Computing and Informatics, Vol. 41, 2022, 56–77, doi: 10.31577/cai_2022_1_56

TOPIC EXTRACTION IN SOCIAL NETWORKS

Chaima Messaoudi*

Mars lab
Higher Institute of Computer Science and Telecom (ISITCom)
University of Sousse, Tunisia
&
CReSTIC, EA 3804
University of Reims Champagne-Ardenne, France
e-mail: chaima.messaoudi@etudiant.univ-reims.fr

Zahia Guessoum

CReSTIC, EA 3804
University of Reims Champagne-Ardenne, France
&
LIP6, UMR 7606
Sorbonne University, France

Lotfi BenRomdhane

Mars lab
Higher Institute of Computer Science and Telecom (ISITCom)
University of Sousse, Tunisia

Abstract. The number of Twitter users is increasing and the quantity of produced data is growing. Using this big data to analyze user behavior has become a very active field. The two key challenges of this paper are extracting data from Twitter and extracting topics from user tweets. The proposed approach uses data crawling to collect data from Twitter and a bunch of natural language processing techniques

* Corresponding author

to extract information from the so collected data and build a dataset. Thereafter, we use K-means clustering and Latent Dirichlet Allocation to extract the prevalent topics from this dataset, as they are the most common in the literature. Our proposal is generic, it can be reused by scientists to annotate any text collection.

Keywords: Twitter, topic extraction, big data, data crawling, social networks

1 INTRODUCTION

Nowadays, one can notice the booming rise of microblogging sites fad among many individuals to communicate their ideas using short texts on various topics. Those microblogging sites are gaining more popularity among individuals, celebrities, and even politicians to widely distribute short messages to their followers. The most widely recognized microblogging site is Twitter. The latter enables users to post a message within 280 characters.

Topic extraction is beneficial for identifying the network's trending and dominant topics, as well as covering more user goals. For instance, the government or companies may track people's satisfaction and adopt their methodology in real-time. This paper aims to find the most common topics in a collection of tweets and then create a dataset based on those topics and tweets. For this, we present in this paper a step-by-step methodology to extract topics from general twitter data. In order to extract the topics within these tweets, we use and compare two well-known techniques; namely, Latent Dirichlet Allocation (LDA) and a variant of K-means clustering. These are coupled with other standard techniques, as Word2Vec, in order to define a workflow suitable to the purpose. A key feature of our approach is that it is generic, it can be easily adopted by scientists and practitioners to annotate other datasets.

The paper is organized as follows: First, we review the existing literature to determine the most effective and efficient topic extraction approaches. Second, we present our method to extract data from Twitter starting from an existing user network. Third, we provide an empirical evaluation of two different approaches for the topic extraction task on our dataset. We include results and discussion in this section. Finally, we end up this paper with a conclusion and some perspectives.

2 RELATED WORK

With the prominence of the available online text data, many researchers focus on text mining techniques in order to find the relationship between data and text documents. Among those techniques, we are interested in topic extraction also known as topic modeling. Topics can be used to analyze the interaction of the different users of a social network.

There have been several approaches in the area of topic extraction (see [1, 2, 3, 4, 5]). Those approaches have been used in a variety of areas, including political science, medical science, and linguistic science, among others [6].

To reduce a high-dimensional dataset, Paulraj and Neelamegam [1] use principal component analysis (PCA), which is analogous to using the singular value decomposition (SVD) technique on the covariance matrix. They also suggested a new initial centroid approach that calculates the variance of data in each dimension, defining the column with the highest variance, sorting it in any order, partitioning the data into k subsets, and then deciding the median of each subset to be the cluster's initial centroid. Both distances between the data and the centroid in each cluster are used to determine an inertia value. The goal is to find initial centroids with the smallest inertia value, which is the global optimum. Although this approach is interesting, one of the main issues is that the experiments are limited to a sampled dataset and haven't been tested on a real-world dataset.

The work presented in [7] used Term Frequency-Inverse Document Frequency (TF-IDF) to extract features, and the SVD was used to reduce the high-dimensional dataset while still retaining the most relevant features. In fact, in microblogging, messages are short and noisy which causes high sparseness and produces high-dimensional datasets. In order to compute the initial centroids in K-means, Artificial Bee Colony (ABC) algorithm was used. The sentiment polarity was then calculated using K-means and SentiWordnet. For the experiments, they extracted tweets including "iPhone X". They applied K-means to generate clusters which were then scored by SentiWordNet for class labeling. This approach demonstrates that combining different techniques can significantly boost K-means performance. However, there is still a need for improving the Word Sense Disambiguation (WSD) technique to be suitable to the short messages and to consider other data than hashtags in tweets.

The model proposed in [3] aims to develop Twitter sentiment analysis by using a topic-based mixture extraction method and semi-supervised training. They initially built a state-of-the-art baseline for a rich feature set then a topic-based sentiment mixture model was built having the topic-specified data arranged in a semi-supervised training structure. The information regarding the topic is generated with the help of topic extraction which is based on an application of Latent Dirichlet Allocation (LDA). Experiments on the SemEval test set revealed that the topic-based sentiment mixture model achieved an F score of 71.7 % on the tweet-topic with semi-supervised data.

The works [5, 4] analyze tweets using clustering techniques. The approach of [5] aims to cluster the tweets based on core topics and re-tweets while the approach of [4] uses a K-means-based clustering algorithm. A major drawback of those two approaches is that they solve the topical clustering by using hash-tags [4] or by searching specific words [5] and they do not give a comprehensive vision of the dominant topics in a set of tweets.

Although much progress has been made in the field of topic extraction, some major difficulties remain, such as existing datasets' ignoring of the user's network structure. This shortcoming makes the dataset disadvantageous for further future applications like influence maximization. In our work, we extract data from Twitter. Then, we compare the results of some popular topic extraction algorithms. We adopted those algorithms to improve our results.

3 TOPIC EXTRACTION FROM TWEETS

To explore and extract topics from the collected data, we focus on the two most popular and well-cited algorithms for topic extraction in the literature: Latent Dirichlet Allocation [2] and K-means [8]. In this section, we give a brief overview of those algorithms.

3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a well-known generative model used for topic extraction [2]. It treats documents as probabilistic distribution sets of words or topics. These topics are not strongly defined – as they are identified on the basis of the likelihood of co-occurrences of words contained in them. More precisely, it is a three-level hierarchical Bayesian model that takes as input a collection of text documents and extracts clusters of words such as each cluster defines a topic. Noteworthy a word may belong to several clusters. To each document, this model also attributes the distribution of topics.

Starting from a collection of M documents and each of those documents has N number of words, we initialize two concentration parameters α and β, and k topics. We run an iterative process to obtain k clusters of words where each cluster refers to a topic, the frequency ψ of words in each topic, and the distribution ϕ of topics per document.

The iterative part of LDA can be described by Algorithm 1.

In line 7 of Algorithm 1, for each word w of each document d, we calculate two values for each topic z:

- $p(z|d)$: the probability that the document d will be assigned to the topic z,
- $p(w|z)$: the probability that the topic z in the corpus is assigned to the word w.

We then choose the new topic z with the probability $p(z|d){*}p(w|d)$. This corresponds to the probability that the topic z generates the word w in the document d.

3.2 K-Means

K-means is a simple and powerful clustering algorithm that deploys K centroids. The parameter k is an input of the algorithm and specifies the number of clusters. K centroids are defined in the first place and the principle is that each of those centroids regroups the nearest data based on a chosen distance function. The Euclidean distance is widely used to calculate the distance between those centroids

Algorithm 1: Latent Dirichlet Allocation Algorithm

```
Data: $D = \{d_1, \dots, d_M\}$, $\alpha$, $\beta$, $T$
Result: Topic assignments $z$, $\psi$ and $\phi$
Randomly initialize $z$;
while $iterations < \max_{iterations}$ do
    foreach $w \in D$ do
        Resample topic $z$ for $w$ given words and their current topic
          assignments;
        foreach $topic \in 0, \dots, k-1$ do
            Compute $p(z|d) * p(w|d)$;
        end
    end
    Compute $z$ ;
    Compute $\phi$ ;
    Compute $\psi$ ;
    Get results ;
    Evaluate model ;
end
```

and each data. K-means is an iterative process and it stops when classes become stable; i.e., centroids do not change. K-means algorithm has many advantages such as simple mathematical formulation, fast convergence, and easy implementation [9]. Algorithm 2 represents the general principle of K-means.

Algorithm 2: K-Means Algorithm

```
Data: $D = \{d_1, \dots, d_M\}$, $k$
Result: $k_c lusters$
Randomly initialize $k$ centroids;
while The centroids change do
    Assign each document to the cluster which has the closest centroid;
    Calculate new centroid for each cluster;
end
```

4 EXPERIMENTS

In this section, we first present the considered method for data acquisition. The latter is a key step in any data mining process. Thereafter, we present and analyze the experimental results of the considered algorithms for topic extraction (LDA and K-means).

4.1 Data Acquisition

We aim to collect several data related to the tweets and the relative user's profile from Twitter while preserving a connected network's structure. We started with the network's file crawled in [10]. We extracted a network of one million users from this network. We consider the ID of the tweet, the ID of the user, the name of the user, the content of the tweet, the language of the tweet, the number of likes, the number of replies, the number of retweets, the retweet origin, the time, the following, the followers, the link, the source of video, the cover of video, and the images.

We crawled the data related to tweets and users' profiles using Beautiful Soup: Selenium and Web Driver [11]. To achieve the purpose of web scraping using Python, we may only need to use the Beautiful Soup. It is a strong library that makes online scraping easier to develop by traversing the Document Object Model (DOM). However, it simply scrapes in a static manner. JavaScript is ignored by static scraping. It does not require the use of a browser to retrieve web pages from the server. "View page source" gives exactly what we see. Dynamic scraping, on the other hand, comes to the rescue if you need data from components that are rendered when JavaScript links are clicked. Dynamic scraping will be accomplished by combining Beautiful Soup and Selenium. Selenium is a Python library that automates web browser interaction. As a result, the data rendered by JavaScript links may be made available by using Selenium to automate button clicks, which can subsequently be retrieved using Beautiful Soup [11].

The search of users' profiles has been assured by the ID or the name of the user and only the existing accounts with following existing links are collected. To begin, we used Selenium to automate the search clicks and scroll of web pages in order to gather the necessary data. To put it another way, the primary goal of this technique is to automate the data collection process through the browser. The experiments are conducted using Python on a machine (Intel Core i5 Quad CPU 3.2 GHz with 15.0 GB of memory). The collected data is described in Table 1.

Total number of tweets	11 695 015
Number of tweets with text	11 695 015
Number of tweets with text and images	1 143 190
Number of users	1 245 474
Number of edges	12 388 966
Number of images	1 335 898

Table 1. Properties of the collected data

Having this dataset at hand, we have proceeded thereafter to the evaluation of unsupervised machine learning models for topic extraction (LDA and K-means) on this data, as detailed subsequently.

4.2 Latent Dirichlet Allocation Experiment

In this section, we present the experimental results of the LDA algorithm implemented in [12] on our crawled dataset. One of the most used measures to evaluate the topic extraction algorithms is the coherence measure c_v [13]. Based on the experimental results, the best-performing coherence measure is a new combination found by a systematic study of the configuration space of coherence measures. It combines the indirect cosine measure with the normalized pointwise mutual information (NPMI) and the Boolean sliding window [13].

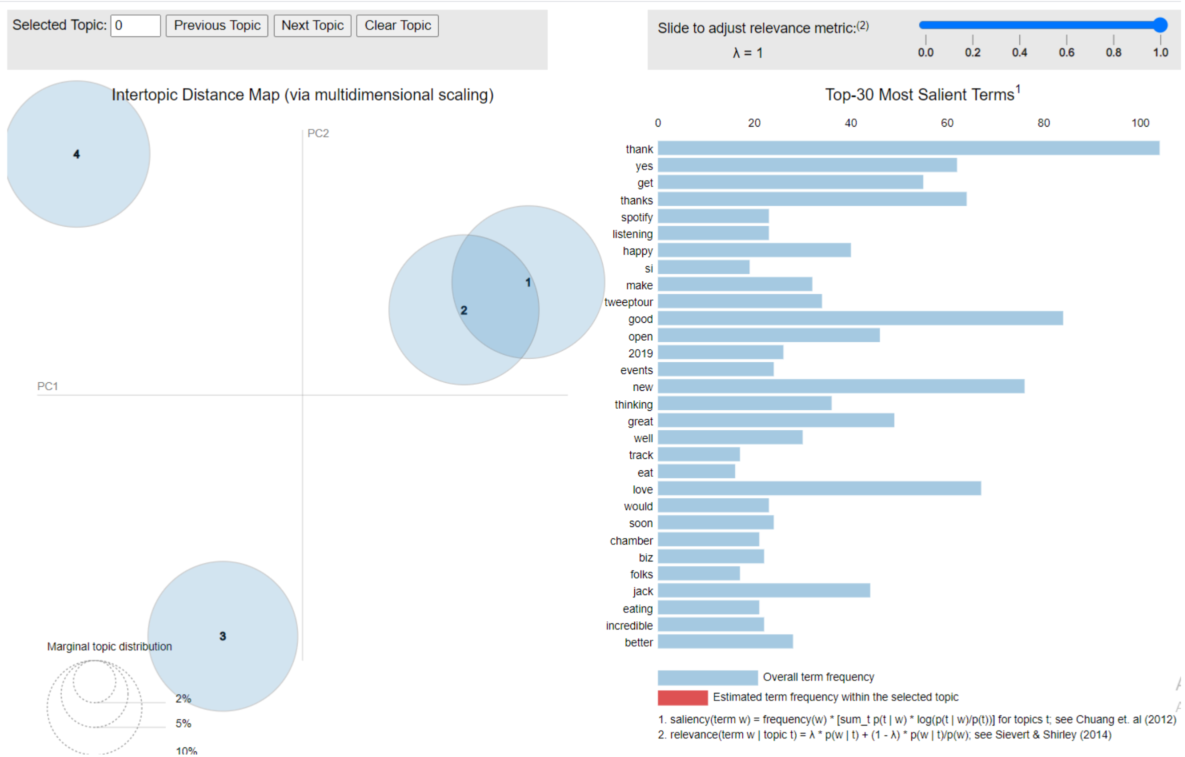
We used a wrapper to add the new measure to the existing code. We also used a mallet called LDA mallet to modify the implementation. The results are visualized by the LDAvis, a web-based interactive visualization of topics estimated using Latent Dirichlet Allocation that is built using a combination of R language and Data-Driven Documents (D3) [14]. We obtained $C_v = 0.42$ for a number of topics equal to 4 which is the best result in our experiment. We opted to a symmetric α and the number of passes = 3. After iterating, the system obtained $\beta = 0.25$ and $\alpha = 0.25$.

In this experiment, the preprocessing includes the steps of removing contractions, whitespaces, punctuation, URL, numbers, detecting abbreviations, emoticons, mouth repetitions, hashtags, mentions and stop words. Then we tokenized the tweets assuming that the text has no HTML escaping. After the preprocessing, we build a term dictionary of our corpus, where every unique term is assigned an index. The prepared dictionary is used to build a corpus by converting the list of documents (corpus) into a document term matrix. The next step is to create an object for the LDA model and train it on a document term matrix. Furthermore, the gensim module in Python is employed to run the LDA model with some predefined number of topics. The LDA model is then visualized using gensim and pyLDAvis library. PyLDAvis is a Python library for interactive topics model visualization which helps users interpret the topics in a topic model [15]. PyLDAvis provides two visualization panels.

Figure 1 presents the results of the whole clustering process of LDA. In this figure, we have the list of the most frequent words on the right, and the set of the extracted topics on the left.

The results of LDA for topic extraction are reported in Figures 2, 3, 4 and 5. In each of these figures, the right panel represents the 30 most relevant terms with the percentage of them among the total number of tokens to each topic appearing on the left panel. The inter-topic distance is also visualized. For each term, the overall frequency is represented.

If tweets tend to cluster along topic lines, each LDA topic should ideally correspond to a specific topic. This is not the case in practice, as we see above. Each topic includes a variety of related words that are semantically distant based on our observations. Those results have further strengthened our need for applying another method for topic extraction to get better results.



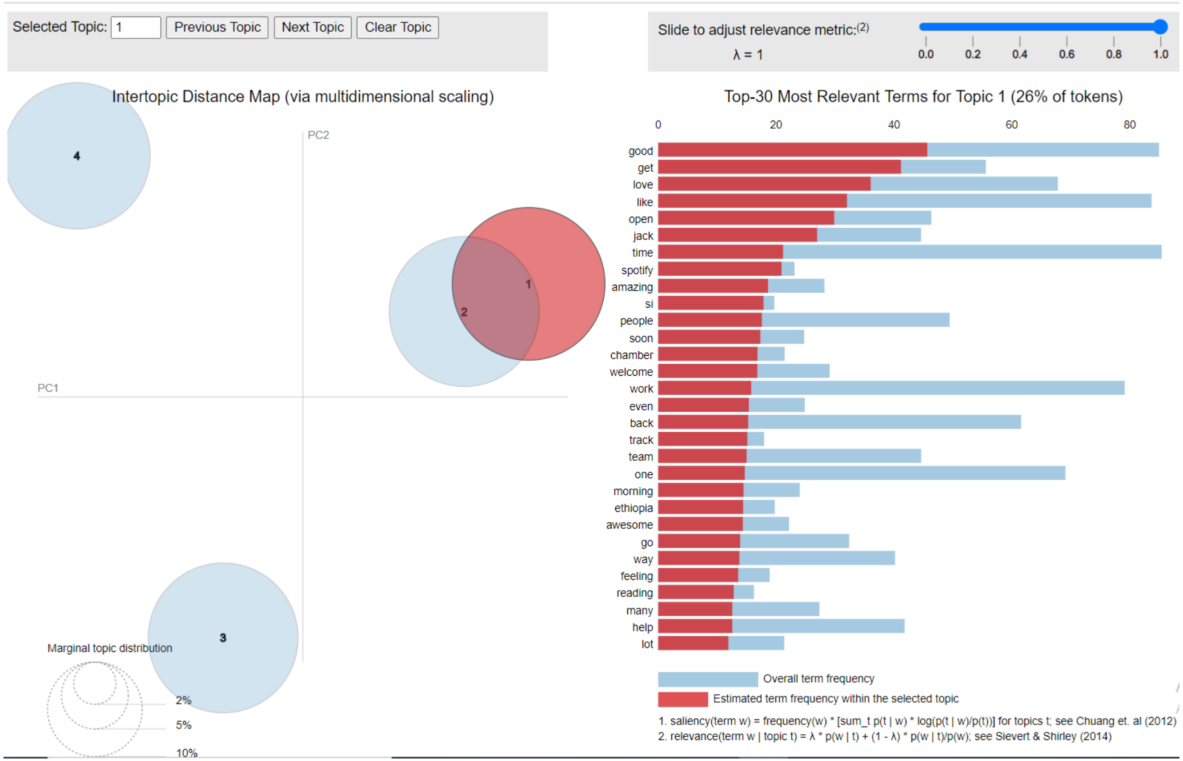
Selected Topic: 0

$\lambda = 1$

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)

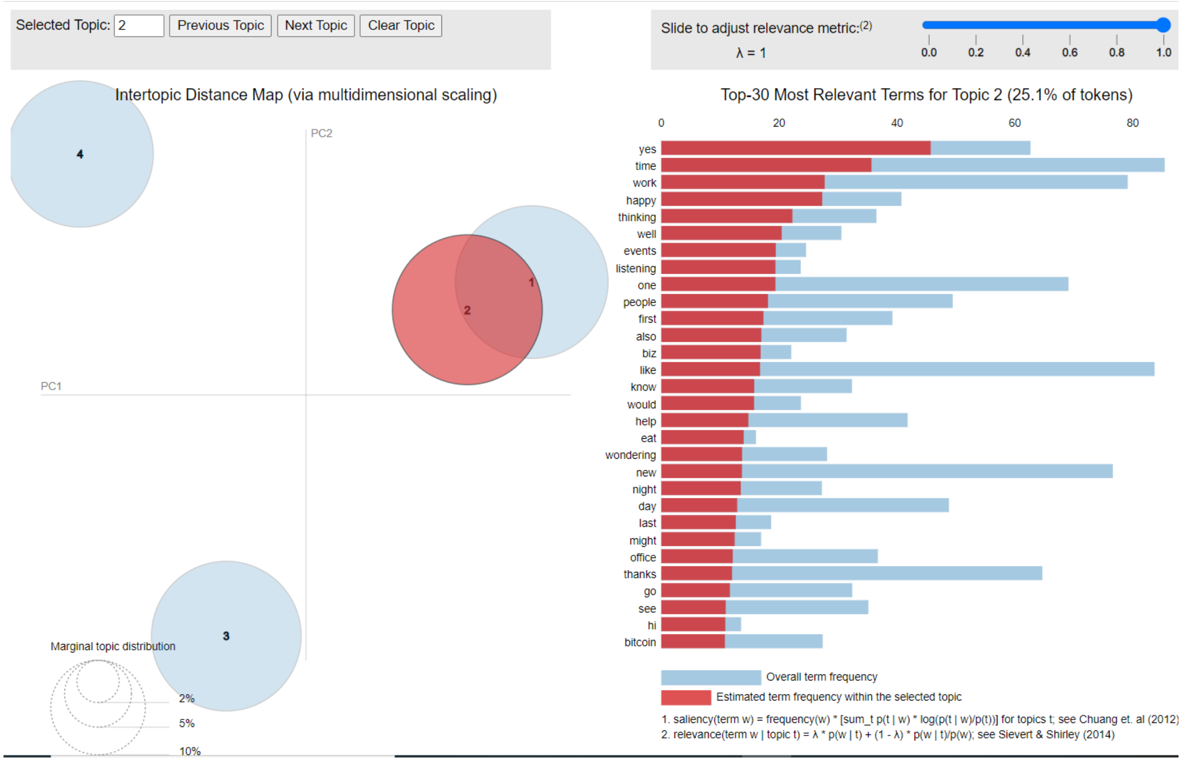
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 1. Selecting *topic4* (on the left) reveals the most used terms (on the right) for interpreting the selected topic



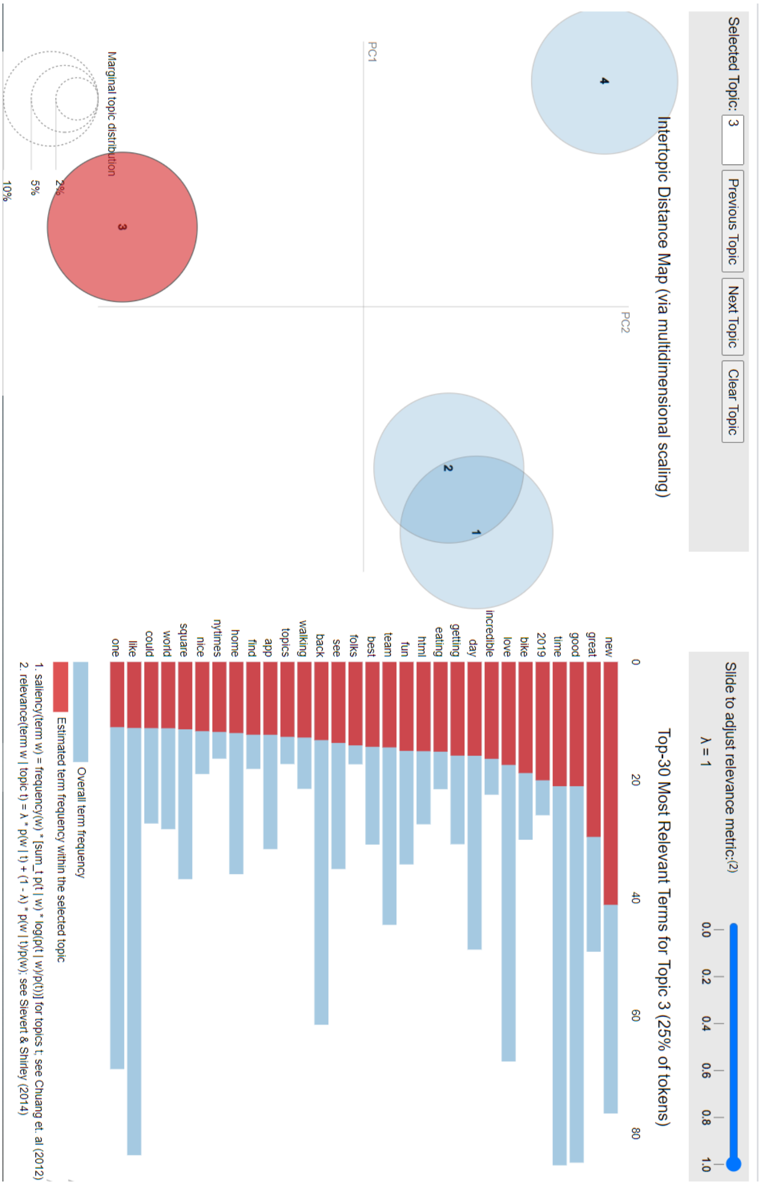
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 2. The layout of LDAvis, with top 30 most salient terms represented by bar charts on the right and inter-topics distance map on the left



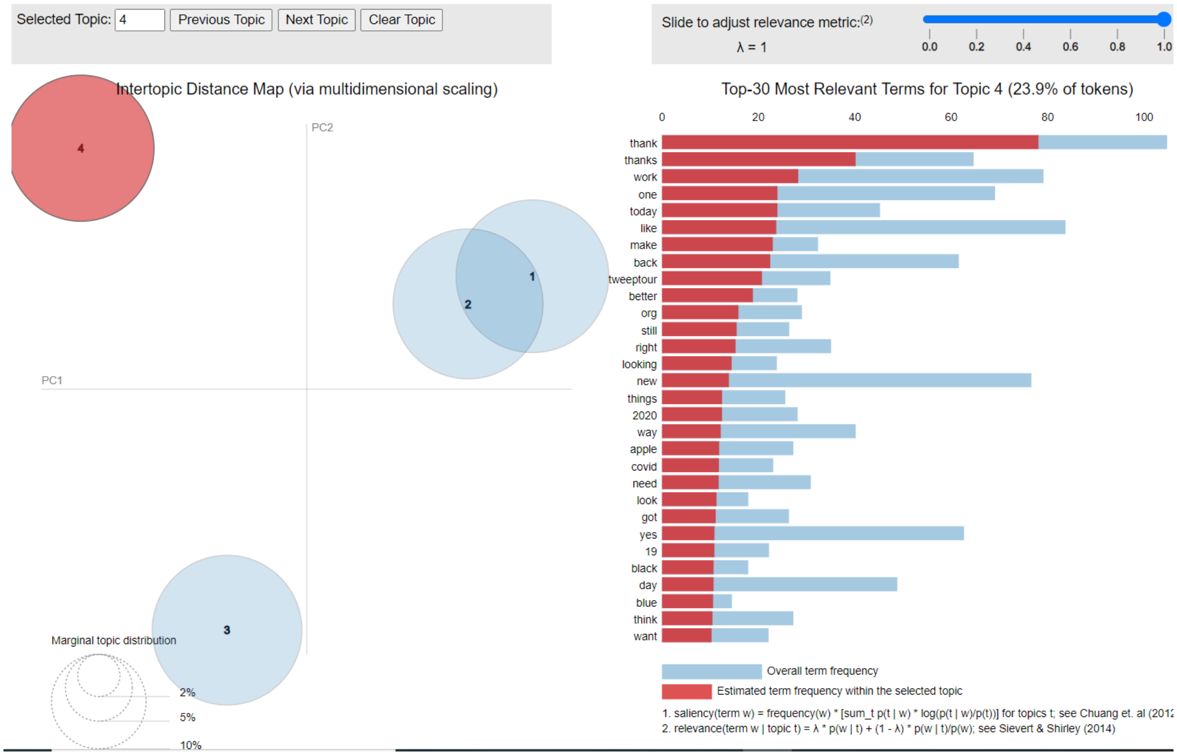
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 3. Selecting *topic1* (on the left) reveals the most used terms (on the right) for interpreting the selected topic



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 4. Selecting *topic2* (on the left) reveals the most used terms (on the right) for interpreting the selected topic



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012

2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 5. Selecting *topic3* (on the left) reveals the most used terms (on the right) for interpreting the selected topic

4.3 K-Means

In this section, we focus on the K-means algorithm. A major concern to evaluate the experimental results is the lack of a ground truth. For this, first we have proceeded to the manual annotation of sample tweets, as described in the next section.

We conducted different experiments based on the variation of the original K-means algorithm on our dataset for better topic extraction.

4.3.1 Manual Annotation

It is quite obvious that manual annotation of the tweets is a tedious task and almost impossible to achieve on a big data. For this, we have proceeded to the manual annotation on a small subset of tweets of randomly sampled 2 219 tweets belonging to 400 users. The minimal number of tweets per user is 10; while the maximal number is 50. Worth mentioning that a tweet can belong to more than one topic. Table 2 presents the statistical properties of this manual annotation.

lol (funny)	368
functionality (Functionality)	329
development (Development)	233
asparagus (Food)	144
curtis (Person)	753
vat (Fee)	149
riverside (Place)	271
mariner (Sport)	241
merely (Morality)	245
wool (Clothes)	98
insulting (Insulting)	115
government (Government)	154

Table 2. Statistical properties of the manual annotation

Having this annotated dataset at hand, now we are ready to outline the topic extraction process based on K-means in the next section.

4.3.2 Preprocessing and Topic Extraction

In this second series of experiments, we use K-means clustering coupled with Word2vec to produce word embedding. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of texts and produces a vector space, with each unique word in the corpus being assigned a corresponding vector in the space. Stated otherwise, words are "vectorized" in the vector space such as words that share common contexts in the corpus are located close to one another in the space.

Additionally, we use *gensim.utils.simple_preprocess*() to do some preprocessing on the tweets. The model's construction is straightforward. We create a Word2Vec object and sent the outputs we read in the previous step. As a result, we have a list of lists. Each list in the main list includes a collection of tokens. Both are used by Word2Vec to construct a vocabulary.

The previous step establishes the vocabulary and begins to train the Word2Vec model. The resulting embeds can be thought of as features that define the target term. Now we will feed word embeddings into the most common clustering algorithm K-means. Doing it this way, we will be able to cluster tweets into a set of topics w.r.t. their contents. However, a major issue in K-means clustering is setting the optimal number of clusters k which refers in our case to the number of topics. For this, we run the algorithm with several values of k ranging from 2 to 20.

In K-means, we use the cosine similarity to measure the distance between words. Thereafter, the optimal number of clusters is computed using two criteria. The first criterion is inertia defined as the sum of the squares of the distance between each point and the nearest centroid. We use the elbow method to find the optimal k. The second criterion is the standard average of silhouette coefficients. We also vary the minimum number of words min-count value from initially 5 to 100 and 500. The model would ignore words that do not satisfy the min-count. In fact, extremely rare words are usually unimportant and therefore are removed. Simulation results for both criteria are reported in Figures 6 and 7 for min-count = 100; and in Figures 8 and 9 for min-count = 500.

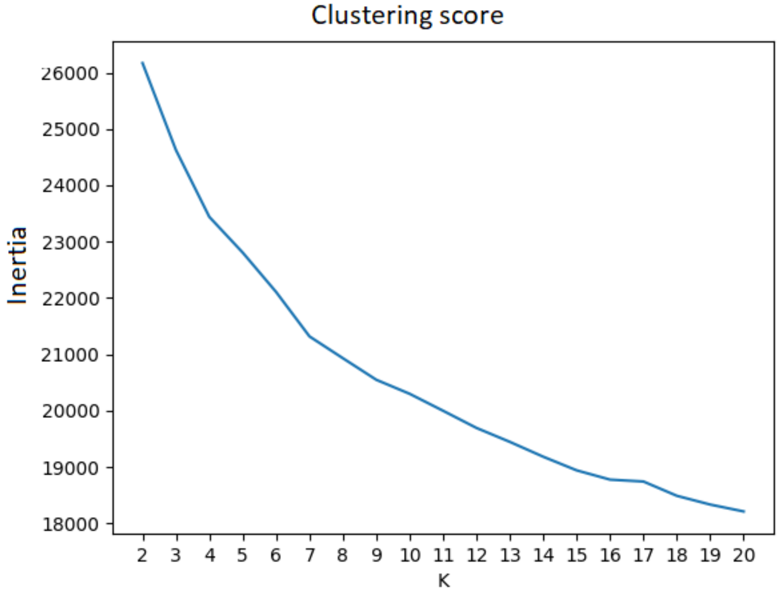


Figure 6. Inertia criteria for $k = 2$ to 20 with min-count = 100

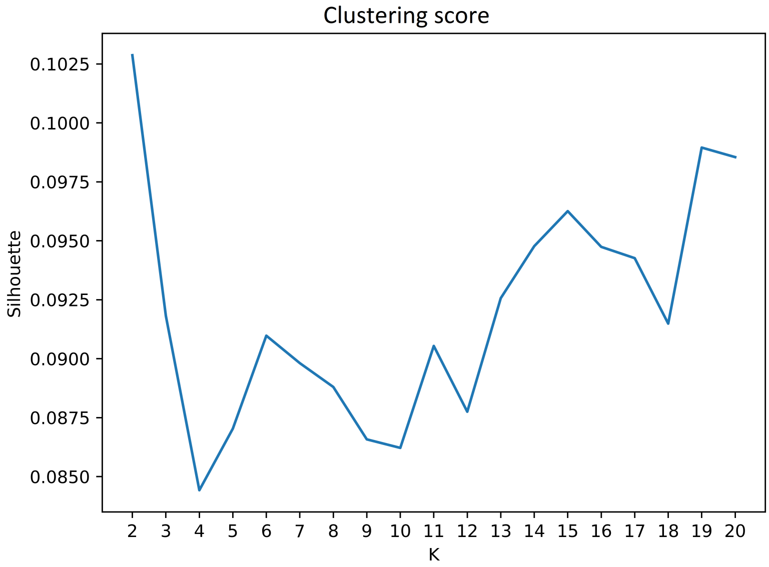


Figure 7. Average of silhouette coefficients criteria for $k = 2$ to 20 with min-count = 100

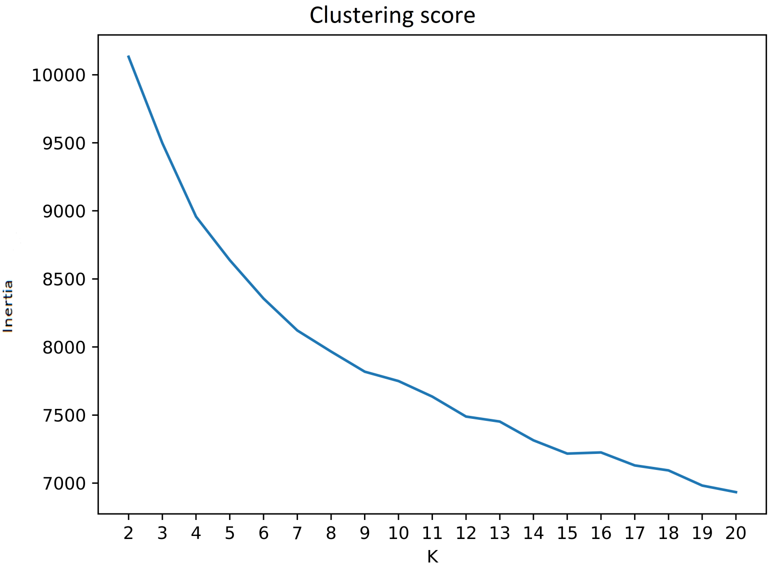


Figure 8. Inertia result with min-count = 500

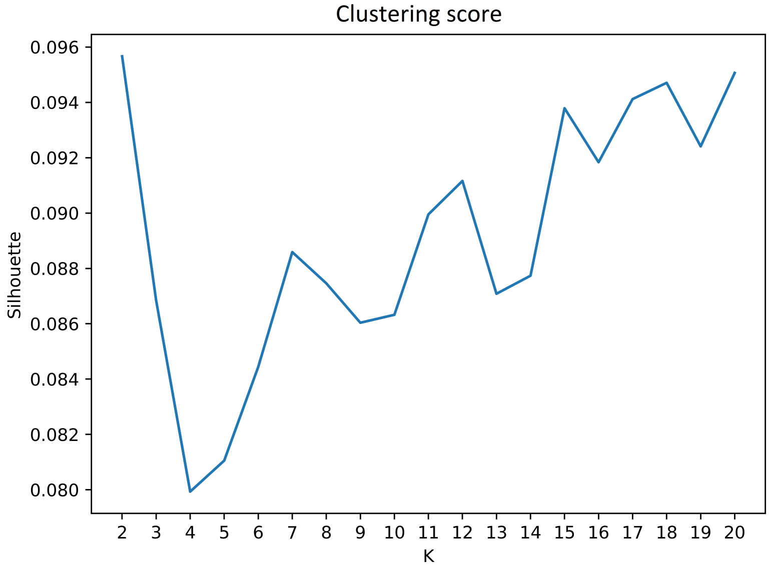


Figure 9. Average of silhouette coefficients with min-count = 500

Referring to Figures 7 and 6, one can easily notice that the optimal number of clusters k is $k_{opt} = 4$. We respectively obtained 368 043, 36 601, and 13 541 words in our vocabulary for min-count = 5, 100, and 500. This explains why the visualization of the results obtained by setting the min-count to 5 was not possible considering the vocabulary size. The Top 10 similar words to the centroids using the cosine distance are outlined in Table 3.

Centroid	Top 10 Similar Words
Complicit	enablers (83.65 %), complicity (81.71 %), lawlessness (79.04 %), reprehensible (78.67 %), shameful (77.99 %), criminality (77.59 %), disingenuous (77.23 %), hypocrisy (76.81 %), hypocritical (76.75 %), repugnant (76.63 %)
Eww	eeww (79.71 %), yuck (78.2 %), stinky (74.83 %), lmfao (74.81 %), lmao (73.64 %), ahahaha (73.52 %), barf (72.76 %), ahaha (72.37 %), shutup (72.16 %), lol (72.04 %)
Integrating	simplifies (84.94 %), integration (83.09 %), integrates (82.72 %), leveraging (80.76 %), integrated (80.29 %), utilizing (80.0 %), middleware (78.94 %), erp (78.59 %), adaptive (78.36 %), modernizing (77.93 %)
Elli	lynne (82.14 %), curtis (81.95 %), richardson (79.87 %), clark (79.18 %), phillips (78.76 %), connell (77.47 %), nichols (77.39 %), jerome (77.27 %), dickson (77.21 %), johnston (77.18 %)

Table 3. Top 10 similar words obtained with K-means

In summary, the results obtained with K-means are significantly better than LDA as they give a clearer vision on the dominant topics with the most similar words to that topic. Also, the words look semantically related. However, they remain ambiguous for the dataset annotation. This can be explained by the inconsistency of the tweets. This motivated us to reapply K-means on the existing topics and extract 3 subtopics for each topic. Once iterated, we obtained the following topics: *Development*, *Functionality*, *Fee*, *Funny*, *Clothes*, *Food*, *Insulting*, *Morality*, *Government*, *Place*, *Sport* and *Person*. In the next section, we propose a variant of K-means for the automatic annotation of tweets.

4.4 Automatic Annotation and Relevance Validation Using the Mean Method

In this subsection, we aim to provide a variant of K-means to choose the associated topic with a specific tweet based on similarity. The basic steps of this approach is outlined in Algorithm 3.

```
Algorithm 3: K-means based automatic annotation of tweets
1 Build a representative vector of tweets using the Word2Vec template to
    represent text as mean vectors
2 Compute the vector semantic similarity between tweets and
    topics/subtopics
3 Compute the cosine similarity between the words of the tweets and the
    mean vectors topics/subtopics
4 Label the tweet with the “closest” topic ($similarity > \delta$) with $\delta$ being
    a given threshold
```

As evaluation criteria, we used: the binary accuracy for subtopics, the subtopics smoothed accuracy, the subtopics Jaccard accuracy, the binary accuracy for topics, the topics smoothed accuracy, the topics Jaccard accuracy, subtopics precision, subtopics recall, subtopics F1-score, the precision of the topics, topics recall and topics F1-score. Regarding the threshold δ, we varied it from -1 to 1. Simulation results are summarized in Table 4. It is clear in this table that best results are obtained with a threshold $\delta = 0.1$. The statistics of this automatic annotation are reported in Table 5.

In the next section, we will experiment another annotation method which is the nearest neighbor.

4.5 Automatic Annotation and Relevance Validation Using the Nearest Neighbor Method

The basic idea is to choose a representative word of the tweet, thereafter label this tweet with the most similar topic to this word. For this, and for a finer granular-

δ	−1	−0.5	0	**0.1**	0.2	0.5	0.9	1
Accuracy subtopics Binary	0.1095	0.1095	0.3683	0.5485	0.7054	0.8535	**0.8905**	**0.8905**
Accuracy subtopics Smooth	0.0635	0.0632	**0.0648**	0.0622	0.0637	0.0639	0.0641	0.0639
Accuracy subtopics Jaccard	0.1095	0.1095	0.1194	**0.1206**	0.1186	0.0213	0.0	0.0
Accuracy topics binary	0.3085	0.3085	0.4499	0.5432	0.6257	**0.7094**	0.6915	0.6915
Accuracy topics smooth	0.1849	0.1849	0.1849	**0.1849**	0.1849	0.1849	0.1849	0.1849
Accuracy topics Jaccard	0.3085	0.3085	0.3265	**0.3395**	0.3265	0.1106	0.0	0.0
Precision subtopics	0.1095	0.1095	0.1245	0.1225	**0.1557**	0.0837	0.0	0.0
Recall subtopics	1.0	1.0	**0.778**	0.5255	0.379	0.772	0.0	0.0
F1 score subtopics	0.1947	0.1947	0.2109	**0.5255**	0.379	0.0772	0.0	0.0
Precision topics	0.3085	0.3085	0.3373	**0.3659**	0.3653	0.1323	0.0	0.0
Recall topics	1.0	1.0	**0.7796**	0.6227	0.4733	0.1151	0.0	0.0
F1 score topics	**0.4619**	0.4619	0.4531	0.4376	0.3904	0.1193	0.0	0.0

Table 4. Results of automatic annotation in Algorithm 3

lol (funny)	4 216 253
merely (Morality)	7 866 197
government (Government)	6 154 439
insulting (Insulting)	4 656 332
functionality (Functonality)	3 316 758
vat (Fee)	4 784 470
development (Development)	3 682 972
mariner (Sport)	983 138
asparagus (Food)	1 029 943
wool (Clothes)	1 851 799
riverside (Place)	1 012 534
curtis (Person)	648 857

Table 5. Statistics of the automatic annotation

ity, we make use of the subtopic. The basic steps of this approach is outlined in Algorithm 4.

```
Algorithm 4: Nearest Neighbor automatic annotation of tweets
1 Compute the nearest word to the other words in the tweet
2 Compute the subtopic with the highest similarity to the nearest word
3 Find the topic of the computed subtopic
4 Label the tweet with this found topic
```

For evaluation, we use the following standard criteria: binary accuracy, smoothed accuracy, Jaccard accuracy, precision, recall, and F1-score. Simulation results are summarized in Table 6. We can see from this table that the nearest neighbor method surpassed the mean method in terms of binary accuracy and precision, and slightly surpassed it in terms of Jaccard topic accuracy. Meanwhile, it shows similar results according to F1-score and slightly weaker results in terms of recall. We notice that this method represents a viable alternative to the mean method.

Binary accuracy	0.676
Smoothed accuracy	0.212
Jaccard accuracy	0.389
Precision	0.467
Recall	0.389
F1-score	0.415

Table 6. Results of the nearest neighbor method

As a summary to these experiments, we can say that K-means gave better results than LDA regarding topic extraction from tweets. This may be explained by the following facts: tweets are very short documents and are ill-structured by nature meaning that they do not follow grammatical and syntactic linguistic rules. Hence, unlike the case of classical structured and long documents, LDA is less efficient for tweets. Another major issue is the automatic annotation of tweets. Therefore, we have explored a K-means based approach and the nearest neighbor approach. Simulation results reveal that the latter is, in general, more efficient in the automatic annotation of tweets.

5 CONCLUSION

In this paper, we applied two unsupervised learning algorithms to topic extraction: LDA and K-means. Those algorithms were first implemented and applied to data extracted from Twitter. We then carried out several experiments to compare those algorithms. Regarding LDA, the obtained results (that are visualized using Pyldavis) were not easy to exploit and topics were not appropriate for a future annotation. Regarding the clustering algorithm K-means, we began by setting the optimal number of clusters k which is a major issue in this algorithm. In our case, the number of clusters represents the number of topics. For this, we adopted two common techniques: the elbow method and the average of silhouette coefficients. First simulation results were not satisfactory. So, we moved to a deeper level of granularity by considering the subtopics. We tested two annotation methods: the mean method and the nearest neighbor. Experimental results are very encouraging and should stimulate further investigations. Future work will entail applying our generic approach to other data and problems.

REFERENCES

[1] Paulraj, P.—Neelamegam, A.: Improving Business Intelligence Based on Frequent Itemsets Using K-Means Clustering Algorithm. In: Meghanathan, N., Nagamalai, D., Rajasekaran, S. (Eds.): Networks and Communications (NetCom 2013). Springer, Cham, Lecture Notes in Electrical Engineering, Vol. 284, 2014, pp. 243–254, doi: 10.1007/978-3-319-03692-2_19.

[2] Blei, D. M.—Ng, A. Y.—Jordan, M. I.: Latent Dirichlet Allocation. Journal of Machine Learning Research, Vol. 3, 2003, No. 1, pp. 993–1022.

[3] Xiang, B.—Zhou, L.: Improving Twitter Sentiment Analysis with Topic-Based Mixture Modeling and Semi-Supervised Training. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2014, pp. 434–439, doi: 10.3115/v1/P14-2071.

[4] Rosa, K. D.—Shah, R.—Lin, B.—Gershman, A.—Frederking, R.: Topical Clustering of Tweets. Proceedings of the ACM SIGIR 3rd Workshop on Social Web Search and Mining (SWSM '11), 2011.

[5] Kim, S.—Jeon, S.—Kim, J.—Park, Y. H.—Yu, H.: Finding Core Topics: Topic Extraction with Clustering on Tweet. 2012 Second International Conference on Cloud and Green Computing, IEEE, 2012, pp. 777–782, doi: 10.1109/CGC.2012.120.

[6] Jelodar, H.—Wang, Y.—Yuan, C.—Feng, X.: Latent Dirichlet Allocation (LDA) and Topic Modeling: Models, Applications, a Survey. 2017, arXiv: 1711.04305.

[7] Orkphol, K.—Yang, W.: Sentiment Analysis on Microblogging with K-Means Clustering and Artificial Bee Colony. International Journal of Computational Intelligence and Applications, Vol. 18, 2019, No. 03, Art. No. 1950017, doi: 10.1142/S1469026819500172.

[8] Hartigan, J. A.—Wong, M. A.: Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society: Series C (Applied Statistics), Vol. 28, 1979, No. 1, pp. 100–108, doi: 10.2307/2346830.

[9] Du, Z.—Wang, Y.—Ji, Z.: PK-Means: A New Algorithm for Gene Clustering. Computational Biology and Chemistry, Vol. 32, 2008, No. 4, pp. 243–247, doi: 10.1016/j.compbiolchem.2008.03.020.

[10] Kwak, H.—Lee, C.—Park, H.—Moon, S.: What Is Twitter, a Social Network or a News Media? Proceedings of the 19th International Conference on World Wide Web (WWW '10), 2010, pp. 591–600, doi: 10.1145/1772690.1772751.

[11] Selenium. Github, 2014, `https://github.com/SeleniumHQ/selenium.git`.

[12] Phongpanangam, O.: Twitter-LDA. Github, 2017, `https://github.com/panangam/Twitter-LDA`.

[13] Röder, M.—Both, A.—Hinneburg, A.: Exploring the Space of Topic Coherence Measures. Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15), 2015, pp. 399–408, doi: 10.1145/2684822.2685324.

[14] Sievert, C.—Shirley, K.: LDAvis: A Method for Visualizing and Interpreting Topics. Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces, ACM, 2014, pp. 63–70, doi: 10.3115/v1/W14-3110.

[15] Hidayatullah, A. F.—Ma'arif, M. R.: Road Traffic Topic Modeling on Twitter Using Latent Dirichlet Allocation. 2017 International Conference on Sustainable Information Engineering and Technology (SIET), IEEE, 2017, pp. 47–52, doi: 10.1109/SIET.2017.8304107.

Chaima Messaoudi holds a Master's degree from the Higher Institute of Computer Science and Telecom (ISITCom), University of Sousse, Tunisia. She received her Bachelor's degree from the Hammam Sousse School of Science and Technology, University of Sousse, Tunisia. She is currently pursuing a joint Ph.D. at the Higher Institute of Computer Science and Telecom (ISITCom) and the University of Reims. Her current research interests include opinion mining and influence maximization in social networks. She is a member of the research laboratory MARS (Modeling of Automated Reasoning Systems). She is also a member of the Centre de Recherche en STIC (CReSTIC) of the University of Reims Champagne Ardenne.

Zahia Guessoum is Associate Professor at the University of Reims Champagne-Ardenne. She received her doctorship/Ph.D. (1996) and then her "habilitation à diriger des recherches" (2003), both in computer science and from the University Pierre and Marie Curie (Paris 6), France. She is the Head of the MODECO Team of "Centre de Recherche en STIC" (CReSTIC) at the University of Reims Champagne-Ardenne. Her general research interests are about adaptive agents and multi-agent systems, fault-tolerant MAS, multi-agent oriented software engineering and coordination mechanisms. She authors more than 140 journal and conference papers in the fields of AI and multi-agent systems.

Lotfi BenRomdhane is currently Professor in computer science at ISITCom, University of Sousse, Tunisia. He holds a Ph.D. degree from the University of Sherbrooke, QC, Canada, and the engineering degree from ENSI, Tunisia, both in computer science. His research interests fall within the general area of data science and social networks analysis. He published more than 100 papers in international ranked conferences and impacted journals. He was awarded the CIDA fellowship to pursue his Ph.D. degree at the University of Sherbrooke, QC, Canada. He currently heads MARS (Modeling of Automated Reasoning Systems) Research Lab in computer sciences.