Journal of Universal Computer Science, vol. 27, no. 11 (2021), 1203-1221
submitted: 11/2/2021, accepted: 9/8/2021, appeared: 28/11/2021

Incremental autoencoders for text streams clustering in social networks

Amal Rekik
(Multimedia InfoRmation systems and Advanced Computing Laboratory, MIRACL, Sfax University, Tunisia
Digital Research Center of Sfax, DRCS, Sfax, Tunisia
https://orcid.org/0000-0002-9505-5930, rekik.amal91@gmail.com)

Salma Jamoussi
(Multimedia InfoRmation systems and Advanced Computing Laboratory, MIRACL, Sfax University, Tunisia
Digital Research Center of Sfax, DRCS, Sfax, Tunisia
https://orcid.org/0000-0003-1521-3351, jamoussi@gmail.com)

Abstract: Clustering data streams in order to detect trending topic on social networks is a challenging task that interests the researchers in the big data field. In fact, analyzing such data needs several requirements to be addressed due to their large amount and evolving nature. For this purpose, we propose, in this paper, a new evolving clustering method which can take into account the incremental nature of the data and meet with its principal requirements. Our method explores a deep learning technique to learn incrementally from unlabelled examples generated at high speed which need to be clustered instantly. To evaluate the performance of our method, we have conducted several experiments using the Sanders, HCR and Terr-Attacks datasets.

Keywords: Social network, Topic extraction, Data streams, Clustering, Deep learning, Incremental autoencoders, Stacked autoencoder
Categories: I.2.7,I.2.4,I.5.3, I.2.6
DOI: 10.3897/jucs.76770

1 Introduction

Today, social networks have become a fact in our society. They play an empowering role and have a tremendous impact on our lives. In fact, social networks have become a modern and revolutionary tool for communication and for sharing news on a very wide scale and through a variety of themes. Recently, such platforms have played a major role in our society mainly in the exposition of oppressive political regimes and even the change of the world. In regards of this emerging popularity of social media, processing and analyzing the sharing data entered as data streams, namely for extracting the hot topics under discussion on these networks, is a challenging task for researchers. The data stream is described as a potentially infinite non-stationary data element sequence ($x^{(1)}$, $x^{(2)}$,…$x^{(n)}$,...) where the objects of the stream $x^{(i)} \in R_n$ can be represented by a vector of attributes of length d, $x^{(i)} = [x^{(i)}_1, x^{(i)}_2, x^{(i)}_3, \ldots, x^{(i)}_d]$. This data stream is characterized by the high volume of continuous data, the quick rate of arriving data and the presence of noisy data. Hence, the problem of exploiting this large amount of data in social networks, has been raised. This form of data has given place to several researches

that aim at establishing tools for mining data streams namely for extracting hot topics. The problem of text clustering can be very complex where the text to be clustered can be of different granularities and written by different users. In addition, data stream presents multiple challenges such as the unknown feature space and the change of distribution over time known as the concept drift problem. Due to these facts, data streams clustering algorithms raise fundamental issues to be addressed such as the requirements of speed and memory, the detection of the concept drift and the identification of outliers.
Achieving text stream clustering in order to detect hot topics is the key aim of this paper. So, we propose a new evolving method of clustering called Stream-Clus that explores a deep learning technique namely the stacked autoencoders. Our method takes into account the feature evolution and learns incrementally from unlabeled examples that need to be clustered instantly. This paper is an extension of a previous work presented in [Rekik et al. 2016]. The extensions include : (1) The emphasis on the concrete aspect of the deep learning process. (2) The improvement of the approach and the going into detail of each step. (3) The introduction of examples that simplify and intuitively describe the method. (4) The extension of the experiments by preparing a new dataset as well as conducting a series of new experiments to assess the method's performance.(5) The inclusion of a discussion section where we have highlighted our contributions.
The reminder of this paper is structured as follows: the next section is devoted to the related work. In section 3, we give an overview of deep learning techniques and especially Autoencoders. In section 4, we explore our method in further details. Section 5 reports the findings of the experimental results and discusses them. Finally, section 6 concludes the paper.

2 Related work

Data strams clustering algorithms require generally an efficient process of partitioning data continuously taking into account memory and computational time restrictions. In this context, several research has been proposed . This section presents a review of some previous works related to the target issue. In [Cao et al. 2006], authors proposed an algorithm called DenStream for discovering clusters in an evolving data stream where the core-micro-cluster is introduced to summarize the clusters while the potential core-micro-cluster and outlier micro-cluster structures are proposed to maintain and distinguish the potential clusters and outliers. However, DenStream requires adjusting many parameters to have a good clustering result. Authors, in [Chen et al. 2007], proposed a clustering algorithm called Dstream which uses a grid-based approach. It maps each input data into a grid and computes the grid density. Then, all grids are clustered based on their densities using a density-based algorithm. Dstream adopts a density decaying technique to capture the changes over the data stream, but it is not very appropriate for the textual data. In [Aggarwal et al. 2003], the proposed CluStream algorithm is separated into online and offline components. It continuously maintains a fixed number of micro-clusters. The online component periodically stores detailed summary statics and the offline component uses only this summary statics and performs clustering using the k-means algorithm. However, CluStream predefines a constant number of micro-clusters and it is also risky when the data contains noise. Authors, in[Tareq et al. 2020], proposed CEDGM algorithm for discovering the clusters of evolving data streams in multi-density environments. During the first step, authors pursue a density-grid based procedure in order to produce the Core Micro-clusters (CMCs). Hence, a simple process is used to minimize the life of the CMCs and allow the removal of unused CMCs. In the second

phase, The suggested approach requires merging the CMCs into global macro clusters. Moreover, in order to handle multi-density data and noises, the grid-based procedure was used as an outlier buffer. Authors, in [Adepu et al. 2019], proposed an Improved Differential Evolution algorithm (IDE) for the data stream clustering. The first phase involves determining the closest cluster center for each object from the input data stream and updating the clusters. Any concept drift occurs around then, the approaching objects are placed in the buffer before a specified amount of time is reached. In order to find the optimal number of clusters, the authors adopt an IDE-based optimization. In [Abid et al. 2018] Abid et al proposed a novelty detection method in data stream clustering called AIS-Clus that uses the artificial immune system (AIS)-meta heuristic. For each incoming data, a scoring function is calculated in order to examine the affinity value of each cluster. The AIS is involved to determine which cluster will absorb this novel invader by applying both the clonal selection principle which is based on the production of antibodies to better match the new invader, and the negative selection mechanism that enables detecting noisy data. However, AIS-Clus results are not constantly good since it is based on a set of arbitrary parameters. In [Zhu et al. 2019], a new approach for identifying hot terms was recommended in order to efficiently classify hot topics. First, the authors proposed a method to discover new terms in network news, that are lost from the word segmentation algorithm. Then, they used the time distribution information and the attention of users in order to propose a refined TF-IDF algorithm entitled TA TF-IDF. Finally, candidate hot news is classified according to the derived hot words and clustered into clusters representing hot topics. Despite the theoretical advantages of the methods mentioned above, most of them did not meet all of the requirements of data stream clustering algorithms: some researches did not handle concept drift, some had lacked in terms of accuracy and others were not concerned with the problem of the vocabulary evolution.

3 Background

Deep learning (DL) methods are described as representation learning methods with multiple levels of representations [Arnold et al. 2011]. Their relevant aspect is their ablity to learn compressed representations as intermediaries between input data and desired output which boost their performances. One of the most important requirements that motivates researchers to explore DL is its adequacy to learn complex functions from a very large set of examples. Several types of DL techniques can be distinguished namely the autoencoders. An autoencoder (AE) is a two-layer neural network with one hidden layer. The first layer along with the hidden layer form the encoder while the hidden layer and the last layer form the decoder. It is an unsupervised learning algorithm that applies backpropagation on a set of unlabeled training examples by setting the target value to be equal to the input. The AE is trained to encode the input x into a representation c(x), so that the input can be reconstructed from that representation. The main goal of the AE is to learn a representation for a set of data which is relevant to the tasks of data mining, typically, for dimension reduction [LeCun et al. 2015]. We present the principal idea of the AE as follows: Starting with random weights, the AE can be trained by minimizing the discrepancy between the original data and its reconstruction [Hinton et al. 2006]. It takes an input vector $x \in [0,1]^d$, and maps it to a hidden representation $y \in [0,1]^{d'}$ using the following function:

$$y = \sigma(W^{(1)}x + b^{(1)}) \quad (1)$$

Where σ is the activation function, W is a d'*d weight matrix and b is the encoding bias vector. The AE is fine-tuned using back-propagation of error-derivatives to adjust its weights and make its output as similar as possible to its input. Several types of AEs exist in the literature [Vincent et al. 2008] [Lucas et al. 2018]. One of the most famous is the Sparse AE [Andrew. 2011] . It has proved its suitability with sparse data like those presented in the social streams. These AEs discover interesting features by putting a sparsity constraint on the weights. This constraint allows only a small number of units to be activated. Given a sparsity parameter p and p_j the average activation of the hidden unit j, we have:

$$\hat{p}_j = \frac{1}{m}\sum_{i=1}^{m}[a_j^{(2)}(x^{(i)})] \tag{2}$$

Where m is the number of the training samples and $a_j^{(2)}(x^{(i)})$ is the activation of the hidden neuron j for the i^{th} training sample. The sparse AE works on enforcing the constraint $\hat{p}_j$=p. Activations of the hidden unit must be near 0 to satisfy this constraint. So, an extra penalty term will be added which is:

$$\sum_{j=1}^{s_2} p\log\frac{p}{\hat{p}_j} + (1-p)\log\frac{1-p}{1-\hat{p}_j} \tag{3}$$

Where s_2 is the number of neurons in the hidden layer. This penalty term can also be written as follow:

$$KL(p||\hat{p}_j) = p\log\frac{p}{\hat{p}_j} + (1-p)\log\frac{1-p}{1-\hat{p}_j} \tag{4}$$

Where KL is the Kullback-Leibler (KL) divergence between p and $\hat{p}_j$. Indeed, the training process of the sparse AE is distinguished from the simple AE by its objective to optimize the cost function which is :

$$cost = (\frac{1}{2n})\sum_{i=1}^{i=n}(x_i - \hat{x}_i)^2 + \beta\sum_{j=1}^{j=m} KL(p||\hat{p}_j) \tag{5}$$

Where n is the number of layers, m is the number of the training samples, x_i is the sample i, $\hat{x}_i$ is the constructed sample i and β is a parameter which indicates the sparsity penalty per layer. AEs, can be stacked to form the so called stacked autoencoder (SAE). This deep nature provides a high level of feature learning where each layer is trained separately with an AE. The SAE encodes the input data X and produces Y and then decodes Y and produces $\hat{X}$. The weights between the layers X and Y are tuned by the error e = $\hat{X}$-X. The networks continue to encode Y and produce Z and then to decode Z and produce $\hat{Y}$. The error e = $\hat{Y}$-Y is used later to tune the weights between layers Y and Z. These encoding and decoding operations will be repeated until it reaches the desired number of layers [Deng et al. 2010]. SAEs have showed highest performance on several problems [Kandaswamy et al. 2016] [Gehring et al. 2013].

4 Proposed method Stream-Clus

In this section, we present a detailed description of our proposed method called Stream-Clus (Stream Clustering) to extract hot topics from social streams and how deep learning techniques are exploited for clustering text streams. Data streams must be processed to extract the useful information instantly. Because of the high speed and the huge size of data that arrive continuously, our proposed clustering algorithm should be efficient to manage the large amount of data and adaptive to manage the unknown change in the distribution of data. Stream-Clus should also take into account the vocabulary evolution and handle the concept drift in real time. Our algorithm involves the training of a stacked sparse AE with evolving vectors of features (attributes) having unknown space, in order to reduce the data dimensionality and to improve the efficiency of the social streams clustering. The overall process of our algorithm is presented in figure 1. Stream-Clus is

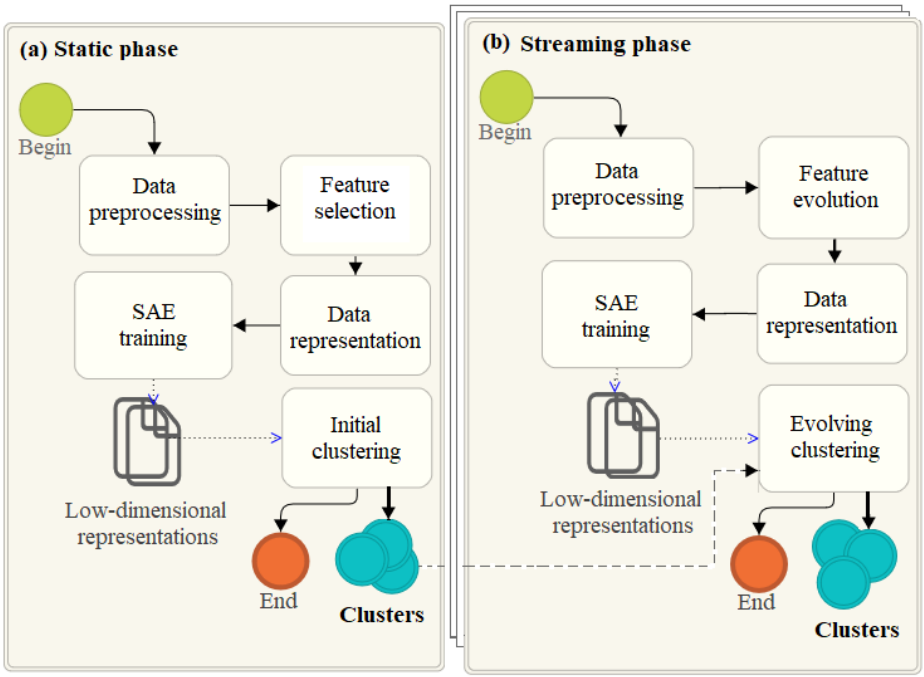


Figure 1: The overall workflow of our proposed method

composed of two phases:

The static phase: During this phase, historical data are clustered. Therefore, a step of preprocessing is achieved. Later, a representing and training step is managed. Then, the clustering algorithm Partitioning Around Medoids (PAM) [Kaufman et al. 1990] is used to obtain the initial clusters.

The streaming phase: It takes into account features selected and clusters obtained during the static phase. In this streaming phase, for each new arriving stream, the text is preprocessed, the features are evolved and the clusters are updated. Some clusters

disappear, others grow and new clusters appear.

4.1 Static phase

4.1.1 Text preprocessing

The aim of this step is to filter and reduce the huge size of textual data. Therefore, this step leads to more compact and structured data. For this purpose, all punctuation marks are removed as well as non-text-character and stop words that have no additional value. Also, non-English text is considered as irrelevant, and, URLs, reply and re-post syntaxes are removed.

4.1.2 Feature selection

The feature selection is one of the most interesting steps of our algorithm. It aims at identifying an optimal subset of interesting attributes to represent tweets. During this phase, a feature selection method is applied to select the best features. Thus, we filter the set of attributes to extract the discriminating and relevant information. Accordingly, the frequency of each word of the historical data is calculated. This is done by dividing the number of tweets where a word appears by the total number of tweets composing the static phase as follow:

$$Frequency(word) = \frac{NumberOfTweetsContainingWord}{TotalNumberOfTweets} \tag{6}$$

If the frequency of the word is greater than the user-defined threshold α, this word is considered as a relevant one, thus, as a feature. The threshold is fixed using a development set composed by a set of tweets. The relevant words will be considered as a first part of the initial feature vector. The second part will be presented as a set of empty cases. Its size is determined by the user. It is meant to take into account relevant words that can appear in the streaming phase. Otherwise, a feature vector is defined by the set of frequent words identified as features. An example of an initial feature vector is illustrated in figure 2.

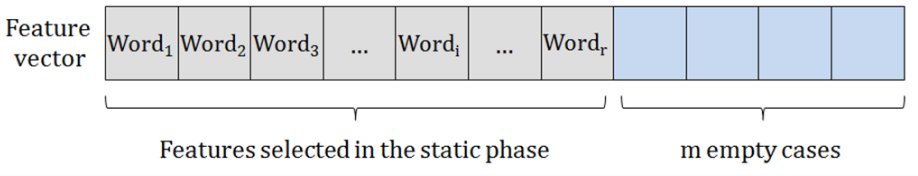


Figure 2: Initial feature vector

4.1.3 Data representation and dimension reduction

During this phase, our method takes, for all batches (also called streams), the same input vector size. Hence, the input vector size should be determined from the beginning. So,

after the selection of all features of the static phase, all tweets of this phase are represented by a binary vector that is composed of two parts. The first part is formed depending on the presence of each attribute, previously selected, in each tweet. If the attribute i is present in the tweet, the case i of the tweet vector representation is filled by one, else, it is filled by zero. The second part which corresponds to the empty part is filled with zeros. It is designed to take into account features that can be added in the following streams (see Fig.3). Then, a binary representative vector of the whole static phase is formed. It is prepared in the same manner than the vector of each tweet. Regarding the high dimension

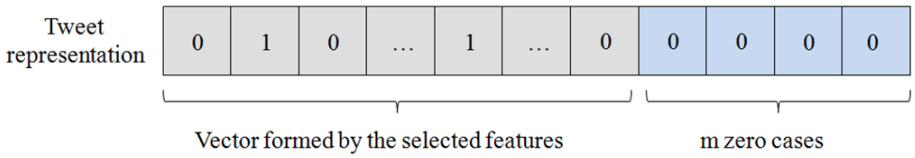


Figure 3: Example of a tweet vector in the static phase

of the data, a SAE is used to reduce dimension. It learns all the tweets vectors of the static phase and returns low-dimensional representations of the input vectors, which will be clustered later (see Fig.4). These representations are defined as reduced vectors presented with values that vary from zero to one. When the SAE finish learning, we introduce as input the representative vector of the static phase, and the learnt SAE will return a low-dimensional representation vector of the static phase. This compact representation aims at representing the whole static phase in a synthetic and global manner. Then, the obtained SAE will be used to represent the tweets of the following batches in order to take into account the temporal evolution of data streams and to integrate the historical data with the new streamings.

4.1.4 Initial clustering

Our algorithm consists, during the static phase, of clustering the low-dimensional representations of tweets obtained by the SAE. For this purpose, the representations of the data produced by the SAE are clustered using the (PAM) clustering algorithm [Kaufman et al. 1990]. To solve the problem of the choice of the number of clusters, we have proposed to reapply iteratively the PAM algorithm with k = k + 1 and to calculate the intra-similarity of the obtained clusters. Then, we compare this intra-similarity with a user defined parameter S that we have fixed following a set of experiments on a corpus of development. Finally, we decide for the optimal number of clusters. Thus, we obtain the initial clusters of tweets representing the initial topics of the historical data.

4.2 Streaming phase

4.2.1 Feature evolution

In the streaming phase, new relevant words appear with each stream, so new features could be added. Therefore, a feature evolution method is applied. Some features will be deleted and others will be added. For this purpose, we have performed the same

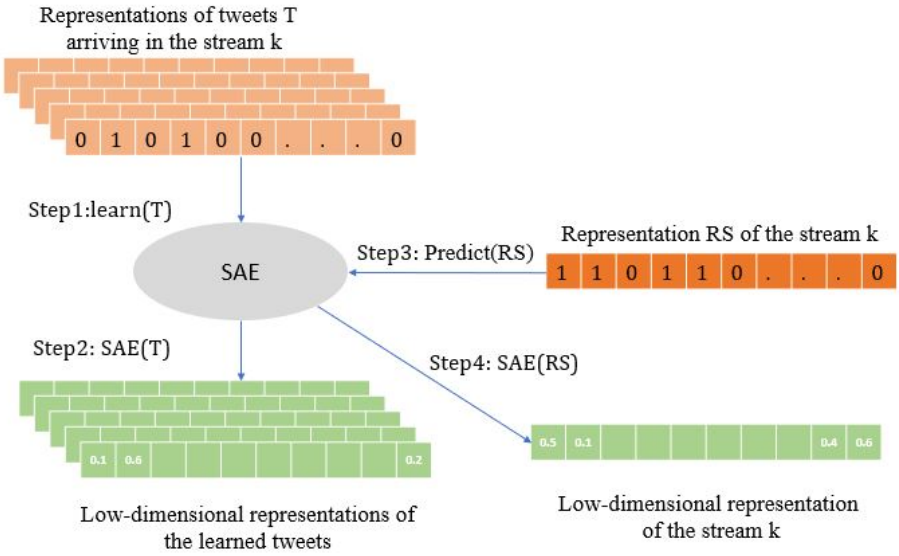
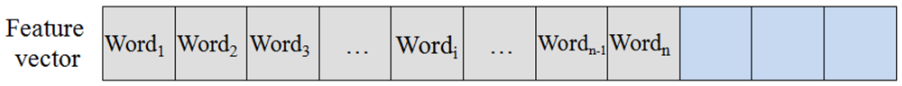
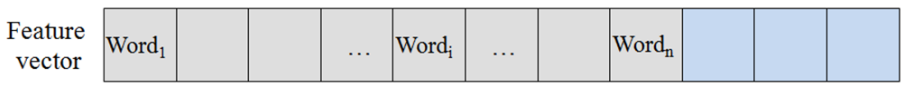


Figure 4: Process of the dimension reduction phase using the SAE model

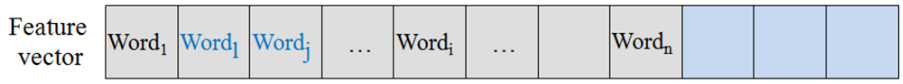
preprocessing step described in 4.1.1 for all the arriving textual data. Then, for each stream, the frequencies of all words are calculated (see Equation.6). If the frequency of a word is greater than the user-defined-threshold β, then, this word is considered as a feature. If a previous feature is missed in k successive streams (k is defined by the user using the development set), then, it is considered as no more relevant, and it should be removed from the feature vector. In fact, even if this word will return to be a feature during the following streams, it will be considered as a new feature that concerns a new topic. The new features take the places of those removed in the feature vector. For example, if the feature vector of the stream S is as follow: And if, during the S+k



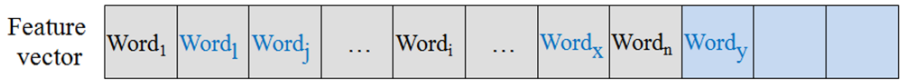
successive streams, the words 2, 3 and n-1 become no more relevant, they should be deleted. Thus, we obtain: We assume, however, that two relevant words have to be added



as new features: the word i and the word j. Hence, they will take the places of the deleted attributes 2 and 3, while the position n-1 remains empty. Thus, the feature vector of the stream S+k is as follow: Furthermore, if there are no more places, new features will



be added in the empty cases of the feature vector. For example, if during the stream, two relevant words appear, then, one relevant word ($Word_x$) will take the place of the remaining case (the place of word n-1 deleted previously). Moreover, we will consume one empty place for the second relevant word ($Word_y$) as follow: Actually, the number



of the frequent words that can be considered as features is limited. So, in some cases, some features will not be considered. This feature evolution process allows to take into consideration the temporal aspect of the streams and to adapt the existing feature to the new ones.

4.2.2 Data representation and dimension reduction

Our method consists of the tweets representation by three parts vectors corresponding to three dimensions: the past, the present and the future. The first part represents the past. It is defined as the low dimensional representation of the previous stream (or the static phase in the case of the first stream), produced by the SAE. This part allows representing all previous data in a synthetic manner. This compact representation is intelligent and incremental since it captures the relevant information extracted from all previous data. In fact, the integration of this part in the representation of the tweets of the current stream ensures a continuous link between the evolving data during the dynamic phase. The second part represents the present. It designs the representation of each tweet in a bag of words (BOW) manner. This part is the only real one which represents each arriving tweet depending on the features it contains. The third part designs the future. We have composed this part of a set of cases filled with zeros to take into consideration relevant words that could be added in the future. This overall representation of tweets of the current stream will be later integrated with the arriving data of the following streams throw the first part of their low dimensional representations. This recurrent process allows adapting current data to historical ones and cope with the evolving nature of the text stream data (see Fig.5). In a similar manner that in the static phase, the representative vector of each stream is prepared at this step. Indeed, at each stream, the SAE learns the

tweets vectors and produces corresponding semantic low-dimensional representations. The SAE produces also the compressed representation of the representative vector of the whole stream based on the model formed by the learned vectors of the tweets of the same stream. This low-dimensional representation of the stream will be used as a first part in the tweet vectors of the following stream in order to integrate all previous data with those who arrive continuously.

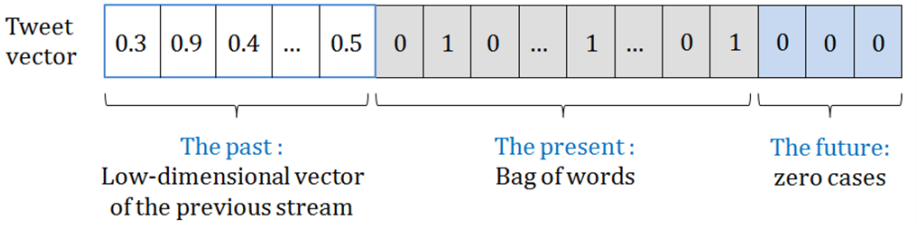


Figure 5: An example of a tweet representation in the dynamic phase

4.2.3 Evolving clustering

For each batch, we achieve the clustering of the low-dimensional representations of the tweets obtained by the SAE with the same method as in the static phase. In fact, a batch is defined as a stream during the streaming phase of our method. e notice that with the progress of streams, clusters (topics) will evolve. Some clusters will disappear, others will grow and new clusters will appear. So, a mapping step between clusterings at time t and t-1 is required to ensure the permanent tracking of the clusters evolution with the progress of streams over time. First, a fusion parameter is calculated between each cluster C, produced by the clustering at time t, of the current batch and every cluster G, produced by the clustering step at time t-1, of the previous batch. As every cluster is represented by the set of words contained in its tweets, then the fusion parameter FP can be calculated as follow:

$$FP(C,G) = \frac{NumberOfCommonWords(C,G)}{Min(TotalNumberWords(C), TotalNumberWords(G))} \quad (7)$$

The FP is calculated to conclude how much two topics from two following streams are similar as they share a set of the same words, and if they consequently should be merged. If the value of the FP between two clusters is greater than the user-defined-fusion threshold γ (fixed on a development set), then these two clusters are considered as similar. So, they are specific to the same topic. Thus, they will be merged together even if they are expressed by different words. . This is explained by the fact that even if the words used to describe a topic change, several other frequent words described the subject will be absolutely used in common. In some cases, a cluster G of the current stream can be merged with several clusters of the previous stream. In this case, the cluster G will be merged with the cluster that ensures the maximum value of the fusion parameter. In addition, a cluster that did not merged during the λ previous streams will be no more considered as a relevant one since it is representing an old topic which is

no more discussed. Therefore, it will disappear. Moreover, if a cluster appears with the current batch, and it cannot be merge with any cluster from those obtained in previous streams, then it will be identified as a new cluster representing a new topic (see Fig. 6). The final clusters obtained are the clusters that represent the current hot topics in social networks streams.

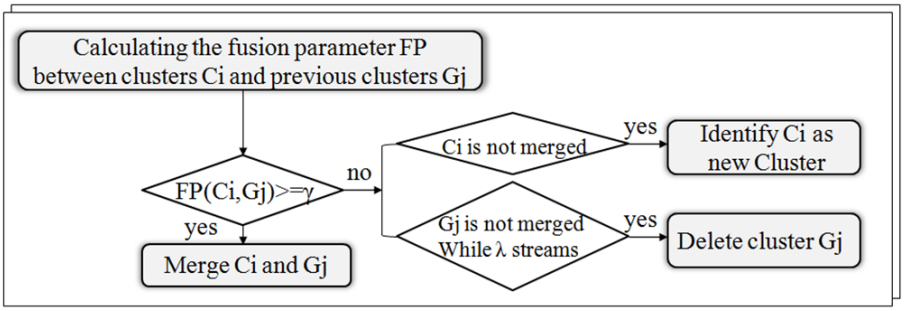


Figure 6: The mapping step in the streaming phase

5 Experimental evaluation

In this section, we discuss the experimental results of our proposed method. In order to evaluate the performance of our algorithm, we conducted several experiments based on the two datasets which have been widely used in the literature and will be firstly introduced: Sanders [$Saifet\ al.$2013] and Health Care Reform HCR [Speriosu et al. 2011]. Also, we constructed our personal dataset that we called it Terr-Attacks which we used to evaluate the performance of our method. In fact, the implementation of our method requires a set of development tools to automate its main tasks. For this purpose, we used the R language to efficiently develop our SAE based method. Therefore, a set of packages in R are used as the SAENET package [Stephen et al. 2015] to ensure the implementation of the SAE. To reach efficient clustering results, different parameters need to be adjusted in our algorithm according to our development set. We compare our results with three of the most relevant algorithms DenStream [Cao et al. 2006], Dstream [Chen et al. 2007] and CluStream [Aggarwal et al. 2003], which have given the best results in the context of the data streams clustering.

5.1 Test datasets

The datasets which are used in this evaluation step are divided into two phases, static and streaming. Sanders dataset consists of 5512 hand-classified tweets about four different topics collected between 18 October 2011 and 20 October 2011 [Saif et al. 2013] (see table1). After a selection step, only English tweets are considered which are 3065 tweets. The second dataset HCR is an annotated dataset composed of tweets dealing with the health care reform in the USA and containing the hashtag (# hcr) in March 2010 [Speriosu et al. 2011]. It consists of 2516 tweets with 8 targets (see table1). After the preprocessing

step, 780 tweets are considered. Moreover, we constructed our own dataset called Terr-Attacks using the twitteR package from the framework R [Gentry. 2016].We focused on five of the most dangerous terrorist attacks and we extracted tweets discussing them using a set of key words. Our dataset consists of 3200 tweets on 5 topics (see table 1). Statistics of the mentioned datasets are presented in table 2.

Dataset	**Phases**	**Number of tweets**	**Topics**	**Tweets (per Topic)**
Sanders	Static	500	Apple	260
			Google	240
	Streaming	2565	Apple	606
			Google	548
			Microsoft	797
			Twitter	614
HCR	Static	365	HCR	180
			Obama	150
	Streaming	450	Democrats	130
			Gop	120
			Conservatives	80
			Liberals	30
			Teaparty	20
			Hcr	70
Terr-Attacks	Static	500	TurkeyAttack	300
			GermanyAttack	200
	Streaming	2700	TurkeyAttack	400
			GermanyAttack	200
			RussiaAttack	500
			SwedenAttack	700
			LondonAttack	900

Table 1: Datasets' clusters distribution

5.2 Evaluation of the proposed method

5.2.1 Evaluation of the data representations

In order to evaluate our approach, we carry out a series of experiments where the first objective is to assess the performance of the SAE with our data. For this purpose, we should evaluate the efficiency and compactness of the low-dimensional representations produced by the SAE. Therefore, we use the SAE with seven hidden layers for the Sanders and HCR datasets, and with four hidden layers for the Terr-Attacks dataset. Also, we adjusted the number of iterations to the minimum necessary during the evolving phase. Thus, each AE is learnt with 3000 iterations except with the static phase where 4000 iterations are done in the learning step since the SAE has enough time to learn. All these parameters are fixed using a development set composed by subsets of our

	Sanders	HCR	Terr-Attacks
Tweet's size minimal	9	9	12
Tweet's size maximal	140	140	138
Tweet's size average	47	71	83
Word's number minimal per tweet	2	2	2
Word's number maximal per tweet	23	26	24
Word's number average per tweet	9	13	15

Table 2: Datasets statistics

datasets separately.Then, we follow the experimental protocol presented in Figure 7. The aim of this evaluation step is to compare results obtained with BOW vectors and SAE-reduced vector using the SVM classifier [Noble. 2006]. First, we apply the SVM method [Noble. 2006] to the binary vectors representing the preprocessed tweets of the static phase (classification1). Second, we train the SAE with the binary vectors representing the preprocessed tweets on a learning corpus (80% of the static phase) to obtain their low-dimensional representations. Then, the learned SAE predicts the low-dimensional representations of the preprocessed tweets composing the test corpus (20% of the static phase). This test corpus was composed by 20% of tweets of each cluster in order to prevent to have a predominant class in the 20% used for test. Actually, we have chosen this split (80/20) by referring to the Pareto principle [Sanders. 1987]. Later, we apply the SVM method to these low-dimensional representations produced by the SAE (classification 2) and we compare the results of the two performed classifications. This experimental

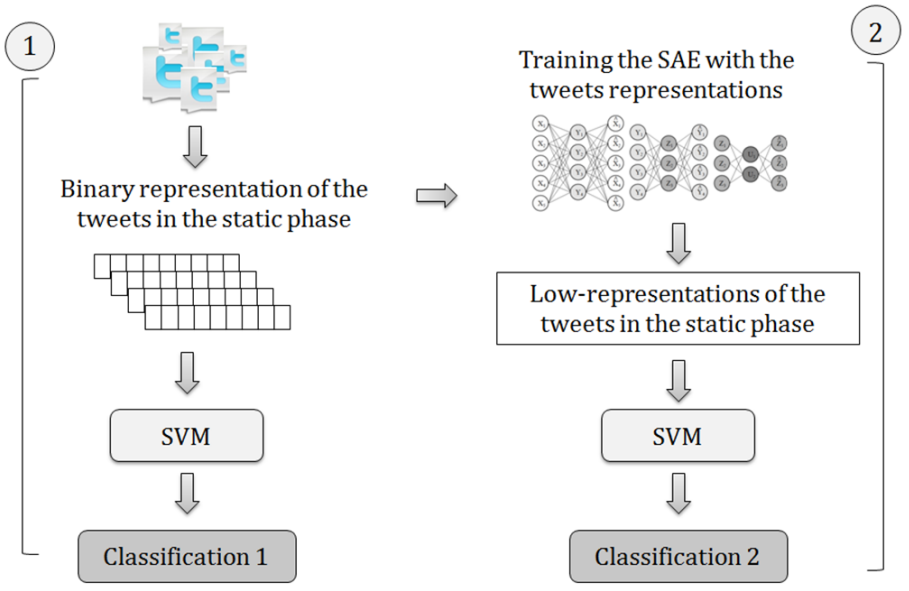


Figure 7: Experimental protocol of the SAE evaluation

protocol is followed mainly to estimate the capacity of the SAE of discovering interesting low-dimensional representations of the input vectors. The obtained results are presented in table 3. The SVM classifier is first applied on the feature vectors of the tweets during

Evaluation	Sanders	HCR	Terr-Attacks
SVM with binary representations	71%	60%	64%
SVM with SAE representations	91%	82%	88%

Table 3: Classification accuracies of the binary and low-dimensional representations using the SVM

the static phase. Second, the SVM is applied to the low-dimensional representation of the same tweets produced by the SAE. The results of the classification of the data by the SVM are improved remarkably with the use of the SAE. Thus, the SAE is well trained and produces interesting low-dimensional representations of the input vectors that will be clustered later.

5.2.2 Evaluation of the dynamic clustering

At this stage, we adjusted the size of the stream to 200 tweets for the Sanders and Terr-Attacks datasets and 100 tweets for the HCR dataset. The choice of the parameters is supported in figure 8 which presents the influence of streams size on the accuracy of our dynamic clustering for the Sanders, HCR and attacks datasets . The accuracies of our method vary each time where a new stream arrives. The obtained results are depicted in figure 9. For the Sanders dataset, the tweets of the static phase are labeled according to 2 topics. In the first stream a new topic appears (see Fig.9). The results are stable and bred. During the fourth stream, another new topic appears. The clustering results are still high, the same as all streams, which proves that the appearance of new topics do not disturb the clustering results of our method. For the HCR dataset, the static phase is composed by tweets about 2 topics. With the first stream, a new topic appears and detected efficiently by our method. During the third, fourth and fifth streams, new topic appears. The accuracy of our method is still high. Concerning the Terr-attacks dataset, the tweets composing the static phase are divided into 2 topics. New topics appear in the third, the fifth and the eighth streams. This apparition does not disturb the performance of clustering and the accuracy keeps its elevation which proves that our method can efficiently detect new clusters. So, the performance of our method is explained by: (1) the efficiency of the evolving representation of the tweets of each stream, and (2) the application of the SAE. However, the low fall of the accuracies during some streams can be explained by the heterogeneity of their data.

5.2.3 Evaluation of the mapping step

Actually, in order to perform scalable clustering, the mapping step between the groups of time t and the groups of time t-1 is required. In fact, the mapping is a phase which guarantees that each old stream's clusters are updated in accordance with the new clusters generated during the current stream. Thus, it allows for the continuous monitoring of the evolution of the clusters over time. In order to evaluate the mapping step, we have

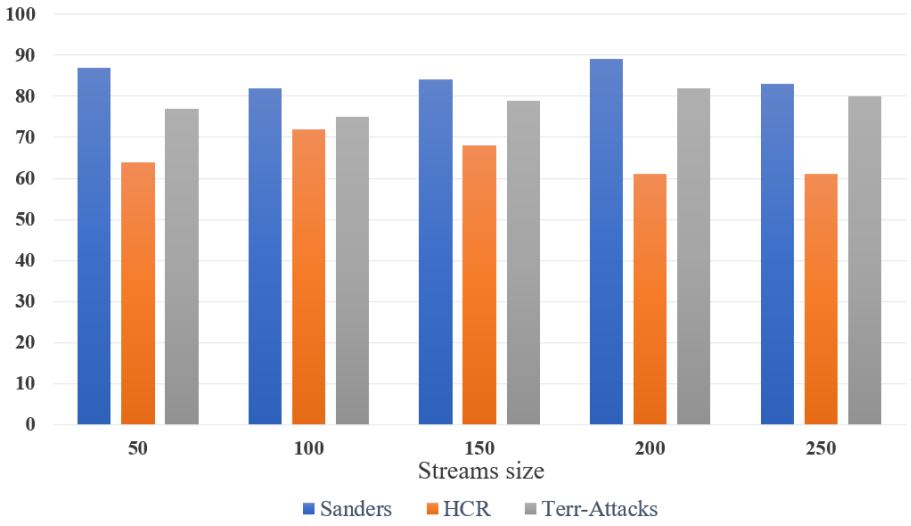


Figure 8: Streams size influence on the accuracy of our dynamic clustering

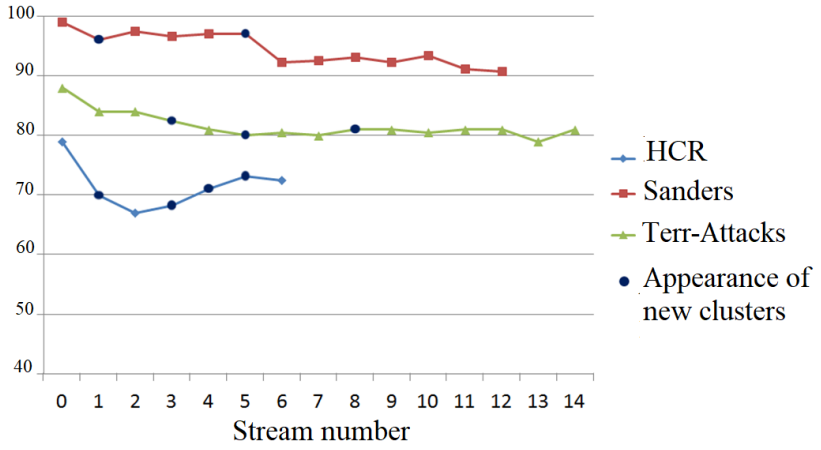


Figure 9: Accuracy evolution over streams of the three datasets

realized several experiments to choose the thresholds parameters of fusion and deletion over streams.Then, we compared the number of real clusters with the number of obtained clusters (see Fig.10). We can conclude that the real number of clusters in each stream is almost the same as the number of clusters that are produced by our method. This proves that the mapping step that we fulfilled to track the evolution of clusters is efficient and requisite. However, the difference, in some streams, between the real number of clusters and the produced ones by our method can be explained by the dependence of the results of the mapping step on the quality of the performed clustering in each stream in the

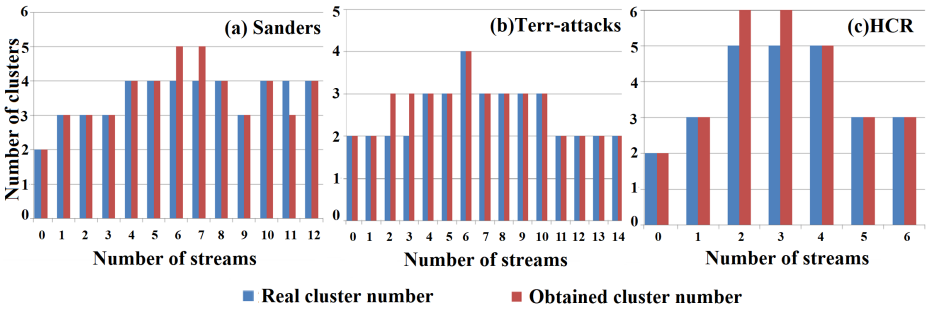


Figure 10: Real number of clusters VS produced number of clusters

Sanders and HCR datasets. Although, for Terr-Attacks dataset, the difference between the real number of clusters and the produced ones by our method can be explained by the heterogeneity of the vocabulary composing the same cluster between two streams. Moreover, when our method did not perform the necessary fusion of the old cluster and the new detected one during a stream, this result affects also the number of clusters in the following stream containing this old cluster. In addition, this new cluster that was not merged will be treated as an outlier and later will be deleted. However, in case our method produces a number of clusters lower than the real number of clusters, the results of the following streams will not be affected. In addition, we also used the measure of the Hamming distance to evaluate the difference between the real clusters and those obtained by our method in each stream [Hamady et al. 2008]. So, we presented the obtained clusters and the real clusters in each stream by a binary vector where each bit represents the presence/absence of each cluster. Then, we calculated the average Hamming distance $\overline{H}$ for all streams. Hence, we obtained $\overline{H}_{Sanders}$=0.23 , $\overline{H}_{HCR}$=0.28 and $\overline{H}_{Terr-Attacks}$=0.13. So, the produced clusters are so similar to the real clusters except in some streams with a slight difference. In fact, in some streams where our method produces more clusters than the real ones, these small additional clusters will be treated as outliers so that they will not affect the results of our method. Thus, the exploited process to track the clusters evolution over the streams allows to extract efficiently the real clusters.

5.3 Performance comparison

For comparison purposes, we used the CluStream, DenStream and Dstream algorithms , and we fixed their parameters by following a set of experiments. Then, we implemented these methods using the StreamMOA [Hahsler. 2020] package of the R language. The final results are tabulated in table 4. In this latter, the precision was obtained by counting the number of positive class predictions that were correct. However, we calculated the recall by counting how many positive class predictions were obtained from all positive examples in the dataset. The obtained results show the efficiency of our proposed system in retrieving hot topics from social streams, as compared to other algorithms in all datasets, in terms of recall, accuracy, F-measure and the number of clusters.

Datasets	Evaluation measures	CluStream	DenStream	Dstream	StreamClus
Sanders (4clusters)	Recall	0,967	0,753	0,694	0,87
	Precision	0,451	0,562	0,37	0,899
	F-measure	0,615	0,644	0,482	0,884
	Number of clusters	4	4	6	4
HCR (7clusters)	Recall	0,88	0,98	0,828	0,69
	Precision	0,35	0,33	0,508	0,759
	F-measure	0,5	0,5	0,63	0,723
	Number of clusters	7	1	2	7
Terr-Attacks (5clusters)	Recall	0,7	0,57	0,65	0,85
	Precision	0,63	0,54	0,52	0,79
	F-measure	0,66	0,55	0,58	0,82
	Number of clusters	5	8	4	5

Table 4: Performance comparison

6 Discussion

Our method is capable of dealing with the evolution of data streams that come continuously at a rapid rate. This is achieved through the mechanism we have suggested for the evolution of features, which is presented by the elimination of the no longer important features and the inclusion of new features that appear with each stream. This process ensures the efficient adaptation of new features with the old relevant ones. Moreover, our method allows interpreting tweets in a scalable manner in order to take into account the incremental character of the data streams. In addition, our approach relied mainly on the application of the SAE. The use of such technique led our algorithm to deal with the immense mass of data streams. The SAE used a series of hidden layers that reduced the representation dimension of the tweets without considering the zero positions and while preserving the most significant characteristics. As a matter of fact,exploring the representations of the tweets produced by the SAE helps our approach to effectively cluster the evolving data streams. Often, one of the benefits of our approach has been the exploration of the PAM method. PAM definitely allows us to obtain results that are not only efficient, but also stable since it does not rely on random values during its execution. Furthermore, our method succeeds in considering the evolutionary temporal aspect of the clusters produced at each stream. This is accomplished via the mapping phase between clusters created during consecutive streams. As it succeeded in following the evolution of the clusters over time, SAE-Clus is also able to detect the drift concept. The benefit of our method lies also in its speed. This is due the RStudio platform that we have used for learning the SAE. Indeed, the R language is known by a high speed, particularly with the DL algorithms. The speed of our method is also due to the adjustment of an optimal iterations number for training the SAE in order to ensure a good performance.However, we are conscious, of the method's limits when dealing with datasets that are completely unbalanced. So, we will strive to improve our method in the future to deal with this kind of datasets.

7 Conclusion and future work

In this paper, we presented a new method based on a DL architecture to extract relevant topics from social streams continuously. First, data are represented in an evolving manner. Then, a SAE is trained to produce low dimensional representation for the data which will be clustered in a scalable way to detect the real time evolution of topics. The experiments prove that our method extracts efficiently the hot topics and exceeds relevant data stream algorithms. Our main perspective will concern the improvement of our method to handle multilingual tweets. Also, we tend to focus on the detection of menacing tweets clusters.

References

[Abid et al. 2018] Abid, A., Jamoussi, S., Ben Hamadou, A.:"Handling concept drift and feature evolution in textual data stream using the artificial immune system"; Conf. Comp. Coll. Intell., Springer, UK (Sep 2018), 363–372.

[Adepu et al. 2019] Adepu, B., Gyani, J., Narsimha, G.:"An improved differential evolution algorithm for data stream clustering"; Journ. Elec. Comp. Engin., 9,4 (Aug 2019), 2659-2667.

[Aggarwal et al. 2003] Aggarwal, C. C, Han, J., Wang, J., Yu, P. S:"A framework for clustering evolving data streams";Proc. 29^{th} VLDB Conf. Very Large databa., VLDB, Germany (Sep 2003), 81-92.

[Andrew. 2011] Andrew, N.:"Sparse autoencoder"; Journ. Lect. not., 72, 2011, Germany (Sep 2011), 1-19.

[Arnold et al. 2011] Arnold, L., Rebecchi, S., and Chevallier, S., and Paugam-Moisy, H.:"An Introduction to Deep Learning"; Proc. Europ. Symp. Artif. Neur. Netw. Comp. Intell. Mach. Learn., Belgium (Apr 2011), 477-488.

[Cao et al. 2006] Cao, F., Estert, M., Qian, W., Zhou, A.:"Density-based clustering over an evolving data stream with noise"; Proc. SIAM Conf. data mining, SIAM,USA (Apr 2006), 328-339.

[Chen et al. 2007] Chen, Y., Tu, L.:"Density-based clustering for real-time stream data";Proc. 13^{th} ACM Conf. kno. dis. dat. , ACM, USA (Aug 2017), 133-142.

[Deng et al. 2010] Deng, L., Seltzer, M. L, Yu, D., Acero, A., Mohamed, A., Hinton, G.:"Binary Coding of Speech Spectrograms Using a Deep Auto-encoder"; Conf. Inter. Spe. Com. Ass., Japan (Sep 2010).

[Gehring et al. 2013] Gehring, J., Miao, Y., Metze, F., Waibel, A.:"Extracting deep bottleneck features using stacked auto-encoders"; Conf. Acou. Spee. Sign. Process., IEEE, Canada (May 2013), 3377-3381.

[Gentry. 2016] Gentry, J,:"Package 'twitteR'"; 2015(Jul) https://www.rdocumentation.org/packages/twitteR/versions/1.1.9

[Hahsler. 2020] Hahsler, M., Bolanos, M., Forrest, J.:"Package 'streamMOA'"; 2020(Dec) https://www.rdocumentation.org/packages/streamMOA/versions/1

[Hamady et al. 2008] Hamady, M., Walker, J. J, Harris, K., Gold, N. J, Knight, R.:"Error-correcting barcoded primers for pyrosequencing hundreds of samples in multiplex"; Journ. Nat. Meth., NPG 5, 3 (Dec 2008), 235–237.

[Hinton et al. 2006] Hinton, G. E, Salakhutdinov, R. R:"Reducing the dimensionality of data with neural networks"; Journ. Scien., AAAS 313, 5786 (Nov 2006), 504-507.

[Kandaswamy et al. 2016] Deng, L., Seltzer, M. L, Yu, D., Acero, A., Mohamed, A., Hinton, G.:"High-content analysis of breast cancer using single-cell deep transfer learning"; Journ. biomo. screen., CA 21,3 (May 2016), 252-259.

[Kaufman et al. 1990] Kaufman, L., Rousseeuw, P. J:"Partitioning around medoids (program pam)"; Journ. Find. Grou. Dat. Clust. Analy., WOL, (Mar 1990), 68-125.

[LeCun et al. 2015] LeCun, Y., Bengio, Y., Hinton, G.:"Deep learning"; Journal Nature, Nature Research 521, 7553 (Dec 2015), 436-444.

[Lucas et al. 2018] Lucas, T., Jakob, V. :"Auxiliary guided autoregressive variational autoencoders"; Conf. Ma. Lear. Kn. Dis. Dat., Springer, Irland (Sep 2018), 443-458.

[Noble. 2006] Noble, W. S:"What is a support vector machine?"; Journ. Nat. Biotech., 24, 12 (Dec 2006), 1565-1567.

[Rekik et al. 2016] Rekik, A., Jamoussi, S.:" Deep learning for hot topic extraction from social streams"; Proc. 16^{th} Conf. Hyb. Int. Sys., Springer,Morocco (Nov 2016), 186-197.

[Saif et al. 2013] Saif, H., Fernandez, M., He, Y., Alani, H.:"Evaluation datasets for twitter sentiment analysis : a survey and a new dataset, the STS-Gold"; Proc. 1^{st} Inter. Work. Emot. Sent. Soc. Express. Med. : Appro. Perspec. AI, Italy (Dec 2013).

[Sanders. 1987] Sanders, R.: "The Pareto principle: its use and abuse"; Journ. of Serv. Market.,1,2 (1987), 37.

[Speriosu et al. 2011] Speriosu, M., Sudan, N., Upadhyay, S., Baldridge, J.:"Twitter polarity classification with label propagation over lexical links and the follower graph"; Proc. 1^{st} Work. Unsuper. Learn. NLP, ACL, Scotland (July 2011), 53-63.

[Stephen et al. 2015] Stephen, H., Eugene, D.:"Package 'SAENET'"; 2015(Jun) https://www.rdocumentation.org/packages/SAENET/versions/1.1

[Tareq et al. 2020] Tareq, M., Sundararajan, E. A, Mohd, M., Sani, N.S.:"Online Clustering of Evolving Data Streams Using a Density Grid-Based Method"; Journ. IEEE Acc., IEEE 8 (Sep 2020), 166472-16649.

[Vincent et al. 2008] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.:"Extracting and composing robust features with denoising autoencoders"; Proc. 25^{th} ACM Conf. Mach. Learn., ACM, USA (Jul 2008), 1096-1103.

[Zhu et al. 2019] Zhu, Z., Liang, J., Li, D., Yu, H., Liu, G.:"Hot topic detection based on a refined TF-IDF algorithm"; Journ. IEEE Acc., 7 (Sep 2019), 26996-27007.