, the Doc2Vec extends this method by adding an *id*, which is unique for each document, to be embedded along with the words. This gives each document a single feature vector regardless of its length (number of words in the document). This result can be used for clustering using traditional unsupervised methods, such as K-Means.

Density Matrix Representation. This method was first introduced by [Zhang et al., 2018] based on the work of [Gonçalves et al., 2013]. It uses quantum probability theory to represent a document as a density matrix. In quantum theory this matrix describes the state of a system, it can be seen as a probability distribution over the different possible states of the system.

A quantum state $\boldsymbol{u}$ is a vector and can be represented as a ket $|u\rangle$ and $\boldsymbol{u}^T$ as a bra $\langle u|$ (quantum bra-ket notation). Hence the density matrix is expressed as the following:

$$\rho = \sum_i \phi_i |u_i\rangle\langle u_i| \tag{2}$$

where ϕ_i is the probability to be in state $|u_i\rangle$.

The product $|u\rangle\langle u|$ is called an event and is represented by an orthogonal projector $\Pi = |u\rangle\langle u|$. Notice that $|u\rangle$ is a column vector therefore its corresponding event Π is a matrix.

Here, a document is seen as a system and each word of the document as state. The quantum state vector of each word correspond to its one-hot encoding. We can then define the projector for each word and hence the sequence of projectors for each documents.

The density matrix of each document is estimated with the Globally convergence based Quantum Language Model (GQLM) algorithm presented in [Gonçalves et al., 2013]. They use Maximum Likelihood estimation to train density matrix with gradient descent by maximizing the product of the probability of each event (or projector).

3.3 Sentiment Analysis

3.3.1 Lexicon-based Approach

For the lexicon-based approach we found SentiWordNet [Baccianella et al., 2010], this is a dictionary composed of a lot of words and each of them is associated with a positive sentiment and a negative one. The dictionary can be used to find the global sentiment of a text using the sentiment of each word.

We go further than using only a word lexicon, we use also a sentiment lexicon. It allowed us to have a better lexicon-based approach. Each emoji is associated to a sentiment. To do so we used the occurrence of an emoji in a positive, negative, or neutral tweets to produce a sentiment vector *(positive, negative, neutral)*. For example, for the emoji 😂 the sentiment vector resemble to this *(0.468, 0.247,0.284)* which means that it expresses a positive feeling most of the time.

3.3.2 Learning-based Approach

We are in our case in a unsupervised context because we don't have the label of the tweets. They are many unsupervised algorithms in the context of opinion mining on twitter data, some of them are presented in [Unnisa et al., 2016]. For example K-Means that use the Euclidean

distance to group documents by similarities in k clusters. It uses Euclidean distance, a variant of it, called Spherical K-Means uses the cosine distance.

Another unsupervised algorithm is the Word Co-Occurence Non-negative Matrix Tri-Factorization (WC-NMTF) [Salah et al., 2018]. This method is based on Non-Negative Matrix Tri-Factorization (NMTF) [Long et al., 2005] and [Ding et al., 2006] which factorizes an input matrix $\mathbf{X} \in \mathbb{R}_+^{n \times t}$ into three matrices $\mathbf{Z} \in \mathbb{R}_+^{n \times g}$, $\mathbf{S} \in \mathbb{R}_+^{g \times m}$ and $\mathbf{W} \in \mathbb{R}_+^{t \times m}$ such that all matrices don't have any negative values and:

$$\mathbf{X} \approx \mathbf{ZSW}^T \tag{3}$$

$\mathbf{Z}$ plays the role of the cluster membership of each document if we have g clusters. The Word Co-Occurrence Non-Negative Matrix Tri-Factorization (WC-NMTF) [Salah et al., 2018] uses the same factorization but also factorizes another matrix $\mathbf{M} \in \mathbb{R}_+^{t \times t}$ into the matrices $\mathbf{W} \in \mathbb{R}_+^{t \times m}$ and $\mathbf{Q} \in \mathbb{R}_+^{t \times m}$ such that:

$$\mathbf{M} \approx \mathbf{WQ}^T \tag{4}$$

WC-NMTF factorizes both $\mathbf{X}$ and $\mathbf{M}$. Note that the same $\mathbf{W}$ matrix is used in equation 3 and 4.

In our case $\mathbf{X}$ is the document-word matrix. $\mathbf{M}$ is a word-word matrix, also called the context matrix. It is made from a co-occurrence matrix which tells the number of times each word occur in the same context (i.e. document) than an other word. Then they use the Pointwise Mutual Information (PMI) on the co-occurrence matrix to make $\mathbf{M}$ (they actually use a variant of PMI, the Sparse Shifted Positive PMI or SPPMI).

The aim of the method is to minimize the difference between $\mathbf{X}$ and $\mathbf{ZSW}^T$ but also between $\mathbf{M}$ and $\mathbf{WQ}^T$, so we have to minimize the loss function:

$$F = \frac{1}{2}\|\mathbf{X} - \mathbf{ZSW}^T\|^2 + \frac{\lambda}{2}\|\mathbf{M} - \mathbf{WQ}^T\|^2 \tag{5}$$

λ is the regularization parameter. When $\lambda = 0$ in equation 5 we get back to the original NMTF.

After rewrite and derive equation 5 they arrived at four update rules, one for each matrix ($\mathbf{Z}$, $\mathbf{W}$, $\mathbf{S}$, $\mathbf{Q}$). Then we just need to iterate and apply these update rules many times to minimize the loss function.

3.4 Topic Modeling

The Latent Dirichlet Allocation (LDA), created in 2003 by David Blei , Andrew Ng and Michael Jordan, is a widely used statistical method for topic modeling. It assumes that similar topics have similar semantic field, therefore expressed using similar words. Thus, LDA groups words into a given amount of topics (the number of topics for the algorithm to look for has to be chosen before running it).

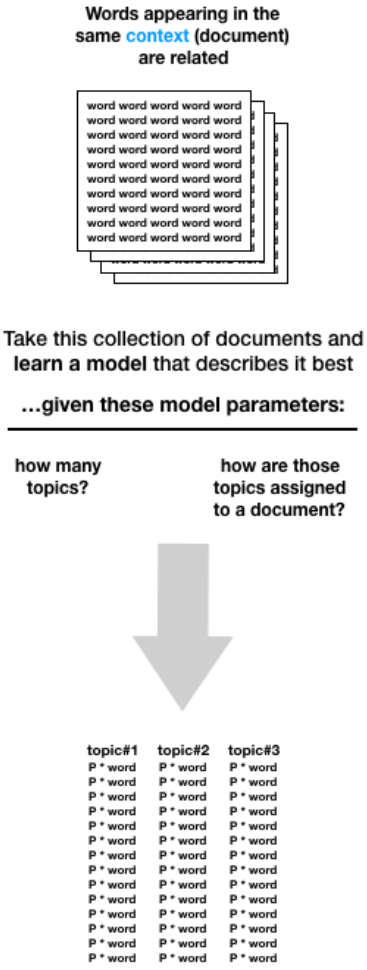

Figure 2: How LDA works.

The classification problem is to find the best predictor that discriminates all observations into groups, following some rules: Maximum likelihood – maximizes density within groups, Bayes Discriminant Rule – maximizes the probability of the classification into a group, and Fisher's linear discriminant rule – maximizes distance between groups and minimize variation within groups.

As a result, the method renders one $term \times topic$ and one $document \times topic$ matrices, containing the probability

of each word and document being related to each topic. In this work, the *Sickit Learn* Python library has been used to implement the LDA.

For example, in a series of papers on environmental regulations, the following topic modeling could be achieved:

- topic Animals: dog, chicken, cat, nature, zoo;
- topic Cooking: oven, food, restaurant, plates, taste, delicious;
- topic Politics: Republican, Democrat, Congress, ineffective, divisive

4 Experiments and Results

At first, we collected about 500000 tweets related to "abortion". This number becomes 86698 because of removing empty and duplicated tweets.

We used the Python library *NLTK* to do all the preprocessings operations presented above (3.1) . After these steps we finally had 10187 different words over all documents for 118204 words before, so we kept 8,61% of all the words.

4.1 Sentiment Analysis

4.1.1 Clustering Evaluation

Clustering can be quite complicated to evaluate when labels are unavailable. However, to evaluate our clustering, some good metrics can be used like the Silhouette Coefficient (SC), the Calinski-Harabasz Index (CHI) or the Davies-Bouldin Index (DBI).

SC uses a distance metric to measure how much an object is similar to its own cluster and how much is different with the nearest one. As a distance metric we can use for example the Euclidean distance or the cosine distance. The higher the value, the better.

CHI is a ratio of the dispersion between each cluster (that we want to maximize) and the dispersion inside each cluster (that we want to minimize). Therefore, the higher the value the better it is.

DBI is a measure of how much on average each cluster is similar to its most nearest cluster. A lower value indicates a better clustering. Indeed if each cluster is not similar to its nearest cluster it is that the clusters are pertinent due to their non-similarity, they are not redundant.

The clustering algorithms was used in order to make 2 clusters, one for positive tweets and one for negative ones. However, once we have the 2 clusters we can't directly find which one correspond to the positive and which one to the negative.

To handle this problem we used the lexicon-based results to compare the most frequent words of its positive group with the most frequent words of each cluster found. The cluster having the higher similarity with the positive group will be considered as the positive cluster. To compare these sets of words we used our Doc2Vec model that allows us to compare the similarity between two set of words.

As the lexicon-based result can't be took as an absolute reference, another way is to take among all words in the vocabulary the ones having the higher positive sentiment weighted by the number of times they appears across all documents. Then we compare this set of words with a set made by the same way with each cluster.

Both methods (the one using the lexicon-based results, and the one with the most positive words of the vocabulary) give the same results.

4.1.2 Lexicon-based Approach

With the SentiWordsNet and Emoji Sentiment Ranking dictionaries we found 63.44% words of our dataset (6463 found for a total of 10187 words).

Method	Positive	Negative
Lexicon-based	60025	26669

Table 1: Sentiment analysis results with lexicon-based approach.

We see that our data seems unbalanced according to the lexicon-based approach. We have twice more positive tweets than negative ones.

Since the lexicon-based approach work directly on the text we can't compute the SC, CHI and DBI metrics because the original dataframe is needed to compute them (dataframe that we don't have when we work directly on the texts). The different texts can't be considered as a dataframe.

However it exists some metrics to compare two distributions like the Normalized Mutual Information (NMI) or the Adjusted Rand Index (ARI). We will use these metrics to evaluate the lexicon-based clustering comparing to the learning-based one.

4.1.3 TF-IDF Representation

Due to the large size of the dataset (86694×10187) we tried to do Principal Component Analysis (PCA) in order to reduce the dimensions for each document but the variance explained was extremely low, almost 10% for the 50 first components, 50% of the variance was explained by the first 769 components. Due to this result we kept the original dataset.

Algorithm	Positive	Negative
K-Means	74464	12230
Spherical K-Means	79752	6942
WC-NMTF	84925	1769

Table 2: Sentiment analysis results with TF-IDF.

Algorithm	SC	CHI	DBI
K-Means	**0.003**	**232.1**	13.4
Spherical K-Means	0.001	169.97	**12.19**
WC-NMTF	0.001	7.69	29.99

Table 3: Clustering evaluation with TF-IDF.

With this representation the algorithms tend to classify very few negative tweets compared to the lexicon-based. Almost all tweet seems positive according to this configuration. These results will be more discussed in 4.1.6.

According to these metrics K-Means seems to have the better clusters. The very low CHI value for WC-NMTF indicates that either it have a high within-cluster dispersion (the points inside each cluster are distant) or a low between-cluster dispersion (the clusters are too close) or both. Since its DBI metric is also high (compared to the others) we can said that the WC-NMTF clusters have at least a lower between-cluster dispersion than the other algorithms.

4.1.4 Doc2Vec Representation

We trained the Doc2Vec model using the *Gensim* library over 2000 epochs in order to encode each document with a 50 dimensions vector. The default value of 5 for the window size was kept.

Since the WC-NMTF method uses a co-occurence matrix between words we can't use it as it is with the Doc2Vec representation. Indeed, each dimension of the vector representing a document is no longer associated with a particular word. Thus, we used the original NMTF (i.e. with $\lambda = 0$ in equation 5).

Algorithm	Positive	Negative
K-Means	43768	42926
Spherical K-Means	46834	39860
NMTF	58307	28387

Table 4: Sentiment analysis results with Doc2Vec.

Algorithm	SC	CHI	DBI
K-Means	0.035	1364	7.661
Spherical K-Means	**0.036**	**1392**	**7.589**
NMTF	0.017	640.6	10.51

Table 5: Clustering evaluation with Doc2Vec.

K-Means and Spherical K-Means have clusters with more or less the same size.

NMTF seems to be outperformed by K-Means and Spherical K-Means with the Doc2Vec representation of documents. However Spherical K-Means is slightly better than K-Means and have the best clusters according to the three metrics used.

4.1.5 Density Matrix Representation

We've used a variant of the GQLM optimization algorithm, instead of only use the term frequency for the computation we used the TF-IDF score normalized with L2 since this score seems to be better than the simple term frequency score.

The density matrix is a diagonal matrix when we use the one-hot encoding in the computation. Only this diagonal can be kept to represent a document instead of the whole matrix to facilitate storage and computation.

Moreover, the one-hot encoding of a word is a very large, but sparse, vector of size t (the number of different words across all documents). The diagonal of the density matrix for each document is the same size, but it is non sparse (because of the optimization process). The density matrices for all documents is therefore a $n \times t$ matrix but non-sparse (with n the number of documents).

Due to this non-sparsity we have a huge non-sparse matrix (86694×10187) that is intractable by our computers.

Therefore we used Word2Vec to encode each word in a vector of 50 dimensions and then use this vector instead of the one-hot encoding. This allow us to have

a better representation of the words and also to reduce the size of the density matrix of each document in order to be more computationally tractable. The density matrix is no longer a diagonal matrix when we work with the Word2Vec representation. This resulting in a 86694×2500 matrix, 2500 because we flatten the matrix to have a $50 \times 50 = 2500$ dimensions vector.

We must notice that there is an interesting trade-off here. On the one hand, the one-hot encoding is faster for the density matrices computation (due to its sparsity) but the resulting matrices takes more in storage (due to the non-sparsity of the density matrices and its large size). On the other hand, the Word2Vec encoding is slower for the computation (the encoding is non-sparse) but takes less in storage (because we have less dimensions to encode each density matrix).

To manage the sparse matrices we used the Python library *Scipy*.

Algorithm	Positive	Negative
K-Means	63133	23561
Spherical K-Means	47206	39488
NMTF	44171	42523

Table 6: Sentiment analysis results with Density Matrix.

Algorithm	SC	CHI	DBI
K-Means	0.012	**1475**	**6.795**
Spherical K-Means	**0.025**	1313	8.025
NMTF	0.002	72.05	34.38

Table 7: Clustering evaluation with Density Matrix.

Once again, according to these metrics, NMTF is outperformed by K-Means and Spherical K-Means, but more blatantly than with the Doc2Vec representation. Like with TF-IDF, the low CHI value and the high DBI one indicate a high similarity between the clusters found by NMTF.

But with the Density Matrix representation this is the K-Means algorithm that is slightly better than Spherical K-Means.

4.1.6 Comparison of Representations

The NMI and ARI between the different learning-based clusters and the lexicon-based clusters are near 0. This means that they seem not to share information and not to represent the same thing. As we don't have the ground truth labels we can't assure which approach seems to be better.

However, the similarity (using our Doc2Vec model) of the most frequent words of each positive group (respectively with negative group) is near 1.

Representation	SC	CHI	DBI
TF-IDF	0.003	169.97	12.19
Doc2Vec	**0.036**	1392	7.589
Density Matrix	0.025	**1475**	**6.795**

Table 8: Best metrics values with each document representation.

Among the different documents representation it seems that the Density Matrix is the best in our context according to the CHI and DBI metrics (even if Doc2Vec seems to be slightly better with the SC metric).

The metrics for the TF-IDF representation are very weak comparing to the other representations, especially the CHI. According to this metric and the results, TF-IDF seems to have clusters too near to each other. It can explain why its clusters classify almost all tweets as positive. The positive cluster seems to encroach on the negative one. This representation appears to not be pertinent in our context. Indeed, it take words independently without taking account their context like Doc2Vec or the Density Matrix do.

The very low SC values for each representation show us that the average distance between a point of a cluster with the other points of this cluster is almost equal to the average distance of this point with the points of the nearest cluster. That means that the clusters seems to be very close to each other.

4.2 Topic Modeling

Using the LDA method discussed above, a few tests were taken in order to find the optimal number of topics. After the preliminary analysis, it has been decided to keep only three main clusters. For this analysis, Gensin python library has been used. The two figures below plot the three clusters found through LDA, how they relate to each other considering the two main principal components and the top 30 words in each cluster.

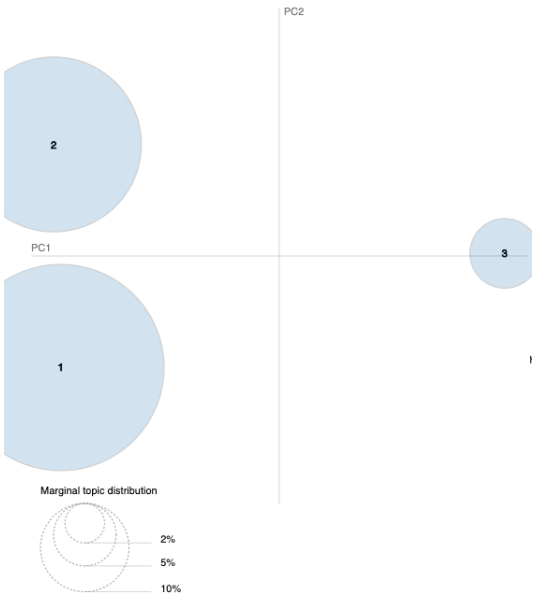

Figure 3: Intertopic Distance Map (via multidimensional scaling).

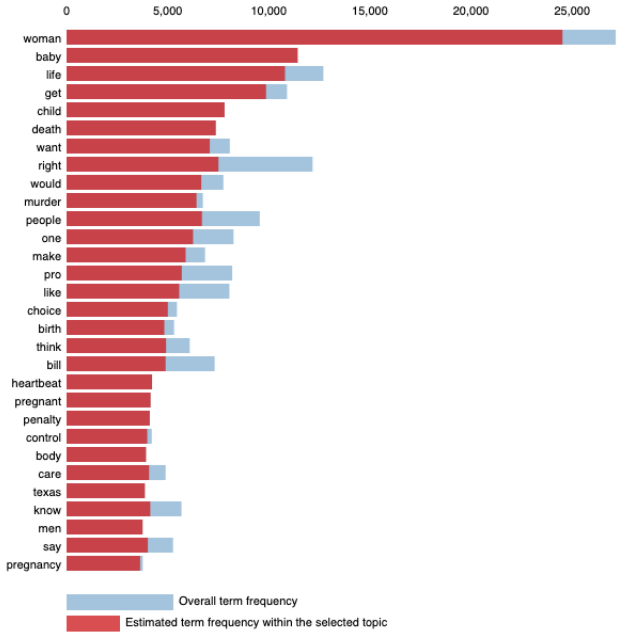

Figure 4: Top-30 Most Relevant Terms for Topic 1 (54.6% of tokens)

To name each cluster is a supervised and rather subjective task. While Topic 1 retains the greatest correlation to the words *baby, child, body* and *pregnancy*, Topic 2 retains it to the words *law, state, ban, country, christian* and *right*. It seems that Topic 1 is more related to individual rights, ethics and beliefs, while Topic 2 words can be summarized as political issues, legislation.

The following figures show the top-30 relevant words for each topic.

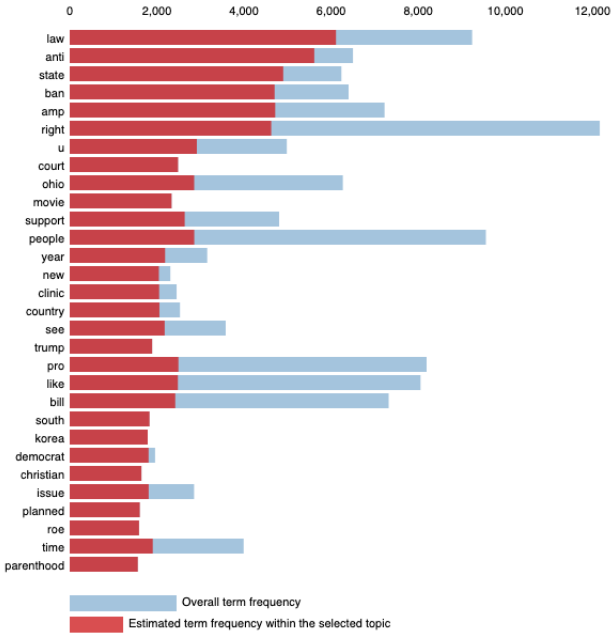

Figure 5: Top-30 Most Relevant Terms for Topic 2 (39.2% of tokens)

As described in the graphs, Topic 3 has not been considered relevant to the overall analysis. The words in the cluster are mostly nonsense. Despite preprocessing the dataset, it seems the algorithm made a cluster with names, mentions, slang and abbreviations. Moreover, this topic is not very representative of the dataset.

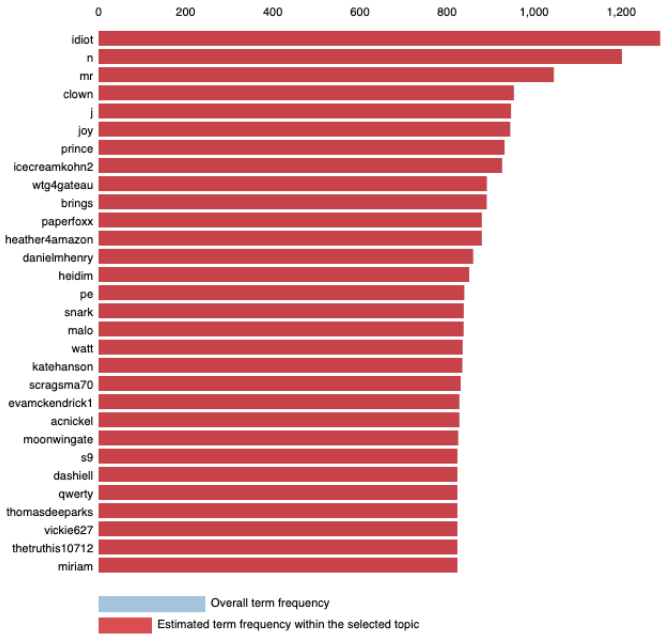

Figure 6: Top-30 Most Relevant Terms for Topic 3 (6.2% of tokens)

Thus, in order to keep it simple and get useful results, only two clusters have been chosen:

- **Topic 1**: Individual rights
- **Topic 2**: Legislation and politics

Having inferred the names of each topic, it becomes easy to analyze the document distribution among them. As discovered from the keywords, the third topic is not very relevant and will not be considered in the final results.

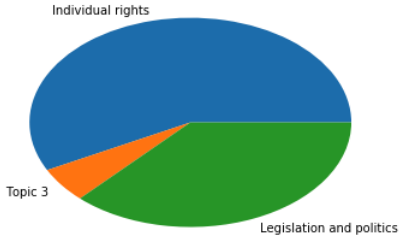

Figure 7: Document distribution among topics.

	Individual rights	Legislation and politics
Number of tweets	49877	32107

Table 9: Number of tweets for each topic.

For each method and each topic we compute the Normalized Difference between the number of Positive and Negative Documents score (NDPND). This score is the difference between the number of positive tweets and negative ones for a particular topic divided by the number of tweets in this topic. Indeed, we can say that is the probability to have a positive tweet in this topic minus the probability to have a negative one.

It ranges from -1 (only negative tweets) to 1 (only positive). Hence a positive score means that there is more positive tweets than negative.

For each learning-based method we took the result of the unsupervised algorithm having the higher SC, CHI and DBI values.

Representation	Individual rights	Legislation and politics
Lexicon	0.38	0.41
TF-IDF	0.88	0.44
Doc2Vec	0.06	0.08
Density Matrix	0.58	0.20

Table 10: NDPND score according to each topic with each method.

We see that the TF-IDF representation have a very high NDPND score for the "Individual rights" topic, that corroborates with its clustering results because it tends to classify almost all tweets as positive.

Since Doc2Vec clusters have more or less the same number of tweets in its positive and negative clusters it

make sense to see a NDPND score near 0.

In table 10 we see that there are no negative values, which means that each method seems to find more positive tweets than negative ones according to each topic. However, this result can be biased if we really have more positive than negative tweets like the lexicon-based result tells us.

5 Conclusion and Future Works

5.1 Conclusion

We saw through this work a comparison of different vector representation of documents and unsupervised algorithms to analyze them. In our context the Density Matrix representation seems to be the best. Regarding unsupervised algorithms, K-Means and Spherical K-Means give the best results whereas WC-NMTF and NMTF wasn't able to make good cluster.

With our data, "abortion" seems to be seen as positive either at the level of individual rights or for the legislation aspect. However, our data can be biased and may not represent a good distribution.

5.2 Future Works

In order to have more precision about opinions towards abortion, we can collect more tweets using the Twitter API.

We can also run our sentiment analysis and topic discovery algorithms according to a geographical area. This will allow us to see what aspects of abortions are the most relevant in each region.

In the future, we can try to run our algorithms with removing emojis from the tweets. Thus, we will notice how emojis can be important in a text document for a sentiment analysis.

What can also be done, is to increase the dimension of the document representation in the *Doc2Vec* representation to see if we can obtain a better result.

Another way to improve our work, is to find a correct method to evaluate the lexicon-based approach and compare it to the learning-based one.

Trying other unsupervised algorithms could create more precised results on the sentiment analysis.

It would have been interesting to use the original Density Matrix (with the term frequency and the one-hot encoding) to see if the use of TF-IDF score and Word2Vec encoding in the computation really improved the metrics.

Finally, our github repository corresponding to this project is open source and free to use[1].

[1] https://github.com/lucarp/opinion-mining

References

Long et al., 2005 - Bo Long, Zhongfei (Mark) Zhang and Philip S. Yu. Co-clustering by Block Value Decomposition. 2005.

Ding et al., 2006 - Chris Ding, Tao Li, Wei Peng Haesun Park. Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering. 2006.

Baccianella et al., 2010 - Stefano Baccianella, Andra Esuli,and Fabrizio Sebastiani. SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. 2010.

Palpanas et al., 2011 - Mikalai Tsytsarau and Themis Palpanas. Survey on mining subjective data on the web. 2011.

Gonçalves et al., 2013 - D. S. Gonçalves, M. A. Gomes-Ruggiero and C. Lavor. Global convergence of diluted iterations in maximum-likelihood quantum tomography. 2013.

Hu et al., 2013 - Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. Unsupervised Sentiment Analysis with Emotional Signals. 2013.

Le et al., 2014 - Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. 2014.

Novak et al., 2015 - Petra Kralj Novak, Jasmina Smailovic, Borut Sluban, Igor Mozetic. Sentiment of Emojis. 2015.

Unnisa et al., 2016 - Muqtar Unnisa, Ayesha Ameen and Syed Raziuddin. Opinion Mining on Twitter Data using Unsupervised Learning Technique. 2016.

Joshi et al., 2018 - Shaunak Joshi and Deepali Deshpande. Twitter Sentiment Analysis System. 2018.

Karami et al., 2018 - Amir Karami, London S. Bennett and Xiaoyun He. Mining Public Opinion about Economic Issues: Twitter and the U.S. Presidential Election. 2018.

Salah et al., 2018 - Aghiles Salah, Melissa Ailem and Mohamed Nadif. Word Co-Occurrence Regularized Non-Negative Matrix Tri-Factorization for Text Data Co-Clustering. 2018.

Zhang et al., 2018 - Yazhou Zhang, Dawei Song, Xiang Li and Peng Zhang. Unsupervised Sentiment Analysis of Twitter Posts Using Density Matrix Representation. 2018.