High Dimensional Data Stream Clustering using Topological Representation Learning

Maha Ben-Fares, Nistor Grozavu, Parisa Rastin, Pierre Holat

High Dimensional Data Stream Clustering using Topological Representation Learning

Maha Ben-Fares[1,3], Nistor Grozavu[1], Parisa Rastin[2] and Pierre Holat[3,4]
ETIS, CY Cergy Paris Université - Pontoise, France[1]; LORIA, Université de Lorraine - Nancy, France[2]
FI Group - Puteaux, France[3]; LIPN, Université Sorbonne Paris Nord - Villetaneuse, France[4]
E-mail: maha.ben-fares@cyu.fr, nistor.grozavu@cyu.fr, parisa.rastin@loria.fr, pierre.holat@fi-group.com

***Abstract*—Due to the high dimensionality of the data, storing the whole set of data during stream processing is impractical. Therefore, only a summary of the input stream is maintained, necessitating the development of specialized data structures that permit incremental summarization of the input stream. The problem becomes more complex when dealing with high-dimensional text data due to the high sparsity. In this paper we propose a new topological unsupervised learning approach for high dimensional text data streams. The proposed method simultaneously learns the representation of the stream and cluster the data in a smaller dimension space. The evaluation of the proposed OTTC (Online Topological Text Clustering) approach and the comparison with the state of art methods is done by using the framework MOA (Massive Online Analysis), an open-source benchmarking software for evolving data streams. The proposed approach outperforms the classical methods and the obtained results are very promising for clustering high dimensional text data streams.**

***Index Terms*—Data stream, Clustering, high dimensional data.**

I. Introduction

In the last few years and due to technological breakthroughs, large amounts of continuous data flow are generated every day and are referred to as data streams. Real-time processing of these data requires a large amount of memory and presents a significant challenge compared to static data due to the many constraints that must be considered:

- One-time pass: each data object can be read just once.
- Unlimited data: the amount of arriving data is extremely large to be stored.
- Evolving data over time: the capacity to discern new data probability distributions over time is known as concept drift.

Recently, a lot of research [1] [2] [3] has been conducted in the field of data stream clustering due to the large number of its applications in different domains. One of the tasks on data streams is to detect their structure using clustering approaches. The goal of clustering methods is to group similar data objects into groups called clusters. However, the nature of stream data requires the development of new methods capable of performing an incremental clustering of data while taking into consideration the constraints mentioned above. The problem of data stream clustering was mostly done with numeric data streams. There hasn't been much research done on data stream clustering with large textual data because of the many problems it poses, such as its high dimensionality, which can't be solved with the traditional methods of data stream clustering.

In this paper we propose a new method that combines data projection model and online clustering in order to continuously cluster high-dimensional streaming text data. The proposed method was evaluated using real data sets, it achieves better clustering quality in comparison with the traditional stream clustering methods.

Because of its volume, data stream processing cannot store all incoming data. Only a summary of the input stream is retained, therefore specific data structures are needed to gradually summarize it. We distinguish the following most used data structures [4] [5] [6]:

- Feature vectors: keep the summary of the data instances.
- Prototype arrays: only some representative instances that summarize the data is kept.
- coreset trees: a summary of the data is kept in a tree structure.
- grids: keep the data density in the feature space.

In order to process only a part of data stream, the time-window process usually is applied. The use of window models aims to reflect which part of the stream history is important, i.e. the recent history of the stream. There are several types of time-windows [4]:

- **Damped window model**: a weight is assigned to each object that arrives and this weight decreases over time according to some decay functions. One of such functions is $f(t) = 2^{-\lambda t}$, where t is the time passed and λ is the decay rate (Fig 1.a).
- **Sliding window model**: which considers the most recent observations in the stream. The window swaps one instance at each step. The older instance moves out of the window, and the most recent instance moves into the window by FIFO style. All instances within the active sliding window have equal weight and consecutive windows mostly overlap (Fig 1.b).
- **Landmark window model**: considers the data in the data stream from the beginning until the current time instant. All the instances have the same weight. It doesn't differentiate between past and present data which can affects the results of the model in view of the nature of

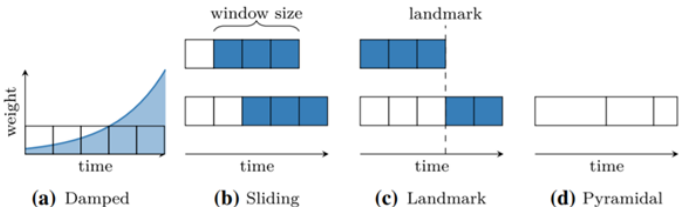

Fig. 1. Time window models [8]

the data stream which evolves continuously. (Fig 1.c).

- **Tilted window model** or the pyramidal time model [7]: it keeps summaries at different levels of granularity in a particular moment and based on the recency of data. The granularity level of weights gradually decreases as data points get older (Fig 1.d).

All these types of time-windows can be used for clustering data streams, but the sliding window model is well adapted in order to keep the topology of the stream. This type of window will be used as an online process of the stream in the proposed OTTC model.

The rest of this paper is organized as follows: after we introduced the basic concepts on data stream clustering in Section I, we introduce the related work on text clustering in Section II. The proposed OTTC approach is described in Section III and the obtained results and comparison with state of art methods in Section IV. Finally, the paper ends with a conclusion and some future work for the proposed method.

II. Related work

A. Data stream clustering algorithms

According to the state of the art and some survey papers on data stream clustering [5] [8] [9], data stream clustering algorithms can be divided into 3 main categories: Partitioning-based methods that divide the data into a number of pre-defined partitions, where each partition represents a cluster and where each cluster is formed based on similarity or distance to the cluster centroids. These methods are known for their simplicity, their easy implementation and they are very suitable for datasets with well separated spherical clusters. There are several partitioning based algorithms proposed in the literature, the most popular are STREAM [10], Clustream [7], StreamKM++ [11] and SWClustering [12]. The second category is Density-based methods that regroup the data into clusters based on the dense areas that are separated by sparse areas like DenStream algorithm [13], OPTICS-Stream [14] and D-Stream [15]. These methods have the ability to handle noise and detect arbitrary shape clusters but they are sensitive to the setting of input parameters. Finally, the third category of data stream clustering algorithms is model-based methods which assume that data are generated by a mixture of probability distributions and each component of the mixture represents a cluster.

The method that we propose in this work is partitioning-based method and it is based on one of the most well-known algorithms in the field of data stream clustering called Clustream proposed by Aggarwal et al [7].
Clustream is a framework that allows to perform clustering of data streams at different time horizons. The clustering is divided into two steps: online micro-clustering and offline macro-clustering. In online micro-clustering, we summarize the stream and we store statistical information about the data using micro-clusters, where a micro-cluster is a temporal extension of the cluster feature vector proposed earlier in BIRCH clustering method [16]. This kind of data structure allows to easily compute some basic cluster measures as cluster centroid and radius that we need in the phase of clustering. A micro cluster is defined as follows:

For a set of d-dimensional points $X_1, ..., X_n$: $X_i = (X_i^1, \ldots, X_i^d)$ with time stamps $T_1, ..., T_n$:

$$MC = (\overrightarrow{CF1}^d, \overrightarrow{CF2}^d, CF1^t, CF2^t, N) \quad (1)$$

where N is the number of d-dimensional data points in cluster, $\overrightarrow{CF1}^d$ and $\overrightarrow{CF1}^d$ are the linear and squared sum of the N data points in cluster and $CF1^t$ and $CF2^t$ are the linear and squared sum of all timestamps of a cluster.

$$\overrightarrow{CF1}^d = \sum_{i=1}^{N} X_i \qquad \overrightarrow{CF2}^d = \sum_{i=1}^{N} X_i^2 \quad (2)$$

$$CF1^t = \sum_{i=1}^{N} T_i \qquad CF2^t = \sum_{i=1}^{N} T_i^2 \quad (3)$$

In the offline macro-clustering, we obtain the macro-clusters by applying the k-means algorithm over the micro-cluster summaries stored before in conjunction with other defined parameters (such as time horizon and number of clusters).
In the proposed OTTC approach, we will use the Clustream algorithm as an online step to incrementally cluster high dimensional text data. Before using this online step, our proposed method will use two embedding methods in order to learn the representation of the stream without losing the structure knowledge of the data.

B. Representation learning

In order to learn the representation space of textual data there are several embedding methods, as classical vector representation using TF-IDF or neural networks based models as BERT.

TF-IDF: is a traditional NLP approach for text representation that compute the importance of a word in a document in a collection of documents D. It consists of two terms: Term Frequency (TF) [17] that reflects the frequency of a term t_i in a document d_j, and Inverse Document Frequency (IDF) [18] which provides a weight to each term t_i according to its frequency in the corpus D.
The TF-IDF of a term t in a given document d is defined as follows:

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t, D) \quad (4)$$

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \qquad idf(t,D) = \log \frac{N}{|\{d \in D, t \in d\}|}$$

where $f_{t,d}$ is the frequency of term t in document d, $\sum_{t' \in d} f_{t',d}$ is the total number of terms in document d, N is total number of documents and the denominator of the IDF term represents the number of documents containing the term t.

The terms with a high score of TF-IDF are the most relevant terms.

TF-IDF is a method that is simple to calculate and easy to use, it is used in several machine learning applications, but it has some drawbacks like the curse of dimensionality since the size of TF-IDF vectors is equal to the corpus vocabulary size, also it cannot capture the context and the semantic meaning of words which makes it a bad choice to represent textual documents for clustering. Thus, we need an advanced technique that can convert text to numerical vectors while capturing the important semantic information such as BERT.

***SVD**:* Before clustering the dataset, the single value decomposition can be used to learn the representations of the data and to deal with the sparsity of the stream. The singular value decomposition (SVD) is a matrix factorization. For example, the membership matrix M is an $|\mathcal{V}| \times |S|$ matrix, then we may write M as a product of three factors:

$$M = U \Sigma V^* \ , \tag{5}$$

where U is an orthogonal $|\mathcal{V}| \times |\mathcal{V}|$ matrix, V is an orthogonal $|\mathcal{V}| \times |S|$ matrix, V^* is the transpose of V, and Σ is an $|\mathcal{V}| \times |S|$ matrix that has all zeros except for its diagonal entries, which are nonnegative real numbers. If σ_{ij} is the i, j entry of Σ, then $\sigma_{ij} = 0$ unless $i = j$ and $\sigma_{ii} = \sigma_i \geq 0$. The σ_i are the "singular values" and the columns of u and v are respectively the right and left singular vectors. Then use the top k eigenvectors of M (resp. M') corresponding to the k smallest eigenvalues as the low dimensional representations of membership matrix (resp. nodes attribute similarity matrix).

Stochastic Neighbor Embedding (SNE) [19]: Let $X = \{x_1, \ldots, x_N\}$ be a N samples data set in a high dimensional space endowed with some distance $d(\cdot, \cdot)$. In general, in dimension reduction methods for data-visualization, the goal is to map the high dimensional dataset X to a low (usually 2 or 3) dimensional sample $Y = \{y_1, \ldots, y_N\}$ while preserving the topological relationship among X. The starting point in Stochastic Neighbor Embedding (SNE) is to convert the available distance $d(x_i, x_j)$ in the considered high dimensional space (e.g. the Euclidean distance $\|x_i - x_j\|$) to some probability $p_{j|i}$ that represents similarity of x_j to x_i. To compute this probability/similarity we use a normalized Gaussian kernel as follows:

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum\limits_{k=1, k \neq i}^{N} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}, \text{ and } p_{i|i} = 0, \tag{6}$$

where σ_i^2 is the variance of the Gaussian kernel. Note that the variance σ_i^2 is dependent on the position of the x_i sample.

In order to determine the value of σ_i, the perplexity parameter $2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$ is used, which is exactly equal to some user-defined value u. This computation could be done even by a binary search [19] or robust root-finding method [20].

In the same manner, in the low-dimensional space, the probability $q_{j|i}$ from the Euclidean distance $\|y_i - y_j\|$ by using normalized Gaussian kernel is computed as follows:

$$q_{j|i} = \frac{\exp\left(-\frac{\|y_i - y_j\|^2}{2}\right)}{\sum\limits_{k=1, k \neq i}^{N} \exp\left(-\frac{\|y_i - y_k\|^2}{2}\right)}, \text{ and } q_{i|i} = 0. \tag{7}$$

Note that in the low dimensional space we don't have an adaptive kernel with respect to the variance. To determine the coordinates of y_i we use the distributions P_i and Q_i, corresponding to the probability distribution in the higher and lower dimensional space respectively. Therefore the coordinates of y_i are obtained by the minimization of the KL divergence between the distribution P_i and Q_i :

$$C_{SNE}(Y) = \sum_{i=1}^{N} KL(P_i \| Q_i) = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \tag{8}$$

For the minimization of functional in Equation (8) the following gradient is used:

$$\frac{\partial C_{SNE}(Y)}{\partial y_i} = 2 \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j). \tag{9}$$

Let us remark that in the low dimensional space $q_{j|i} \neq q_{i|j}$ while in the high dimensional space $p_{j|i} \neq p_{i|j}$, therefore these probabilities are not symmetric. Another point to remark is the crowding problem, i.e. even for small values of $p_{j|i}$ and $p_{i|j}$ we want that the data points y_i and y_j in the low dimensional space are well-separated. While some strategies to compensate the crowding problem in SNE was proposed [19], this drawback motivates the definition of the t-SNE approach.

t-Distributed Stochastic Neighbor Embedding (t-SNE) [21]: In this section we present the t-Distributed Stochastic Neighbor Embedding (t-SNE) having two main differences compared to classical SNE. Firstly, the t-SNE uses symmetrical probability $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$, which allow that the resulting gradient of the cost function is simpler and easier to be optimized.

Secondly, in t-SNE, to define the similarity between the data points y_i and y_j in the low dimensional, the probability q_{ij} doesn't use a Gausssian kernel, but a t-distribution kernel with one degree of freedom, resulting in the following expression:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k,l(k \neq l)} (1 + \|y_k - y_l\|^2)^{-1}}, \text{ and } q_{i|i} = 0. \tag{10}$$

With this definition the cost function in the t-SNE writes as follows:

$$C_{tSNE}(Y) = KL(P\|Q) = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \quad (11)$$

While its gradient is given by:

$$\frac{\partial C_{tSNE}(Y)}{\partial y_i} = 4 \sum_{j\neq i} (p_{j|i} - q_{j|i})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j) \quad (12)$$

Recall that the t-distribution has a far more complex history than the Gaussian distribution. To simulate a tiny value of p_{ij} using the t-distribution, the distance between y_i and y_j must be considerable. This is the mechanism in t-SNE that compensates for the crowding issue caused by the fact that neighboring points in low-dimensional space occupy less space than in high-dimensional space.

III. Online Topological Text Clustering: OTTC

In this section we describe the proposed Online Topological Text Clustering (OTTC) approach which consists of three embedding steps and an online incremental clustering. The embedding steps consist of one textual embedding and two topological representation learning. For the textual embedding, the BERT method is used, and for the topological representation learning, the UMAP is used.

***BERT**:* is a transformer based language model developed by Google [22] that learns contextual embeddings of words in sentences using the attention mechanism. A word can have several representations depending on its context in the sentence. The model was pre-trained on a huge amount of textual data and it uses two training strategies: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). MLM consists in predicting the identity of a random masked input token using its context only while NSP predicts if two sentences are adjacent or not. BERT has achieved state of the art results on a wide range of NLP tasks such as text classification, question answering and Information Extraction. In our experiments, we used the sentence-transformers model, precisely all-MiniLM-L6-v2 pre-trained model, to convert our textual data to a 384 dimensional dense vector space.

***UMAP**:* Let $Y = \{Y_1, \ldots, Y_N\} \in \mathbb{R}^d$ be a low dimensional $(d << n)$ representation of X such that Y_i represents the source data point X_i. It is necessary to have a comparison method in place given the fuzzy simplicial set representations of both X and Y. If we simply take into account the 1-skeleton of the fuzzy simplicial sets, we will be able to represent each one as a fuzzy graph, or more precisely, as a fuzzy set of edges. In order to evaluate the similarity of two fuzzy sets, we will be using fuzzy set cross entropy. In light of these considerations, we shall return to the traditional fuzzy set notation. To put it another way, a fuzzy set may be defined as the product of a reference set A and a membership strength function $\mu : A \to [0, 1]$. The same reference set is used to compare fuzzy sets. We can go from a sheaf representation P to classical fuzzy sets by setting $A = a \in (0, 1]P([0, a))$ and $\mu(x) = sup\{a \in (0, 1]|x \in P([0, a))\}$.
The cross entropy C of two fuzzy sets (A, μ) and (A, v) is defined as :

$$C((A, \mu), (A, v)) = \sum_{a\in A} \left(\mu(a) \log \left(\frac{\mu(a)}{v(a)} \right) + (1 - \mu(a)) \log \left(\frac{1 - \mu(a)}{1 - v(a)} \right) \right) \quad (13)$$

By using stochastic gradient descent in a manner similar to that of t-SNE, we are able to maximize the embedding Y with regard to the fuzzy set cross entropy C. However, in order to do this, a differentiable fuzzy singular set functor is required. If the expected minimum distance between points is zero, then the fuzzy singular set functor can be used for these purposes without needing to make a differentiable approximation. However, if the expected minimum distance between points is not zero, then we will need to make a differentiable approximation (chosen from a suitable family of differentiable functions). This brings the process to its conclusion: by making use of manifold approximation and piecing together local fuzzy simplicial set representations, we are able to generate a topological representation of the high-dimensional data. Then, in order to reduce the amount of difference in accuracy between the two topological representations, we optimize the data arrangement inside a low-dimensional space.

The proposed approach is described in the Algorithm 1.

TABLE I
DATASETS DESCRIPTION

Datasets	#documents	#classes
Newsgroup20	18 000	20
AG News	127 600	4
BBC News	2225	5

IV. Experimental results

A. Datasets and test setup

We evaluate the proposed OTTC method on 3 public textual datasets namely Newsgroup20 [23], AGNews [24] and BBC_News [25] with varying numbers of documents and different numbers of classes. The summary of datasets is given in Table I.

Newsgroup: The 20 newsgroups dataset comprises around 18000 newsgroups posts on 20 topics split in two subsets: one for training and the other one for testing. The split between the train and test set is based upon messages posted before and after a specific date.

AG News: is a subdataset of AG's corpus of news articles constructed by assembling titles and description fields of articles from the 4 largest classes of AG's Corpus. It contains 127600 documents.

BBC_News: the dataset consists of 2225 documents from the BBC News website corresponding to stories in five news areas

Algorithm 1 Online Topological Text Clustering: OTTC

Input:
- Set of documents $D = \{d_1, \ldots, d_N\}$; k: number of macro-clusters; q: number of micro-clusters.
- maxkernels: maximum number of micro-clusters; d: the dimension of the target reduced space.
- n: the number of neighborhood to use for local metric approximation.
- n-epochs: is the number of iterations; h: the length of time window.
- min-dist: the minimum distance for the points to be together in the low-dimensional representation.

Output: Set of k clusters.

Contextual embedding:
for all $d_i \in D$ **do**
Generate document embedding e_i by using sentence-transformers of Bert model where $e_i \in E$.
end for
# Construct the relevant weighted graph
for $e \in E$ **do**
fs-set$[x] \leftarrow LocalFuzzySimplicialSet(E, e, n)$
end for
top-rep $\leftarrow \bigcup_{e \in E}$ fs-set$[e]$
Perform optimization of the graph layout
$S \leftarrow SpectralEmbedding(top\text{-}rep, d)$
$S \leftarrow OptimizeEmbedding(top\text{-}rep, S,$ min-dist, n-epochs$)$
Initialization:
- Create the first q micro-clusters by applying k-means on the first data that arrives from the stream S.
Online: Micro-cluster maintenance as a new point p arrives, where p ∈ S.
- Compute the distance between the point p and each of the q micro-cluster centroids.
- clu ← the closest micro-cluster to p.
- Calculate the max boundary of clu.
if p is falling within the max-boundary of clu **then**
- p is assigned to clu
- Update the micro-cluster clu by using the incrementality property.
else
- A new micro-cluster will be created using p.
- Create memory space for the new micro-Cluster.
if its safe **then**
- We delete the most old micro-cluster.
else
- We merge the two closest micro-clusters.
end if
end if
Offline: Macro-clustering
- Apply k-means over the active micro-clusters during h to get the k macro-clusters.

from 2004-2005.
All the results were obtained by running the experiments on a PC HP EliteBook 840 G5 with Intel Core i5-8250U CPU, 1.60GHZ and 8Go RAM.
The implementation of Clustream was obtained from framework MOA (Massive Online Analysis) [26], an open-source benchmarking software for data stream mining which is written in JAVA. It contains several machine learning algorithms and evaluation measures for running experiments.

B. Evaluation measures

To evaluate the clustering results, we used two evaluation metrics: purity [27] and silhouette score [28].

Purity is an external evaluation criterion of cluster quality, it compares the clustering results to the true partitions and represents the percent of data points that were correctly classified. It is defined as follows:

$$purity(\Omega, C) = \frac{1}{N} \sum_{k} \max_{c \in C} |\omega_k \cap c| \tag{14}$$

where Ω is the set of resulting clusters, C is the set of true clusters, N is number of objects and k is number of clusters. A high degree of purity indicates a successful clustering process. The purity of a cluster is defined as the proportion of data in the majority class.
The Silhouette index estimates the average silhouette width

for each sample, cluster, and overall data. Using this method, each partition may be evaluated by a silhouette depending on tightness and separation. The average Silhouette can be used to quantify the quality of the obtained clustering result. This index can also be an indicator for the number of clusters.

The silhouettes $S(i)$ are created using the following expression:

$$S(i) = \frac{(b(i) - a(i))}{max\ \{a(i), b(i)\}} \tag{15}$$

where $a(i)$ represents the average dissimilarity of the i^{th} item to all others in the same cluster, and $b(i)$ represents the average dissimilarity of the i^{th} object to all other clusters (in the closest cluster).

The Silhouette index states for $-1 \leq s(i) \leq 1$. If the silhouette is close to 1, the corresponding data belongs to a good cluster (well clustered). If the silhouette is close to zero, the sample is equally far from both closest clusters. If the silhouette value is close to -1, the sample is misclassified and is between the clusters. An object's $S(i)$ is the average silhouette width for the whole dataset. The ideal number of clusters is the one with the highest overall average silhouette width.

C. Results

In the first set of experiments, we prove that the choice of text embedding method to convert the textual data into numeric vectors can play an important role on the quality and the results of clustering. To this end, we tested two different approaches to represent the text documents. First we applied TF-IDF model, one of the most frequently used method that reflects the importance of a word in a document from a corpus, then BERT (Bidirectional Encoder Representations from Transformers), a transformers based model which uses pre-trained model to represents text data into fixed feature vectors. We calculate the purity and silhouette measures for the two methods on the three datasets using the algorithm K-means for clustering, in order to compare the two approaches and choose the one that improves the clustering results.

The clustering results are presented in Table II. We can clearly see that the clustering results using text data representation with BERT outperform the clustering using text data representation with TFIDF in all three datasets which can be explained by the fact that the representation of TF-IDF does not take into consideration the semantic meaning and the context of the words in the corpus compared to BERT. Therefore, we used the BERT model to represent text documents in the method we proposed and the conducted experiments.

The second part of experiments that we conducted concern text stream clustering on the dataset Newsgroup20 using Clustream and different methods of dimensionality reduction with 2 dimensions. The problem using text data is that these type of datasets are in very high dimensionality and the classical data stream clustering methods can not be used. We tested 3 approaches: SVD, t-SNE and UMAP.

TABLE II
CLUSTERING RESULTS USING K-MEANS

Method	NewsGroup20		AG News		BBC News	
	Purity	Sil	Purity	Sil	Purity	Sil
TF-IDF + K-means	0.368	0.007	0.889	0.013	0.280	0.006
BERT + K-means	**0.611**	**0.040**	**0.957**	**0.066**	**0.831**	**0.036**

TABLE III
DATA STREAM CLUSTERING RESULTS

Method	Purity	Silhouette
OTTC using SVD	0.25	0.37
OTTC using t-SNE	0.57	0.44
OTTC using UMAP	**0.63**	**0.47**

The results shown in Table III were obtained by setting the different parameters of the algorithm Clustream in the Framework MOA as follows: decayHorizon = 1000, evaluationFrequency = 1000, decayThreshold = 0.01, normalize = True, maxNumKernels = 100, kernelRadiFactor = 2.

From the above table, we can see that UMAP results outperform SVD and t-SNE in terms of Purity: 0.63 instead of 0.25 and 0.57 with SVD and t-SNE respectively, and for the silhouette coefficient, we obtain 0.47 instead of 0.37 with SVD and 0.44 with t-SNE. We can also notice that the results obtained with t-SNE and UMAP are very similar as UMAP is based on t-SNE.

The figures 2, 3 and 4 show a visualization of stream clustering results based on dimensionality reduction methods that we mentioned above. The circles in green represent the micro-clusters, while the circles in red represent the final clusters.

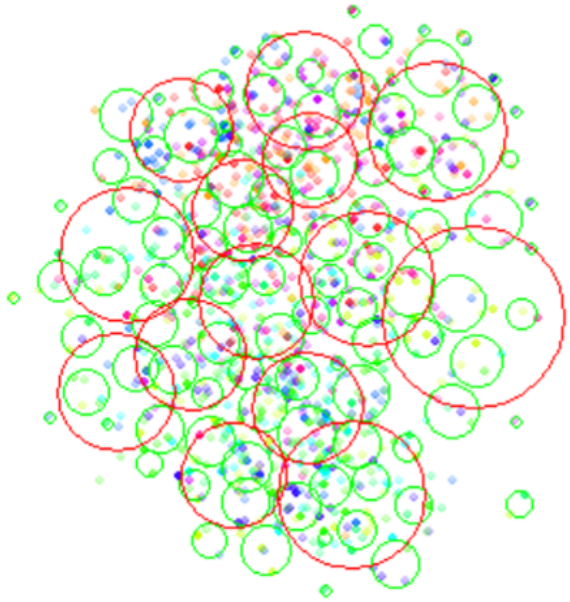

Fig. 2. Stream clustering of Newsgroup20 using SVD

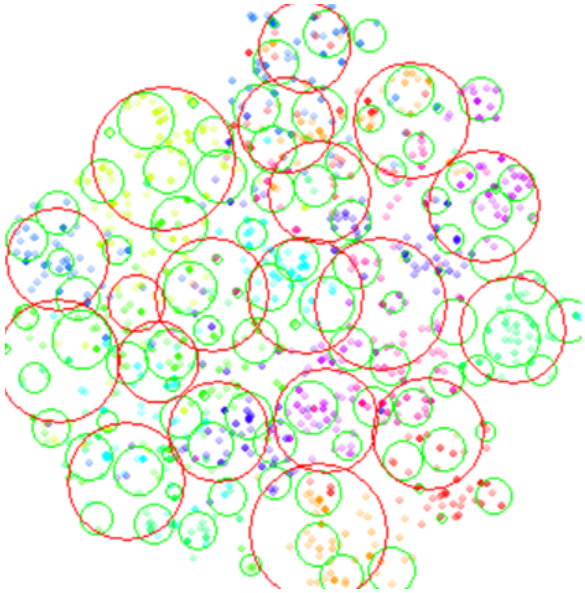

Fig. 3. Stream clustering of Newsgroup20 using t-SNE

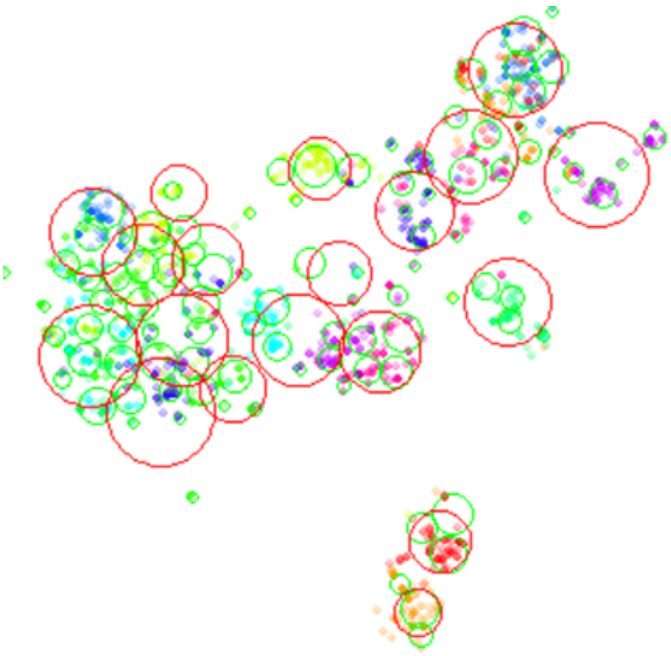

Fig. 4. Stream clustering of Newsgroup20 using UMAP

The color of points indicates the real classes of data.
We can confirm by these figures the performance of the UMAP method over the other methods. It can be observed that the data points of different clusters in the clustering results with SVD and t-SNE are not well separated compared to the results that we obtained with UMAP, where the whole clusters of data points agglomerated in the same cluster zones.

Reducing the dimensionality of high dimensional data is a essential step on the text stream data clustering which allows u to reduce the computational cost and to improve the clustering

TABLE IV
STREAM CLUSTERING RESULTS USING DIFFERENT # OF DIMENSIONS

Method	NewsGroup20		AG News		BBC News	
	Purity	Sil	Purity	Sil	Purity	Sil
OTTC_2	0.63	0.47	0.79	0.73	0.90	0.80
OTTC_10	0.81	0.59	0.88	0.81	0.99	0.90
OTTC_20	0.93	0.56	0.93	0.66	1	0.77
OTTC_30	0.95	0.48	0.60	0.54	0.64	0.50
OTTC_40	0.95	0.34	0.35	0.52	0.27	0.50
OTTC_50	0.99	0.39	0.37	0.52	0.18	0.50

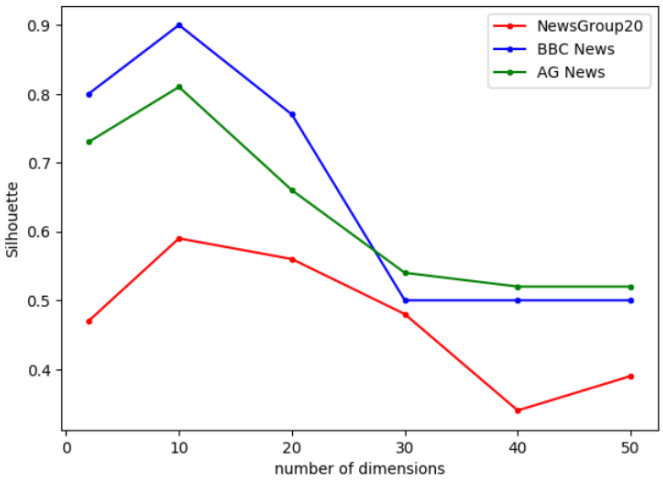

Fig. 5. Variation of silhouette according to number of dimensions

performance. One of the disadvantage of the dimensionality reduction is the loose of the information by choosing a wrong number of reduced dimensions.

We evaluate the results of the proposed OTTC method using different numbers of components for data embedding on the three textual datasets: Newsgroup20, BBC News and AG News. For each dataset, the parameters of the stream clustering algorithm were fixed as follows: For the newsgroup20 and AG News datasets we set: decayHorizon = 1000, evaluationFrequency = 1000, decayThreshold = 0.01, normalize = True, maxNumKernels = 100, kernelRadiFactor = 2, horizon = 1000. And for BBC News, we set: decayHorizon = 200, evaluationFrequency = 200, decayThreshold = 0.01, normalize = True, maxNumKernels = 100, kernelRadiFactor = 2, horizon = 200. The choice of these parameters were obtained using empirical computations with a cross-validation.

The table IV shows the obtained results. The index of the OTTC in the Method column indicates the number of dimensions vectors in the embedding process. The number of dimensions stop at 50, because overall, after 50 dimensions the quality of the OTTC approach will decrease.

We plot these results in order to observe the variation of purity and silhouette measures regarding the number of dimensions and analyze their impact on the clustering results and its quality.

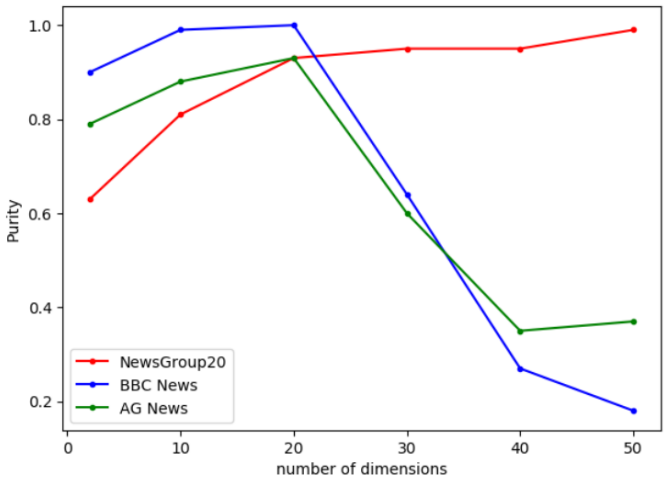

Fig. 6. Variation of purity according to number of dimensions

It can be noticed from the figure 5 that the silhouette of all the three datasets starts to decrease as the number of components used in the projection using UMAP method increases after 20 dimensions. For the purity measure (figure 6), the same behavior was observed for the two datasets BBC News and AG News, which decreased after 20 dimensions, while for Newsgroup20 dataset, we notice that the purity sometimes goes up and down depending on the number of dimensions which can be explained by the nature of dataset which has a large number of classes that makes high purity easy to achieve.

V. Conclusion

In this paper, we proposed a new method of clustering high dimensional text data stream called OTTC (Online topological text clustering). This method combines the topological representation learning and the online incremental clustering model in order to learn the representation of the stream and regroup similar textual data in a smaller dimension space. The proposed approach were validated on several textual datasets and obtained results outperforms the classical methods in terms of the Purity and Silhouette index. We note also, that the proposed method is scalable due to the use of the representation space learning. As future works, meta-data available in the dataset will be used in order to outperform the obtained results, and the multi-view context of the problem will be also analysed.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2002, pp. 1–16.

[2] S. Guha and N. Mishra, "Clustering data streams," in *Data Stream Management*. Springer, 2016, pp. 169–187.

[3] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71–80.

[4] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. de Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–31, 2013.

[5] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 535–569, 2015.

[6] A. Zubaroğlu and V. Atalay, "Data stream clustering: a review," *Artificial Intelligence Review*, vol. 54, no. 2, pp. 1201–1236, 2021.

[7] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB conference*. Elsevier, 2003, pp. 81–92.

[8] M. Carnein and H. Trautmann, "Optimizing data stream representation: An extensive survey on stream clustering algorithms," *Business & Information Systems Engineering*, vol. 61, no. 3, pp. 277–297, 2019.

[9] C. C. Aggarwal, "A survey of stream clustering algorithms," in *Data Clustering*. Chapman and Hall/CRC, 2018, pp. 231–258.

[10] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proceedings 18th International Conference on Data Engineering*. IEEE, 2002, pp. 685–694.

[11] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "Streamkm++ a clustering algorithm for data streams," *Journal of Experimental Algorithmics (JEA)*, vol. 17, pp. 2–1, 2012.

[12] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowledge and Information Systems*, vol. 15, no. 2, pp. 181–214, 2008.

[13] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 2006, pp. 328–339.

[14] D. K. Tasoulis, G. Ross, and N. M. Adams, "Visualising the cluster structure of data streams," in *International Symposium on Intelligent Data Analysis*. Springer, 2007, pp. 81–92.

[15] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 133–142.

[16] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM SIGMOD Record*, vol. 25, no. 2, pp. 103–114, 1996.

[17] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of Research and Development*, vol. 1, no. 4, pp. 309–317, 1957.

[18] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, 1972.

[19] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in Neural Information Processing Systems*, vol. 15, 2002.

[20] M. Vladymyrov and M. Carreira-Perpinan, "Entropic affinities: Properties and efficient numerical computation," in *International Conference on Machine Learning*. PMLR, 2013, pp. 477–485.

[21] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[22] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[23] K. Lang, "20 newsgroups dataset," empty. [Online]. Available: http://people.csail.mit.edu/jrennie/20Newsgroups/

[24] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015.

[25] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 2006, pp. 377–384.

[26] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," in *Proceedings of the First Workshop on Applications of Pattern Analysis*. PMLR, 2010, pp. 44–50.

[27] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," *in Technical Report TR 01–40, Department of Computer Science, University of Minnesota*, 2001.

[28] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.