Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 207 (2022) 790–799

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

Using Topic Modeling and Word Embedding for Topic Extraction in Twitter

Amna Meddeb[a], Lotfi Ben Romdhane[b]

[a]*ISITCom, MARS Research Lab LR17ES05, University of Sousse, Tunisia*
[b]*ISITCom, MARS Research Lab LR17ES05, University of Sousse, Tunisia*

Abstract

Topic analysis (also called topic detection, topic modeling, or topic extraction) is a machine learning technique that organizes and understands large collections of text data, by assigning "tags" or categories according to each individual text's topic or theme. Topic analysis uses natural language processing (NLP) to break down human language into blocks (speech, words, sentences, context) so that you can find patterns and unlock semantic structures within texts to extract insights and help make data-driven decisions. The two most common approaches for topic analysis with machine learning are NLP topic modeling and NLP topic classification. Topic modeling faces several challenges: some are general to the NLP task (as extracting the context of document), and some are specific and are related to the nature (or properties) of the documents. One particular type of documents that raises several challenges are short-text documents that we find in Social Networks. First, we should note that with the growth of online social network platforms and applications, large amounts of textual user-generated content are created daily in the form of comments, reviews, and short-text messages. However, despite their ubiquity, extracting topics from shorts texts remains a difficult task for several reasons. First, unlike traditional normal texts, short texts typically only include a few words. Therefore, directly applying traditional models on short texts will suffer from the severe data sparsity problem (i.e., the sparse word co-occurrence patterns in individual document). Second, the limited contexts make it more difficult to identify the senses of ambiguous words in short texts. Third, in general, the performance of the used machine learning models relies on labeled data. Unfortunately, due to their volume, labeling short-text from social networks remains a tedious and hard task. In this paper, we propose a model for extracting topics in short-texts, and more specifically in Twitter. The key feature of our proposal is the use of word embedding technique for topic modeling, and k-Means clustering for the semi-automatic annotation of tweets. In addition, and unlike most existing approaches, we assign in our model a set of topics to a tweet each with a confidence degree.

Keywords: topics extraction; topic modeling; topic classification; natural language processing; semi-automatic annotation; Twitter

* Corresponding author. : +2-165-536-9080
E-mail address: amna.meddeb@isitc.u-sousse.tn

1877-0509
10.1016/j.procs.2022.09.134

1. Introduction

Natural Language Processing (NLP) is defined as the field that combines the domains of computer science, artificial intelligence, and linguistics to build systems that can process and understand human language [11]. The natural language is composed of several blocks [11] for each level we distinguish a variety of NLP applications. Phonemes represent the speech or pronounced words with a sound. Phonemes' NLP applications are converting Speech to a text, or text to speech or identifying the speaker, etc. Morphemes and lexemes represent words, and word-embedding, tokenization, etc are some of the NLP applications that deal with words. Syntax represents the next block of language, and we distinguish entity extraction, relation extraction as some NLP applications that deals with the latter. The context represents the highest level, and we distinguish summarization, sentiment analysis, information retrieval, topic modeling, topic classification as some of its NLP applications.

The two most common approaches for topic analysis with machine learning are NLP topic modeling and NLP topic classification. Topic classification is a supervised technique where we train a model with some of our labeled data and based on that training, the model will try to classify the unlabeled data. Generally, text classification techniques deal with a limited and already fixed number of values for example for credibility the model will be able to predict a label (credible, or not credible), or for movies rating based on reviews the model can predict a rating value (bad, ok, or good). The problem of text classification here is that we have a predefined set of topics or labels. However, when the set of topics is unknown apriori, then we need topic modeling that can be defined as a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. Topic modeling analyses unlabeled data and extracts topics from it (without knowing already the topics' labels and it's generally an unsupervised Machine Learning technique).

Topic modeling can be categorized using several criteria as the nature of document collection (i.e., long text documents vs short-text documents). In this paper, we propose a model for topic modeling using short-text documents. Our model is composed of three main steps. The first step aims at reducing the noise in the data. In the second step, we perform topic modeling as well as build a word2vec model. In the third step, we extract the topics from each tweet using the models developed in step 2; and for each topic we assign a confidence degree. The rest of this paper is structured as follows. Section 2 outlines related work while section 3 details our proposal. An extensive experimentation is conducted in Section 4 and the final section offers concluding remarks and future research directions.

2. Background and Related work

The main purpose of this section is to outline the main research in the field of topic extraction. We will classify related research using the nature of the document; and thereby distinguish two main classes: approaches dealing with "long-text" documents, and those dealing with "short-text". But, before going any further, let us remind that topic modeling requires NLP processing as an initial phase. Hence, and for the shake of readability, we need to briefly remind in the next section the basic principle of fundamental NLP tasks required for topic modeling.

2.1. Natural Language Representation

As a first task, we need to represent text documents into a suitable format comprehensible by the machine. There are mainly two representations: Vector Space Model (VSM) and Term Frequency–Inverse Document Frequency (TF-IDF).

The VSM was introduced by Gerard Salton et al. [8]. It allows us to present phrases as vectors in an n-dimensional space where n is the number of all unique words in the input data. To generate the document's vector, we first extract the unique words of all the documents (n-words). Second, for each document d, we build a vector with n columns (corresponding to the unique words). The values in each column represent the number of occurrences of the column's word in the document d. Hence, VSM is an algebraic model for representing text documents as vectors of identifiers (or index terms) and is a suitable representation for the machines. However, the main drawback of VSM is that it doesn't consider the relevancy of the words regarding a document. For example, if we consider the term "the", each document will have a high frequency of this term however it's not a relevant word.

The second known representation is the "Term Frequency–Inverse Document Frequency" (TF-IDF). It is a combination of two measures: (i) the Term frequency (TF) that simply counts the number of occurrences of a word in a document (like VSM); and (ii) the Inverse Document Frequency (IDF) which measures a term's informative power by investigating its frequency over all documents. Stated otherwise, TF-IDF evaluates the importance of a word based on its occurrence in a document compared to its occurrence in all documents. Hence, a term becomes important only if it is frequent in a document; and not frequent in the rest. Therefore, TF-IDF resolves the problem of document-words relevancy [9]. . Now, we are ready to outline the main approaches for topic modeling using the length of the document as a classification criterion. In the next section, we will outline the main approaches for classical "long-text" documents.

2.2. "Long-text" Topic Modeling Approaches

Long text topic modeling methods work well with lengthy documents as input. Hereafter we outline the most known algorithms in the field.

Latent Semantic Analysis (LSA) was introduced by Bellegarda in 2005 [2]. LSA relies on the following main idea: words with similar meanings will occur in similar pieces of text. Therefore, LSA decomposes the document-terms matrix into two separate matrices: the first represents the document-topic matrix, and the second represents the topic-term matrix. Matrix decomposition in LSA is based on Singular Value decomposition. Two main key properties make LSA attractive. The first key property is related to the mapping of the documents (in a discrete space) onto a continuous parameter space which is very beneficial to machine learning algorithms. The second key property is the dimensionality reduction inherent in the process. Dimensionality reduction helps uncover important structural aspects of a problem while filtering out "noise". However, a major drawback of LSA is its expensive computational cost.

Latent Dirichlet Allocation (LDA) was introduced by Blei *et al.* in [3]. The main idea of LDA is to generate documents composed of corpus words randomly chosen from a mixture of latent topics. After generating the documents, the LDA model investigates the documents-topic distribution and the topics-word distribution using Gibbs sampling. The purpose of Gibbs Sampling is to ameliorate the obtained distributions in a way that the distance between words in the same latent topic is minimized and the distance between words from different latent topics is maximized. In [12] the author introduced some of the advantages and disadvantages of LDA and mentioned some works that can overcome some of the drawbacks of LDA. The major benefit of LDA is that it can deal efficiently with the variation of both words and documents [12]. However, LDA doesn't label the detected topic and the interpretability of topics can be quite difficult. To overcome this shortcoming, the author in [12] mentioned the Labeled-LDA model which is a text classification approach where LDA is trained using labeled data; therefore the number of topics is fixed before training. We should mention that by doing this way, the labeled-LDA becomes a supervised model (or a text classification model) where the topics are known apriori. Another major concern is that LDA doesn't consider the correlation between the detected topics. Therefore, Correlation Topic Models [12] detects the interdependency between topics via covariance matrix computation. Finally, we should note that LDA doesn't detect the evolution of topics in time-organized data. Therefore Dynamic Topic Models, [12] do consider the evolution over sequentially organized documents. [6] investigated three topic modeling models (LDA, LSI, and HDP): they were applied to the same dataset, and LDA gave the best results regarding the coherence evaluation metric.

2.3. "Short-text" Topic Modeling Approaches

Nowadays, short-text documents are ubiquitous mainly due to the tremendous growth of social networks and their daily use almost by everyone. In this kind of platform, short texts are available in the form of comments, reviews, and short text messages. The application of the approaches for topic extraction in long-text documents to short-text ones was deceiving for several reasons [1]. One major obstacle is the length of short-texts which include by nature a few words. Hence, traditional models will suffer from *data sparsity* problem (i.e., the sparse word co-occurrence patterns in the individual document). Another fact is that the context of short texts is very limited and therefore identifying the sense of ambiguous words remains non-trivial. Another major issue is related to the nature of the machine learning models: their performance relies on labeled data. Unfortunately, due to their volume, labeling short text from social networks remains a tedious and hard task. Finally, short texts (especially in social networks) are linguistically very noisy since they embed unstructured text, URLs, emoticons, abbreviations, etc. Therefore, eliminating the noise from

this text becomes fundamental before applying any approach for topic modeling. Unfortunately, separating text from noise is not trivial and may cause a loss of information. In this section, we will outline the main recent research for topic modeling in short-text documents.

In [7] the authors classified short text modeling methods into 3 classes, the first class is based on the idea that each document is inferred from only one topic (contrary to LDA that assumes that each document is composed with a variety of latent topics), we can mention the work mentioned in the related work of [13] which is sentence-LDA that uses the classical LDA algorithm with the only difference that sentence LDA relies on the hypothesis that each document is inferred from only one topic. This type of methods doesn't really resolve the sparsity problem. The second class which the author call Global Word Co-Occurrences Based Methods can enrich the information within the short text by investigating words' contexts. The main idea here is documents with similar context tend to share the same topics. In this section we can mention the work of [13]. The proposed model is named FSL FastText-based Sentence-LDA. It surmounts the previously mentioned sparsity and ambiguity of words problems by using a word embedding model which is FastText. FastText captures similarities between words. It associates each word with a group of similar words with a similarity degree or weight. The main idea of using FastText is to enrich the short text by the detected similar words. As a result, we will have an enriched "short text" and that is what solves the sparsity problem at the same time the data is enriched with similar words, and that is what solves the ambiguity of words problem. After enriching the text, the labeling problem will be resolved using Sentence-LDA mode, where the topics of the documents are extracted based on a statistical approach like LDA but with the main hypothesis that each short text is composed of one topic contrary to LDA which suppose that a document is composed of several topics. The result of this approach is clusters of similar words where each cluster will represent a topic. The main disadvantage of this approach is that the obtained clusters are unlabelled and it's quite hard to label the clusters. Also, another disadvantage is that the noisy data problem is not treated (lack of cleaning the noise within the input data).The third class of methods is based on the idea that a short text can be merged in a long text, so we can make advantage of the rich co-occurrence information within long texts.

In [5], the authors compared the work of different topic modeling algorithms used with short text documents dataset about health. The evaluation metrics that can be used in such comparaison are the Calinski-Harabaz index and the Silhouette Coefficient. For the first measure, it is a ratio between the average (inter and intra)-cluster dispersion matrix. Where the inter-cluster is based on the distance between clusters and the average intra-cluster id is based on the distance within clusters. The second measure studies the separation distance between the resulting clusters.

In our proposed model we extract short text documents topics (tweets) starting with cleaning the noisy data, making use of a word embedding model to overcome supercity and words ambiguity problem and also we apply a topic modeling algorithm proceeded with manual labeling to surmount the labeling problem.

3. Proposed model

The objective of our proposed model is to extract a short text document's topics using a word embedding and topic modeling technique. First, we start with introducing the main hypothesis of this proposed model then we represent the overflow of our model proceeded with a detailed explanation of each step.

3.1. Hypothesis

The main hypothesis of our proposed model is that a document can be about several topics.

3.2. General flow of the proposed model

In this section, we represent the general flow of our model. In the first step we take as input a collection of documents, we preprocess them to obtain as output a group of documents cleaned from noisy information, a dictionary, and the Bag-Of-words representation of all the documents. In the second step, we have two parallel parts. A part where we extract the topics of the input collection of documents and another part where we train a word2vec model to be able to measure the similarity between words later. The last step is the documents' topics extraction step where we define the topics of each document from the initial collection (see figure 1) in the range of the topics extracted in step2.

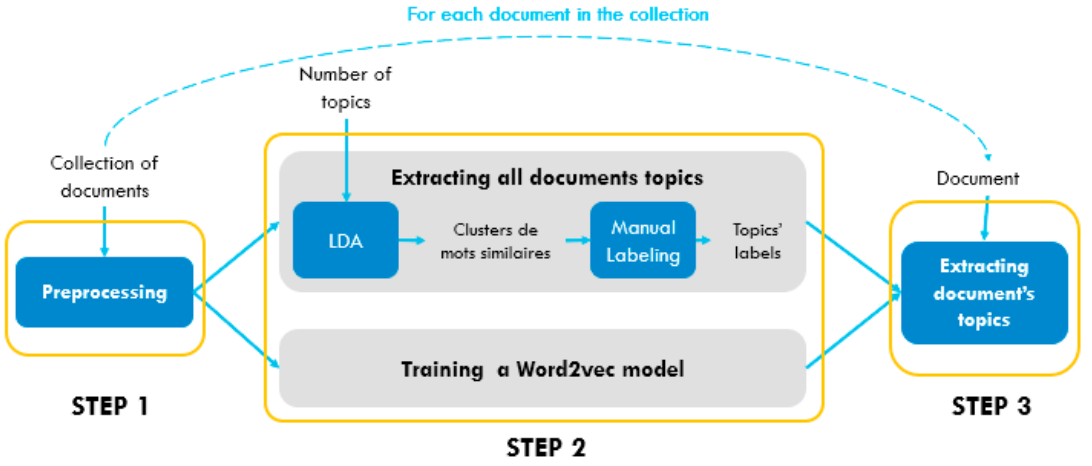

Fig. 1. Model architecture

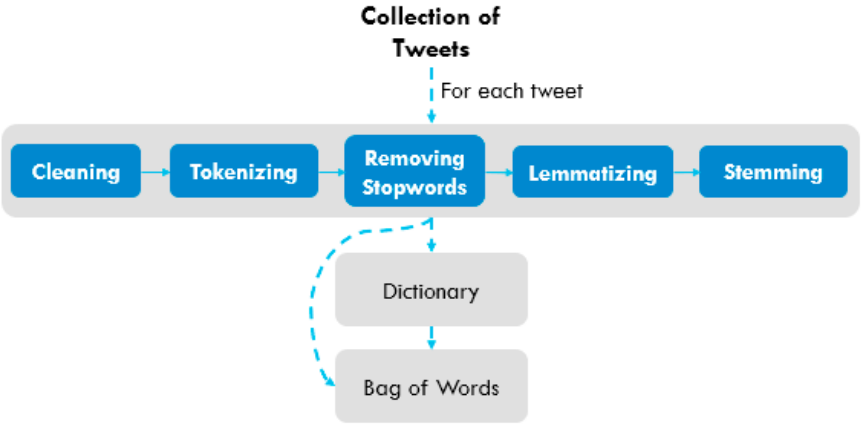

Fig. 2. Document preprocessing

3.3. STEP1: Preprocessing

In the first step, we take a collection of documents (documents) we clean and make some NLP tasks to avoid noisy data. Each Document is converted into a group of words following the steps shown in figure 2: cleaning, tokenizing, removing StopWords, lemmatizing, and stemming. In cleaning, we remove symbols(#,@,\), URLs, and new line characters. In Tokenization, the document text is split into lowercase words, and punctuation is removed. In StopWords removing we delete words that do not express a meaning such as the, a, me, etc. In lemmatization, words are changed to the first person, and verbs are changed into a present tense. and in stemming, words are reduced to their root form. After preparing our documents and dividing them into words. Unique words within all documents will be mapped with a unique id in a dictionary.
After that, the dictionary will help us extract a Bag-Of-Words (BOW) for each document. where BOW represent the group of words' ids and their number of occurrences in a document.

3.4. STEP2: Topic modeling & word embedding

In this step we extract the topics within the preprocessed collection of documents and also, we train a word2vec model. These two tasks are independent and that's what explains the parallel representation in figure 1.

3.4.1. Topic modeling

The main idea of topic modeling is to extract the documents' main topics. To the best of our knowledge, no algorithm can model documents' topics and label them. In our model, we do so starting with extracting the topics' groups of words using the LDA model [4]. LDA will take the preprocessed documents as input and then will cluster the words within the documents in a way words within the same cluster have similar meanings (the meaning here is analysed using the context where words having similar context tend to be similar). Each word in the cluster is associated with a probability representing a confidence degree (how much the word is relevant to the cluster's topic)
The obtained clusters represent the different topics in the documents but as mentioned in [12] one of the main disadvantages of the LDA model is that it doesn't label the topics. To resolve this issue, we tried to label manually the obtained clusters or groups of words. We took into consideration that a cluster can be meaningless and several clusters may have similar labels. thus, we overcome the problem of unlabeled topics. After labeling the obtained clusters we extracted the most representative words of each topic.
As a final result, we obtain the documents' topics and for each topic, we have the label and the most representative words of that topic.

3.4.2. Word embedding

In this step, we train a Word2vec model. As input, we take the group of preprocessed documents and as output, the word2vec model will make a mapping between similar words, more specifically, for each word we will have a vector of similar words with their similarity scores. this way we resolve the sparsity problem where our short text is enriched with its words vectors. and also, the problem of words ambiguity can be resolved since the obtained vector will give us a vector of similar words (semantic similarity)

3.5. STEP3: Extracting documents topics

In this step, we will take each document from our collection (see figure 1) and we will extract its correspondent topics. The main idea here is to detect documents topics in the range of the topics we extracted in section 3.4.1.

To extract the topics of a document we first start with preprocessing (following the cleaning steps in section 3.3) to obtain the representative words of the document.

To calculate a document d's score in a single topic t, we measure the average similarity between the most representative words of topic t (extracted in section 3.4.1) and the document d's relevant words. Calculating a binary similarity between words cannot detect semantic similarity (such as comparing the words "HDD" and "SSD" in fact these two words are different but have almost similar meanings) to detect the semantic similarity we use the word2vec model that we trained in section 3.4.2. As a result the function that allows us to measure a document doc's score in a topic t is:

$$doc\_topic\_score(d,t) = word2vec.Similarity(representative\_words(d), representative\_words(t)) \quad (1)$$

After calculating the document's score in the range of the detected topics (topics detected in section 3.4.1) we obtain as a result a vector where each row represents a topic, and the column-row values represent the score of the document in the row's topic. From a topics' vector of document d, we will extract the most representative topics by fixing a threshold th. So, if the topic's score is greater than th so the latter is one of document d's relevant topics.

$$relevant\_topics(d,th) = \{t \in Topics : withdoc\_topic\_score(d,t) \geq th\} \quad (2)$$

After calculating all relevant topics, we ignore the irrelevant ones (they will take the value 0 in the vector) and we normalize the obtained vector.

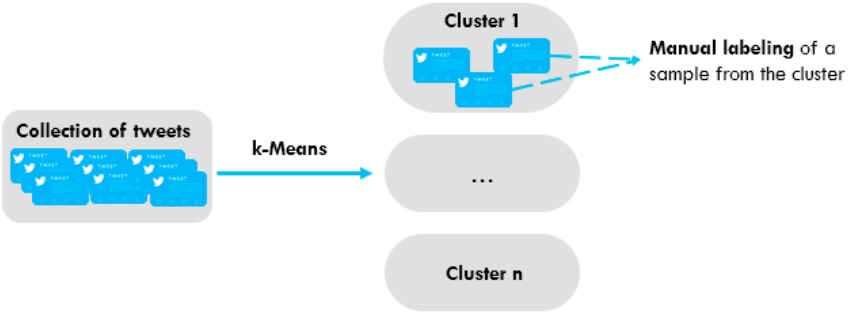

Fig. 3. Semi-automatic annotation

4. Experimentation

In this section, we present the environment, the data used to experiment with our model, how we annotated the data, and finally our model's results.

4.1. Environment

We experimented our model using the distribution; Anaconda, where we can find the programming language python enriched with data science libraries and the notebook Jupiter. The code is implemented on a machine with Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59 GHz processor and 8,00 Gb of RAM

4.2. Semi-automatic annotation

For the data we used 114826 tweets as our short text documents and to collect them we used Twitter API [1]. To label the topics of the collected data we used a semi-automatic annotation technique. The main idea behind the latter is to try to cluster documents or tweets, using k-means, in a way documents within the same cluster are similar to each other. After clustering our documents, we will try to take a few documents (sample) from each cluster and annotate them manually. After the manual annotation, all documents within the same cluster from which we took the sample will take the label of its correspondent sample since all documents in one cluster share similar properties (see figure3).

4.3. Obtained results

To make the evaluation, we compare the predictions of our model with the annotations we obtain following section 4.2. LDA needs a parameter k which represents the number of topics within the input collection. To fix the optimal value of k we tried different k values and we measured the coherence c_v score (see figure 4). In fact, in [10] the author explained the importance of the c_v coherence score based on a study that compare the different coherence evaluation metrics with human rating scores, and c_v coherence score was the measure that is most correlated with human ranking.

From figure 4 we notice that the optimal k value is 10 since it corresponds to the highest coherence c_v value (0.396). Thus, we take the LDA model that corresponds to k=10 and we annotate the clusters manually as mentioned in section 3.4.1 (two clusters may have the same label and can have a "None" label which means there is no significant meaning in that group of words). we choose the most relevant words from each cluster, and we obtain as result: 6 topics labels and their correspondent relevant words. After extracting the topics of all the tweets, we calculate the tweet's score in the range of the detected topics. As a result, we have our model's predictions also the tweets' topics annotations (labels) (section 4.1). But before comparing both results, we notice two things: The first is our comparison

[1] Twitter has its API for researchers to access its content

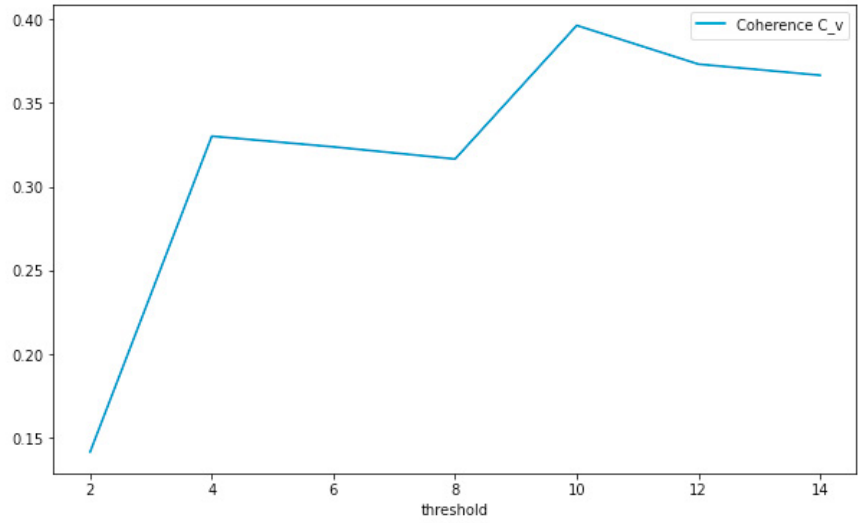

Fig. 4. Optimal k based on C_v score

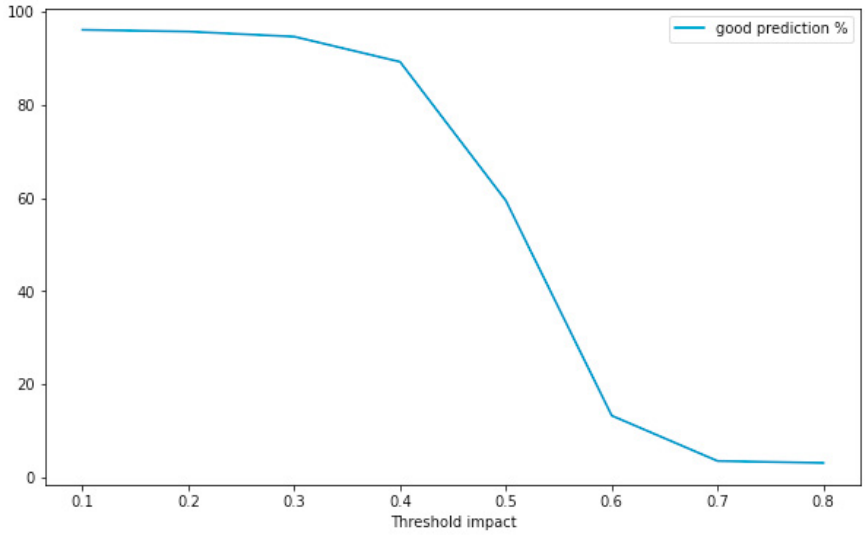

Fig. 5. Threshold impact on the percentage of good predictions

is not a binary comparison so we will not be able to calculate accuracy, precision, etc. however we can calculate the number of correct predictions (count the documents that have similar topic labels in both predictions and annotations). The second thing is while comparing prediction we notice that we cannot compare a vector with a single value; that's in our model we extract different topics for a single document while in the annotation, one document is annotated with one label. To resolve this issue, we considered that if the annotation label of a document is in one of the predictions labels the prediction will be considered true.

The number of predicted labels strongly depends on the fixed threshold (see equation 2) thus the need to analyse the impact of the threshold.

To do so, we varied the threshold value and for each value, we calculated the percentage of good predictions. We notice in figure 5 that the more the threshold increase the more the percentage of good prediction decrease. that's because if the threshold has low values so it's more likely that all topics will be relevant for the tweet (see equation 2). So as a result, most of the tweets will have the labels of all topics and based on our evaluation approach (if the

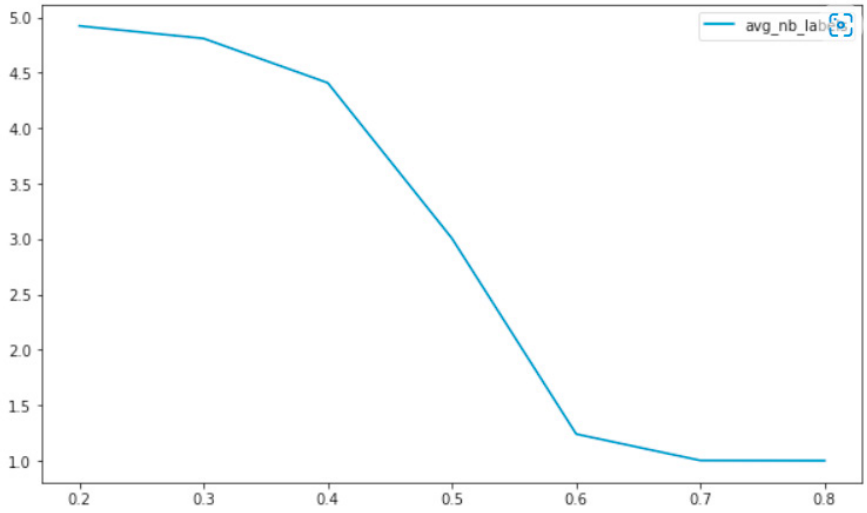

Fig. 6. Average number of labels per tweet

annotation label is in one of the prediction labels, the prediction will be considered as true) we will always find the annotation label in the list of predictions labels and that's what explain the high percentages of good predictions with lower thresholds. The problem here is that with low threshold values the relevancy of topics is not reliable. so, we need to maximize the threshold value. We notice in figure 5 that with high values we obtain bad results (% of good predictions) and that's because if the threshold is high so the tweet will have no relevant topics and so will consider a tweet without topics.

So a good threshold value corresponds to the value that maximizes the percentage of good predictions and at the same time minimizes the number of predictions labels per tweet (or maximizes the threshold value to obtain relevant topics). To make a good decision we need to analyse the number of predicted labels in function of the threshold.

In figure 6, we confirm our analysis of the plot shape in (fig5) where with low threshold values, the number of relevant topics is high and that is not reliable and for high threshold values, we notice that the average number of relevant topics is 0. So, the best value to choose is 0.5 since it minimizes the number of labels "3" and at the same time it maximizes the threshold values so the topics are relevant. With a threshold of 0.5, we obtain: the number of correct predictions is 67259 out of 114826 and so the percentage of correct predictions is 58,57%

5. Conclusion

In this paper, we proposed a new documents' topic extraction model (for short texts). We first started by a cleaning process to make sure we eliminate the noise within the data after we apply topic modeling on our processed data and then we label the obtained topics manually to overcome the major shortage of topic modeling algorithms which is unlabeled detected topics. In parallel, we train a word2vec or a word embedding model that will be used to calculate the similarity between the topic's representative words and the document's representative words. the main interest of word2vect is to overcome the sparsity and the ambiguity problems that we face with short texts. the sparsity problem is resolved since the word embedding technique enrich a short text with its words vector that contains a group of other words, and the word ambiguity problem is resolved by the fact that the vector obtained contain words that are similar in term of semantics.

In fact, our proposal remains in between topic modeling and topic classification where we classify tweets in a range of detected topics. The strong point of our model is that the topics are not fixed at the beginning such as in topic classification models instead they are detected using a topic modeling model proceeded with manual labeling to surmount one of the main problems with topic modeling algorithms which is unlabeled topics in the result. Also, our model overcomes the problems of sparsity and words ambiguity in short-text documents by using a word embedding model.

In the experimentation, we've made a comparison between our models' predictions and semi-automatic annotations results, meanwhile, we varied several parameters to observe their impact on the final result and the best measure we obtained after analysing all experiments is 58.57% of correct predictions.

This model can be a clue for a wide variety of research topics, especially in the field of analysing Online Social Networks users where the interest of a user can be detected analysing the content he creates (especially knowing that in the majority of OSN, users tend to create short text content such in Twitter)

References

[1] Albalawi, R., Yeap, T.H., Benyoucef, M., 2020. Using topic modeling methods for short-text data: A comparative analysis. Frontiers in Artificial Intelligence 3, 42. URL: https://www.frontiersin.org/article/10.3389/frai.2020.00042, doi:10.3389/frai.2020.00042.

[2] Bellegarda, J., 2005. Latent semantic mapping [information retrieval]. IEEE Signal Processing Magazine 22, 70–80. doi:10.1109/MSP.2005.1511825.

[3] Blei, D.M., Ng, A.Y., Jordan, M.I., 2003a. Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022. URL: http://portal.acm.org/citation.cfm?id=944937, doi:http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993.

[4] Blei, D.M., Ng, A.Y., Jordan, M.I., 2003b. Latent dirichlet allocation. Journal of machine Learning research 3, 993–1022.

[5] Lossio-Ventura, J.A., Morzan, J., Alatrista-Salas, H., Hernandez-Boussard, T., Bian, J., 2019. Clustering and topic modeling over tweets: A comparison over a health dataset, in: 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE. pp. 1544–1547.

[6] Manias, G., Mavrogiorgou, A., Kiourtis, A., Kakomitas, D., Kyriazis, D., 2021. Real-time kafka-based topic modeling and identification of tweets, in: 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), IEEE. pp. 212–218.

[7] Qiang, J., Qian, Z., Li, Y., Yuan, Y., Wu, X., 2020. Short text topic modeling techniques, applications, and performance: a survey. IEEE Transactions on Knowledge and Data Engineering .

[8] Salton, G., Wong, A., Yang, C.S., 1975. A vector space model for automatic indexing. Communications of the ACM 18, 613–620.

[9] Scott, W., . Tf-idf from scratch in python on a real-world dataset. URL: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089.

[10] Syed, S., Spruit, M., 2017. Full-text or abstract? examining topic coherence scores using latent dirichlet allocation, in: 2017 IEEE International conference on data science and advanced analytics (DSAA), IEEE. pp. 165–174.

[11] Vajjala, S., Majumder, B., Gupta, A., Surana, H., 2020. Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems. O'Reilly Media.

[12] Xia, L., Luo, D., Zhang, C., Wu, Z., 2019. A survey of topic models in text classification, in: 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD), IEEE. pp. 244–250.

[13] Zhang, F., Gao, W., Fang, Y., Zhang, B., 2020. Enhancing short text topic modeling with fasttext embeddings, in: 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), IEEE. pp. 255–259.