SN Computer Science (2025)6:413
https://doi.org/10.1007/s42979-025-03920-7

ORIGINAL RESEARCH

Online Clustering of Massive Text Data Streams

Maha Ben-Fares[1,3] · Parisa Rastin[2] · Nistor Grozavu[1] · Pierre Holat[3,4]

Received: 4 June 2024 / Accepted: 22 March 2025

Abstract
Clustering large text data streams is a complex and computationally demanding task due to its unique challenges including high dimensionality, data sparsity, evolving data distributions, and the curse of dimensionality. Traditional clustering techniques often struggle to keep up with these dynamic changes and high-velocity nature of streaming text data while maintaining efficiency and accuracy. In this paper, we introduce OMTStream, a novel approach that combines an adaptive text representation technique with an online clustering based CluStream. Our method continuously learns meaningful semantic representations of text while reducing dimensionality, allowing for fast and efficient real-time clustering. By leveraging micro-clusters that evolve over time, OMTStream can track emerging patterns and adapt to shifting data distributions without the need for retraining from scratch. We evaluate our approach on real-world datasets, demonstrating its superior performance compared to existing techniques. Through a comprehensive set of experiments, we also validate the effectiveness of OMTStream across various scenarios, highlighting its robustness, adaptability, and computational efficiency in handling large, continuously evolving text data streams.

Keywords Text data streams · Data stream clustering · High-dimensional data · Clustering · Feature extraction · Dimensionality reduction · Real-time text analysis

Introduction

In today's interconnected world, the rapid growth of digital information has led to an explosion in the volume of data generated and stored. This exponential growth presents a unique challenge: how can we effectively manage and analyze this flow of information in real-time, extracting meaningful patterns and insights as the information arrives? This is where data stream mining techniques, including data stream clustering, come into play to tackle this challenge. Unlike static datasets, where all data are available at once, data streams represent a continuous and unbounded flow of information. This dynamic nature introduces specific challenges, such as the one-time pass constraint, which means that each data object can be read only once without the possibility of revisiting previous data points, the volume of incoming data which is often so large that it cannot be stored in its entirety, necessitating real-time processing and analysis to handle the vast and ever-growing information flow. Finally, there is also the constraint of evolving data over time, requiring algorithms that can quickly adapt to new patterns and changes.

while there has been significant research on general data stream clustering techniques, the specific area of textual data stream clustering has received relatively less attention. Most existing studies focus on short text streams, such as social media posts and instant messages, rather than large or massive text document streams.

In this paper, we focus on data stream clustering for large textual data. Originating from diverse sources such as scientific literature, news websites, and online forums,

✉ Maha Ben-Fares
maha.ben-fares@cyu.fr

Parisa Rastin
parisa.rastin@loria.fr

Nistor Grozavu
nistor.grozavu@cyu.fr

Pierre Holat
pierre.holat@f-initiatives.com

[1] ETIS, CY Cergy Paris Université, Pontoise, France

[2] LORIA, Université de Lorraine, Nancy, France

[3] F.Initiatives, Puteaux, France

[4] LIPN, Université Sorbonne Paris Nord, Villetaneuse, France

Published online: 23 April 2025

textual data presents additional layers of complexity and challenges. Its inherent high dimensionality makes traditional clustering methods less effective and computationally prohibitive, necessitating innovative approaches to manage the high dimensionality and ensure efficient clustering.

We propose a new data stream clustering method designed for large document texts, referred to as OMTStream. this method groups similar documents within the stream into clusters, enabling efficient summarization of textual content over time. it combines dimensionality reduction techniques and text representation methods with online clustering in order to continuously cluster high-dimensional streaming text data in smaller dimension space. The Fig. 1 provides an overview of the different steps in our method.

The proposed method was evaluated using real data sets, it achieves better clustering quality in comparison with the traditional stream clustering methods in both speed and clustering quality.

The rest of this paper is organized as follows: In the section "Basic Concepts of Data Stream Clustering", we explain some basic concepts of data stream clustering, including different type of data structures and time window models used to handle data streams. The section "Related Work" presents the related work and some various types of data stream clustering algorithms from the literature. The sections "Representation Learning" and "Dimensionality Reduction" discuss the methods of representation learning and dimensionalty reduction used in our experiments. The proposed OMTStream approach is described in the section "Our Proposed Approach" and the obtained results and different sets of experiments to valid our technique in the section "Experimental Results". Finally, the paper ends with a conclusion and some future work for the proposed method.

Basic Concepts of Data Stream Clustering

Due to the volume and rapid arrival of streaming data, it is impractical to store entire data permanently. therefore Specialized data structures are needed to gradually summarize it instead of storing it completely. This approach tackles the challenge of handling unbounded data in real-time, ensuring effective clustering result while reducing memory and computational demands. We distinguish the following most used data structures [17, 22, 29]:

– Micro-Cluster: It extends the cluster feature concept introduced the first time in BIRCH algorithm [24]. It stores summary statistics of data points that arrive in the stream. Each micro-cluster keeps track of the number of points it contains, the sum of these points, and the sum of their squares. We define a micro cluster 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 $T_1, ..., T_n$:

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

where N is the number of d-dimensional data points in cluster and:

$$\overrightarrow{CF1}^d = \sum_{i=1}^{N} X_i \quad \overrightarrow{CF2}^d = \sum_{i=1}^{N} X_i^2 \quad CF1^t = \sum_{i=1}^{N} T_i \quad CF2^t = \sum_{i=1}^{N} T_i^2$$

This summary information can be used to quickly update cluster centroid and radius using the Eqs. 2 and 3.

$$radius = \sqrt{\frac{\overrightarrow{CF2}^d}{N} - \left(\frac{\overrightarrow{CF1}^d}{N}\right)^2} \tag{2}$$

$$centroid = \frac{\overrightarrow{CF1}^d}{N} \tag{3}$$

– Tree-based structure: It organize data in a hierarchical tree structure in order to facilitate efficient query processing, allowing for efficient insertion, deletion,

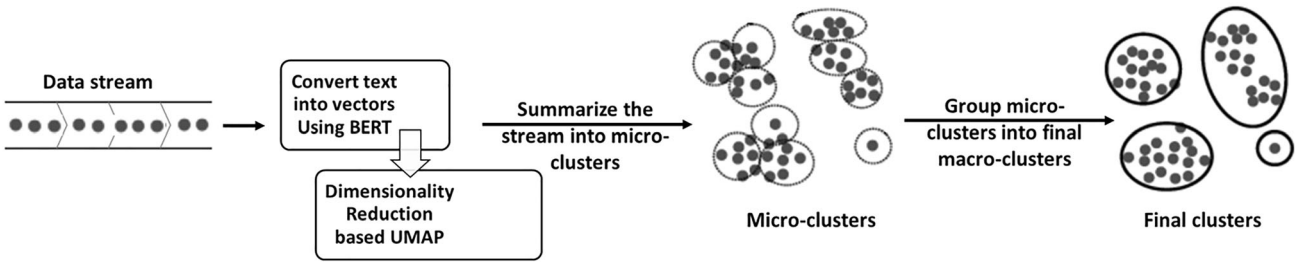


Fig. 1 Overall scheme of the OMTStream method

and nearest neighbor searches which are critical for clustering operations.
- Grid structure: This involves dividing the data space into a grid of cells, each representing a potential micro-cluster. As data points arrive, they are mapped to these cells, and clustering can be performed on the cells istead of individual data points, which reduces complexity.
- Prototype arrays: where only some representative instances that summarize the data is kept.

The selection of time window model is an another important concept used to deal with stream data for clustering. It plays an essential role in managing the temporal aspects of data streams, defining the subset of data to consider for clustering at any given time [22]. There are four main time window models in data stream clustering as illustrated in Fig. 2:

1. Landmark window model which focus on the data received from a specific starting point, known as the landmark, up to the current time. This approach allows the inclusion of all data points arriving after the landmark in the clustering process. This model can be useful in scenarios where understanding the evolution of data patterns from a specific starting point is crucial.
2. Sliding window model: it considers only the most recent portion of data within a defined window that slides over time as new data arrives. This model is particularly useful in environments where it is important to maintain a continuous analysis of data but also necessary to focus on the most recent data points.
3. Damped window model: this model manages and analyzes data streams by gradually reducing the importance or weight of older data points using a decay function while prioritizing more recent information. This model offers a practical compromise between completely forgetting older data (as in a pure sliding window model) and maintaining a full historical record (as in a landmark window model). It provides a way to keep the system's memory requirements while still allowing for some degree of historical data analysis, but the choice of an appropriate decay factor can be challenging and may require extensive experimentation.
4. Titled or pyramidal window model which partitions the stream into multiple windows that increase in size and decrease in resolution as they go back in time. This means that more recent events are captured in smaller, finer-grained windows, while older events are aggregated into larger, coarser-grained windows.

Related Work

This section reviews existing research on data stream clustering algorithms for numerical and textual data. These algorithms are generally classified into three main categories according to the state of the art and various survey papers [2, 8, 17]: partitioning-based, density-based, and model-based methods. Each of these categories offers different solution to the challenges posed by data streaming.

Partitioning-Based Methods

These methods divide the data into several distinct groups or partitions, where each group represents a cluster, and each data point is assigned to a cluster based on its proximity to the cluster's centroid. The goal is to minimize the distance between data points and their respective cluster centroids, thereby optimizing the homogeneity within clusters and maximizing the heterogeneity between clusters. These methods are mostly used because of their simplicity, efficiency, and ease of implementation. We found several popular algorithms in the literature within this category: STREAM, Clustream, StreamKM++ [1] and SWClustering [28].

Developed by Guha et al. [11], the STREAM algorithm represents an early and influential method for clustering data streams, utilizing the k-median technique. This framework processes the stream by dividing it into chunks of fixed-size m. Within each of these chunks, the k-median clustering algorithm is applied minimizing the Sum of Squared Error (SSQ) of the data points. Over time, as more

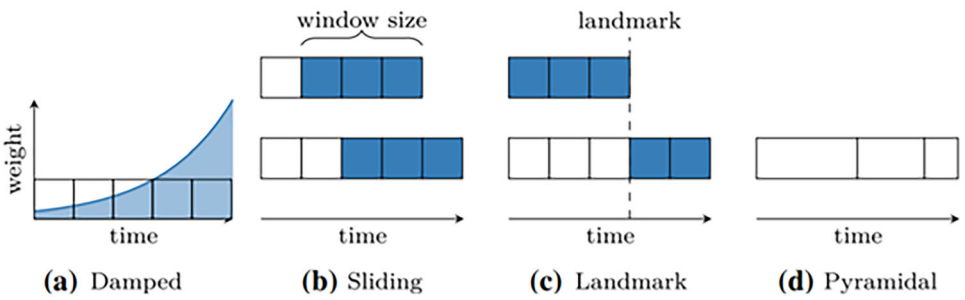


Fig. 2 Time window models

batches are processed, the system accumulates k medians from each batch and stores them in a prototype array. Once the collection of medians reaches a set threshold m, the k-median algorithm is again applied, but this time on the stored medians, to refine and update the set of medians. This cyclic process allows the framework to efficiently manage massive data streams by regularly updating the clustering model to reflect new data, while keeping computational and memory requirements in check.

CluStream [3] is a framework designed for clustering data streams over various time horizons. It uses the online-offline approach to perform clustering. During the online phase, the stream is summarized, and statistical information about the data is stored using micro-clusters. These micro-clusters are then periodically re-clustered into larger, more meaningful clusters through the offline process.

Clustream works as follows: First, for the initialisation, we create the first micro-clusters by applying k-means on the first data that arrives from the stream. Then, there is a micro-cluster maintenance phase, which is described in details by the Algorithm 1, it explains how we affect each data point to his closest micro-clusters and how to manage the total number of micro-clusters by deletion or merging operations. Finally, in the offline phase, we apply k-means over the micro-clusters generated in the online phase to get the final clusters. This phase can be executed periodically or on user-demand.

Algorithm 1 Online micro-cluster maintenance

When a new data point y_i arrive from the stream:
- Compute the distance between the point y_i and each of the maxNumKernels micro-cluster centroids.
- clu $\leftarrow$ the closest micro-cluster to y_i.
- closestDist $\leftarrow$ the distance between y_i and its closest micro-cluster clu.
- radius $\leftarrow$ calculate the max boundary of clu by the equation 2.

if closestDist $<$ radius: y_i is falling within the max-boundary of clu **then**
- y_i is assigned to clu
- Update the micro-cluster clu by using the incrementality property.

else
- A new micro-cluster will be created using y_i.
- 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, by adding their statistical summaries.
- **end if**

end if

Density-Based Methods

These methods focus on grouping data based on the density of areas within the data space, where dense areas of data points form clusters separated by less dense, or sparse, regions. Algorithms like DenStream, SDStream, C-DenStream and D-Stream fall into this category. These algorithms excel in handling noise and identifying clusters of arbitrary shapes. However, their performance heavily depends on the correct setting of input parameters, such as the radius of neighborhood and density thresholds, which can be a limitation if not accurately estimated.

DenStream [7] is an extension of the DBSCAN algorithm for data streams. It is designed to handle evolving data patterns and detect clusters and outliers in real-time. DenStream operates by maintaining two types of micro-clusters: core Micro-Clusters (CMC) and outlier Micro-Clusters (OMC). A micro-cluster in density based-methods is defined by its center, radius, and weight, which represents the density of the data points within it.

When a new data point arrives, DenStream checks if it falls within the radius of one of the existing core micro-cluster, defined by the parameter ϵ. If it does, the micro-cluster's center and weight are updated to include the new point. If the data point does not fit into any core-micro-cluster, the algorithm checks if it can be absorbed into an existing outlier-micro-cluster. If neither is possible, a new outlier micro-cluster is created for the point.

DenStream uses the damped window model. The weight of each micro-cluster decreases over time according to the fading parameter λ, allowing the algorithm to adapt to changes in the data stream by giving less importance to older data.

In the process, an outlier micro-cluster can be promoted to a core micro-cluster if it gains sufficient weight to exceed a threshold β. Conversely, if a core micro-cluster's weight falls below this threshold due to the fading mechanism, it can be demoted to an outlier micro-cluster or discarded if it becomes too sparse.

In the offline phase, like in Clustream algorithm, we apply DBSCAN algorithm over the core micro-clusters to get the final clusters.

Several algorithms that extends DenStream have been developed to improve its performance in different contexts. SDStream, developed by Ren et al. [19], extends DenStream by incorporating a sliding window model and storing potential and outlier micro-clusters in the form of Exponential Histogram of Cluster Feature (EHCF). C-DenStream, developed by Ruiz et al. [21], combines DenStream with C-DBSCAN and incorporates a particular form of background knowledge referred to as instance-level constraints. These constraints specify which instances must belong to the same or different clusters. The instance-level constraints are then converted into potential micro-cluster level constraints and final clusters are generated on the potential micro-clusters using C-DBSCAN.

Model-Based Methods

Model-based clustering algorithms are techniques used to group data points into clusters based on probabilistic models. Unlike distance-based and density-based methods, model-based approaches assume that the data is generated from a mixture of known probability distributions. The parameters of these distributions are frequently learned with an Expectation-Maximization (EM) algorithm as new data arrives, which yields to a good clustering results, especially in situations where complex data necessitates advanced modeling to reveal cluster structures. In this category, we found SWEM algorithm [9] proposed by Dang et al. and SNCStream [5].

Most of the methods we previously discussed are applied to numerical data. Research on data stream clustering for textual data has been limited and particularly focusing on short text data streams such as tweets or social media posts. Among the first works in this area was by Aggarwal et al. [4], who proposed a method for clustering text and categorical data streams. This method summarizes incoming data into fine-grained cluster droplets, which serve as condensed representations of the data, facilitating online analytical processing and allowing for dynamic cluster formation based on various user-defined parameters. Yin et al. [23] introduced the MStream clustering algorithm for short text streams based on the dirichlet process multinomial mixture (DPMM) model. It involves both a one-pass clustering process and an update clustering process for each batch. Additionally, it applies Gibbs sampling multiple times to the same batch of texts, refining the initial clustering results obtained during the one-pass process. Thereafter, by the same author, MStreamF was proposed [23] to enhances the MStream algorithm by efficiently handling outdated documents. It achieves this by using forgetting rules to delete clusters associated with outdated batches, which helps manage memory by removing irrelevant information. Recently, an Online Semantic-enhanced Dirichlet Model (OSDM) for short text streams clustering was introduced by Kumar et al. [13]. This algorithm integrates semantic information derived from word occurrences (context) into a novel graphical model, enabling the automatic clustering of each incoming short text in real-time.

The method that we propose in this paper is partitioning-based method designed for clustering massive textual stream data, it is based on the Clustream algorithm for the online step to incrementally group similar high dimensional text data into clusters. Our method uses two embedding methods for text representation and dimensionality reduction in order to learn the representation of the stream without losing the structure knowledge of the data in reduced space.

Representation Learning

In order to create and learn meaningful representations of document texts and convert it into numerical vectors, various methods are available. These methods fall into two main categories: classical vector representation like TF-IDF and neural networks based models such as BERT and Doc2Vec.

TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency, is a traditional NLP method that reflect how important a word is to a document in a collection of documents or corpus D. It is composed of two parts: the Term Frequency (TF) [16], which measures the frequency of a word in a specific document, and the Inverse Document Frequency (IDF) [12], which weighs the importance of a term based on its rarity across all documents in the corpus, so the words that occur frequently across many documents is less important and less distinguishing. The TF-IDF of a term t in a given document d is defined as follows:

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

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{5}$$

$$idf(t,D) = \log \frac{N}{|\{d \in D, t \in d\}|} \tag{6}$$

where $f_{t,d}$ is the number of times the term t appears in the document d, $\sum_{t' \in d} f_{t',d}$ is the total number of terms in the 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 produces a vector where each dimension corresponds to a term in the corpus, effectively highlighting terms that are especially relevant to a document. Although TF-IDF is easy to calculate and useful in various machine learning scenarios, it has its limitations. One of his major drawback is the curse of dimensionality as the size of TF-IDF vectors is proportional to the vocabulary size of the corpus. Additionally, TF-IDF lacks the capability to understand the context or semantic meanings of words, which limits its effectiveness for more advanced linguistic tasks that require deeper textual understanding like clustering. Thus, we need an advanced neural network-based techniques that can convert text to numerical vectors while capturing the important semantic information, such as BERT or Doc2vec.

Doc2vec

Doc2Vec, also known as Paragraph Vector, is a neural network-based approach that extends the principles of Word2Vec to learn distributed representations of entire documents or paragraphs. While Word2Vec focuses on capturing word embeddings, Doc2Vec extends this concept to generate fixed-length vector representations for larger text units. Introduced by researchers at Google [15], this technique trains a neural network in an unsupervised manner to predict the next word in a context window, considering both word and document vectors. Doc2vec can be implemented in two main modes: Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag of Words Model of Paragraph Vectors (PV-DBOW).

PV-DM: in this model, each word is represented by a unique vector and each document is represented by a unique document vector. To predict a target word, the model considers a window of surrounding context words and the document's vector. These vectors can be concatenated or averaged to form a single, unified feature vector. This combination captures both the local context provided by surrounding words and the global context provided by the document vector.

Figure 3b represents the architecture of the PV-DM model of the Doc2vec, it consists of a neural network with three layers: input, projection, and output. The input layer includes the context word vectors corresponding to the words surrounding the target word v(t) along with the unique document ID vector. These vectors are then projected and concatenated in the projection layer, forming a combined feature vector. This vector is used in the output layer to predict the target word v(t). During training, the model adjusts the word and document vectors to maximize the prediction accuracy of the target word, effectively learning a numerical representation of the document that captures its contextual and semantic essence.

PV-DBOW: in this mode, the model ignores context words and simply predicts words independently based only on the document vector. During training, words are randomly sampled from the paragraph, and the model's objective is to maximize the probability of these sampled words given the document vector.

Unlike PV-DM, the PV-DBOW's architecture illustrated in Fig. 3a only uses the document ID vector as input, without any context words. The document vector connects directly to the projection layer, which is then used in the output layer to predict words from the document.

BERT

BERT is a transformer-based model developed by Google [18] that utilizes the attention mechanism to learn contextual word embeddings. In order to enhance understanding of language context, BERT's training employs two strategies: Masked Language Modeling (MLM) which consists in predicting the identity of a random masked input token using its context only and Next Sentence Prediction (NSP) which predicts if two sentences are adjacent or not. Sentence-BERT (SBERT) extends BERT's capabilities to create contextually rich sentence embeddings. While BERT is primarily focused on word-level representations, SBERT aims to capture the semantic meaning of entire sentences. SBERT achieves this by incorporating siamese and triplet network structures into the BERT architecture. Siamese networks are used for tasks like sentence similarity, where two sentences are compared and a similarity score is computed. Triplet networks, on the other hand, are used for tasks like semantic search, where a query sentence is compared against multiple candidate sentences to identify the most semantically similar ones. We utilized two pre-trained models of SBERT in our experiments, which were recommended by the official website of sentence-transformer itself: all-MiniLM-L6-v2 and all-mpnet-base-v2.

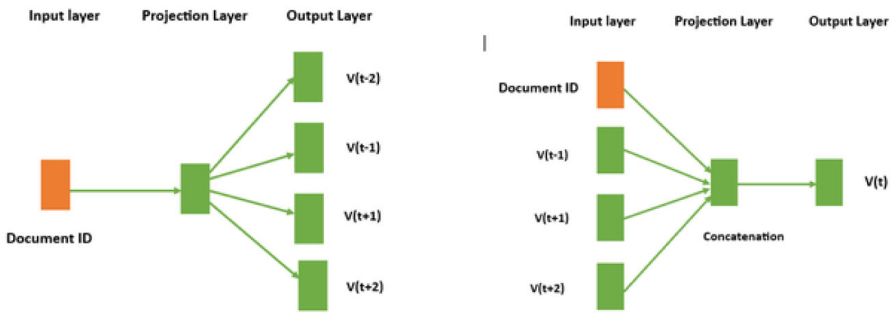


Fig. 3 The architecture of the two variants of Doc2vec [15]

(a) PV-DBOW architecture

(b) PV-DM architecture

Dimensionality Reduction

Text representation methods like TF-IDF, Doc2vec and BERT typically produce high-dimensional vectors where each dimension corresponds to a specific feature or term. However, as the dimensionality increases, several challenges arise, including the curse of dimensionality, sparse data, the increased computational complexity and another significant issue is the "crowding problem," where Euclidean distance becomes less informative because points tend to be equidistant from each other, making it hard to distinguish between them and decrease the efficiency of the clustering algorithms. To address these issues, we apply dimensionality reduction techniques before the clustering step. There are two types of dimensionality reduction methods: linear like SVD and non-linear methods like t-SNE and UMAP.

Singular Value Decomposition (SVD) is a powerful linear algebra technique used in dimensionality reduction. It decomposes a matrix A (e.g., term-document matrix in TF-IDF) into three matrices:

$$A = U\Sigma V^T \tag{7}$$

Where, U and V are orthogonal matrices, and Σ is a diagonal matrix containing singular values that are sorted in descending order. In text processing, reducing dimensionality involves selecting the top k singular values and their corresponding vectors in U and V, resulting in approximations of the original matrix. This approximation captures the most significant linear relationships in the data.

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique used for dimensionality reduction and is particularly popular for visualizing high-dimensional datasets in two or three dimensions. It starts by converting the Euclidean distances between points into conditional probabilities that represent similarities. The similarity of datapoint x_j to datapoint x_i is given by a conditional probability $p_{j|i}$, which is defined as:

$$p_{j|i} = \frac{\exp(-\frac{||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{||x_i - x_k||^2}{2\sigma_i^2})} \tag{8}$$

The joint probabilities p_{ij} are then symmetrized:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{9}$$

In the low-dimensional space, the similarities q_{ij} are computed using a Student-t distribution:

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

The goal of t-SNE is to minimize the Kullback–Leibler divergence between the two distributions P (high-dimensional) and Q (low-dimensional), focusing particularly on maintaining local structures. which effectively groups similar data points together while pushing dissimilar ones apart. The cost function to be minimized is defined as:

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

While its gradient is given by:

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

Uniform Manifold Approximation and Projection (UMAP) is a non-linear technique that is based on manifold learning and topological data analysis. It constructs a high-dimensional graph representing the manifold, and then optimizes a low-dimensional graph to be as structurally similar as possible. UMAP emerges as the more suitable technique for dimensionality reduction compared to SVD and t-SNE. UMAP's superior scalability and faster processing capabilities make it especially

advantageous for handling large datasets efficiently. Moreover, unlike t-SNE, which primarily emphasizes local data clusters and is highly sensitive to parameter settings, UMAP focuses on preserving the local and more global structure of the data. This characteristic is crucial for the clustering algorithms as it ensures that the broader relationships within the data are maintained. UMAP is particularly useful for tasks that require a balance between local and global integrity, such as thematic clustering and outlier detection, while t-SNE is more suitable for exploratory analysis and visualization purposes.

Our Proposed Approach

In our proposed approach, we introduce an innovative approach to effectively cluster text documents in real-time as they are streamed. The Algorithm 2 illustrates the different steps of our method in details. First, each incoming document from the stream is transformed into a vector using the well-known text representation technique Sentence-BERT. This step converts textual data into numerical vectors while capturing the rich semantic information embedded in the text, which is crucial for the subsequent clustering process. Then, in order to enhance the clustering performance and reduce computational complexity, we reduce the dimensionality of the BERT-generated document vector based on UMAP. By reducing the dimensionality of the feature space, we aim to retain relevant information while simultaneously addressing the challenge posed by the curse of dimensionality.

Algorithm 2 Online Clustering Massive Text Data Streams: OMTStream

Input: Given a set of stream text documents $S = \{d_1, \ldots, d_n\}$
- Q: number of macro-clusters; maxNumKernels: number of micro-clusters.
- k: the number of neighbors to use; m: the dimension of the original space;
- r: the dimension of the target reduced space.

Output: Set of K clusters.

1- Generate document embedding e_i for each document d_i by using sentence-transformers of Bert model.
2- Construct weighted kNN graph, that describes the distances between data points in high dimensional space, by calculating p_{ij} as follows:

$$p_{ij} = p_{j|i} + p_{i|j} - p_{j|i}p_{i|j} \quad ; \quad p_{j|i} = \begin{cases} exp\left(-\frac{\|e_i - e_j\|_2 - \rho_i}{\sigma_i}\right) & \text{if } e_j \in \mathcal{N}_i \\ 0 & \text{Otherwise} \end{cases}$$

where $\rho_i = \min\{\|e_i - e_{i,j}\|_2; 1 \leq j \leq k\}$ is the distance from x_i to its k-nearest neighbor. σ_i representing a scale parameter that satisfies the following condition using binary search:

$$\sum_{j=1}^{k} exp\left(-\frac{\|e_i - e_{i,j}\|_2 - \rho_i}{\sigma_i}\right) = \log_2(k)$$

3- Calculate the similarity between points y_i and y_j in the reduced-dimensional projection space with:

$$q_{ij} = \left(1 + a\|y_i - y_j\|_2^{2b}\right)^{-1}$$

4- Find the y_i that minimizes the difference between p_{ij} and q_{ij} using the following cost function:

$$C = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \left(p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right)\right)$$

5- Create the first maxNumKernels micro-clusters by applying k-means on the first data that arrives from the stream.
6- for each new point y_i that arrives, where $y_i \in \mathbb{R}^r$ and $r << m$, online micro-cluster maintenance is performed using Algorithm 1.
7- Apply k-means over the micro-clusters to get the Q final macro-clusters.

Table 1 Datasets description

Datasets	#documents	#classes
Newsgroup20	18,000	20
AG News	127,600	4
BBC News	2225	5
DBpedia	560,000	14

Finally, our approach integrates the concept of micro-clustering, inspired by the CluStream algorithm, to handle the dynamic nature of streaming data effectively. We try to assign the reduced document vector that we obtained to its closest micro-cluster, on condition that it fits well and its inclusion does not exceed the micro-cluster's defined boundary. This boundary is determined by a radius derived from the micro-cluster's statistics. If the data point is successfully added to this micro-cluster, we update its statistical information to reflect the addition of the new data. If it does not fit well into any existing micro-cluster, we create a new micro-cluster if there is available memory. If not, we create space for the new micro-cluster by deleting the oldest micro-clusters that are no longer updated, or we merge the two closest micro-clusters and update their statistical information by combining their attributes. By leveraging the strengths of both text representation techniques, dimensionality reduction, and stream clustering, our proposed approach offers a robust framework for text data stream clustering, capable of handling the challenges posed by dynamic and high dimensional text data streams.

Experimental Results

Datasets and Test Setup

We evaluate the proposed OMTStream method using four public textual datasets: Newsgroup20 [14], AGNews [26], DBpedia [25] and BBC_News [10] with varying numbers of documents and different number of classes. A summary of these datasets is provided in Table 1.

Newsgroup20: is a popular collection of approximately 20,000 discussion group posts, evenly distributed across 20 different newsgroups. Each newsgroup corresponds to a distinct topic, ranging from sports and politics to science and technology. The dataset is divided into training and testing subsets, typically based on the date of the posts.

AG News: is a sub-dataset of AG's corpus of news articles constructed by assembling titles and description fields of articles from the four largest categories within AG's Corpus: World, Sports, Business, and Technology. It contains a total of 127,600 documents.

BBC_News: this dataset consists of 2225 documents from the BBC News website corresponding to stories in five news areas from 2004–2005. These categories include entertainment, politics, business, sport, and tech.

DBpedia: the dataset is constructed by picking 14 non-overlapping classes from DBpedia 2014. From each of these 14 ontology classes, they are 40,000 training samples. The total size of the training dataset is 560,000.The dataset is compose to three columns corresponding to class index (1–14), title and content.

For our experiments, our proposed approach was developed using Massive Online Analysis framework MOA [6], a Java-based, open-source benchmarking tool specifically designed for data stream mining. It contains several data stream clustering algorithms and evaluation measures for running experiments.

Evaluation Measures

To evaluate the clustering results of our proposed model, we used a set of four quality measures: Purity [27], Silhouette [20], Normalized Mutual Information (NMI) and Sum of Squared distance (SSQ). These metrics are categorized in two primary groups: internal and external.

Internal measures, including SSQ and Silhouette Score, focus solely on characteristics of the clusters, such as their compactness and the separation between them. By concentrating on these aspects, internal measures provide insights into how well-defined and distinct the clusters are, offering valuable information about the quality of the clustering structure.

SSQ quantifies how closely data points are grouped within clusters by summing up the squared distances between each data point and the centroid of its assigned cluster. It is defined as follows:

$$SSQ = \sum_{k=1}^{k} \sum_{x_i \in D_k} \|x_i - \mu_k\|^2 \tag{13}$$

Lower SSQ values means better compactness of each cluster.

Silhouette index, on the other hand, calculates the average silhouette width for each sample, each cluster, and the entire dataset. It helps assess each partition based on its compactness and separation from others. To determine the silhouette width $S(i)$ for each data point, the following formula is used:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{14}$$

In this equation, $a(i)$ represents the average dissimilarity of the ith data point to all other points within the same cluster, and $b(i)$ denotes the average dissimilarity of the ith data point to all points in the nearest cluster, other than its own.

The Silhouette index values range from – 1 to 1, where a value close to 1 indicates that the data point is well-clustered, signifying a good fit within its cluster with significant separation from neighboring clusters. A value near 0 suggests that the data point is equidistant from its cluster and the nearest cluster, and a value approaching – 1 implies that the data point is poorly clustered or possibly misplaced, as it lies closer to a neighboring cluster than to its own. The average Silhouette can be also used to quantify the quality of the obtained clustering result.

External measures aims to compare the clustering results produced by the algorithm to a predetermined external ground-truth information.

Purity calculates the proportion of data points that were correctly assigned to their true clusters. It is defined as follows:

$$purity(\Omega, C) = \frac{1}{N}\sum_{k} \max_{c \in C} |\omega_k \cap c| \tag{15}$$

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.

NMI measures the amount of information shared between the predicted cluster assignments and the true class labels, providing a way to assess the agreement between the two. NMI varies from 0 to 1, where 0 indicates no mutual information, that the clustering is entirely independent of the true classes and 1 indicates perfect correlation.

Results

In our initial experiments, we demonstrate the significant influence of text embedding method used and the critical role of dimensionality reduction before the clustering on the quality and effectiveness of the clustering results. The Table 2 presents clustering results using the K-means method, comparing different text representation techniques, specifically TF-IDF and BERT, with and without the application of UMAP (Uniform Manifold Approximation and Projection) for dimensionality reduction. The metrics used to evaluate the performance of clustering results include Purity, Silhouette Score (Sil), and Normalized Mutual Information (NMI). Bold values indicate the highest scores achieved for each metric across the evaluated methods.

We can notice from the Table 2, that the traditional TF-IDF method shows relatively lower performance across almost all datasets and metrics compared to BERT. For instance, in the Newsgroup20 dataset, TF-IDF achieves a purity of 0.341, a silhouette score of 0.007, and an NMI of 0.351, which are significantly enhanced to 0.580, 0.038, and 0.576, respectively, when using BERT. This indicates that BERT's deep contextual embeddings provide a more nuanced and effective clustering capability than TF-IDF. We can notice also that the incorporation of UMAP with TF-IDF or BERT on almost all dataset improves remarkably both the silhouette score and NMI. Overall, the results indicate that the combination of BERT with UMAP generally provides superior clustering outcomes. This synergy enhances the effectiveness of clustering by leveraging BERT's deep contextual embeddings and UMAP's ability to reduce dimensionality while preserving the structure of the data.

We also conducted a comparative clustering analysis using our proposed approach along with various dimensionality reduction methods: SVD, t-SNE, and UMAP compared to the OSDM algorithm. The analysis was carried out on the Newsgroup20 dataset and the results are shown in Table 3.

This comparative analysis clearly shows that UMAP outperforms SVD and t-SNE in terms of both purity and silhouette scores. It achieves the highest purity score of 0.67, indicating a superior alignment of clusters with true categories, and highest silhouette score of 0.52, suggesting that the clusters mostly well-defined and separated from each other. We can also notice that the results obtained with t-SNE and UMAP are relatively close in comparison with results obtained with SVD, this can be explained by the fact that UMAP is based on the foundational principles of t-SNE.On the other hand, our method OMTStream outperforms OSDM regarding purity and silhouette, which proves that the method doesn't support the large textual data.

Choice of Number of Micro-clusters: the number of micro-clusters is an important parameter in our proposed approach that should be chosen carefully, it should be larger than the number of macro-clusters (actual clusters). However, a very large number of micro-clusters is time and memory consuming. In order to make a right decision about the optimal number of micro-clusters to choose, we tested our method using different values of micro-ratio,

Table 2 Clustering results using K-means

Method	NewsGroup20			BBC News			AG News		
	Purity	Sil	NMI	Purity	Sil	NMI	Purity	Sil	NMI
TF-IDF	0.341	0.007	0.351	0.811	0.013	0.669	0.290	0.006	0.018
TF-IDF+UMAP	0.567	0.433	0.484	0.766	0.516	0.672	0.277	0.667	0.018
BERT	0.580	0.038	0.576	**0.958**	0.066	**0.873**	**0.830**	0.036	0.586
BERT+UMAP	**0.630**	**0.443**	**0.631**	0.954	**0.602**	0.865	0.821	**0.468**	**0.621**

defined as the ratio of the number of micro-clusters to the number of macro-clusters. We then evaluated the clustering quality on the four datasets using the Sum of Squared distances (SSQ) as the criterion.

The micro-ratio values that we tested are: 1, 5, 10, 20, 30, and 40; this means that when the micro-ratio is about 10 for the AGNews dataset the actual number of micro-clusters is 40, for DBpedia dataset 140 and 200 for the Newsgroup dataset and 50 for BBC News, respectively. For the other parameteres, we set, window size w = 1000 for the DBpedia and Newsgroup20 datasets, for the BBC news dataset w = 200 and for AGNews dataset w=200.

Figure 4 displays our experimental results. For the four datasets, we can observe that as the micro-ratio increases, the average SSQ decreases and then stabilizes at 20 to 30. These values represent the optimal choice for achieving high-quality clustering while minimizing memory consumption.

Finally, we evaluate our proposed method using different numbers of dimensions of the reduced space and different document embedding methods including Doc2vec and two pre-trained models of SBERT 'all-MiniLM-L6-v2' and 'all-mpnet-base-v2'. For each dataset, the parameters of our text stream clustering algorithm were fixed as follows: decayThreshold = 0.01, normalize = True, kernelRadiFactor = 2. For the newsgroup20 we set: horizon = 1000, maxNumKernels = 400, for AG News: horizon = 200, maxNumKernels = 80, and for BBC News we set: horizon = 200, maxNumKernels = 100 and for DBpedia we set horizon = 1000 and maxNumKernels = 280. The choice of the parameters decayThreshold and horizon (the window size) was made based on the size of datasets and using empirical computations with a cross-validation. The Tables 4, 5, 6, 7 in the appendix A show the obtained results.

We plot these results in order to facilitates the observation of the variation of purity and silhouette measures regarding the number of dimensions and across various embedding methods.

we choose the radar chart format, which is used to compare multiple quantitative variables. This type of chart shows data on axes starting from the same point. The variables are usually evenly spaced along the circumference of the chart, and their values are plotted along the spokes. The end points of the values for each variable are connected by lines, creating a shape that easily visualizes strengths and weaknesses at a glance.

In this chart, three different variants of our proposed OMTStream method are compared across multiple number of dimensions labeled as D2, D10, D20, D30, D40, and D50. The three methods, indicated by different colors, show how they perform on each dimension; a longer line indicates higher values or better performance on that dimension.

It can be noticed from the Figs. 5b, 6b, 7b and 8b that the silhouette of all the four datasets starts to decrease as the number of components used in the projection based UMAP method increases after 20 dimensions. For the purity measure shown in Figs. 5a, 6a, 7a and 8a, a same behavior was observed for the three datasets DBPedia,

Table 3 OMTStream results of Newsgroup20 dataset

Method	Purity	Silhouette
OMTStream using SVD	0.25	0.37
OMTStream using t-SNE	0.57	0.44
OMTStream using UMAP	**0.67**	**0.52**
OSDM [13]	0.55	0.41

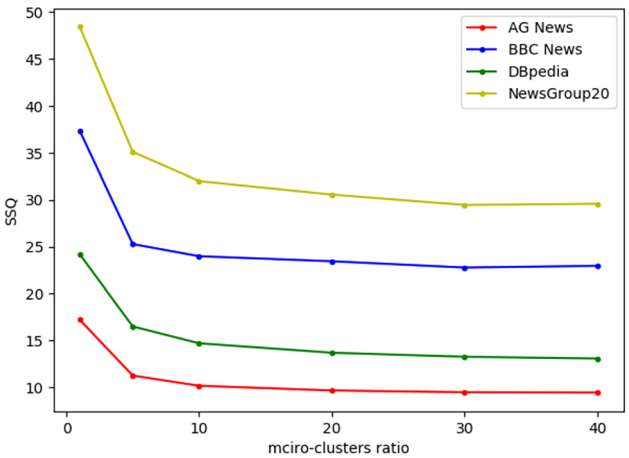
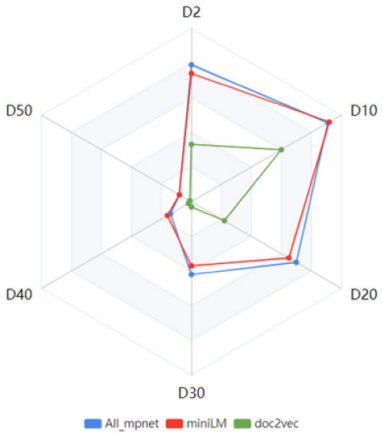
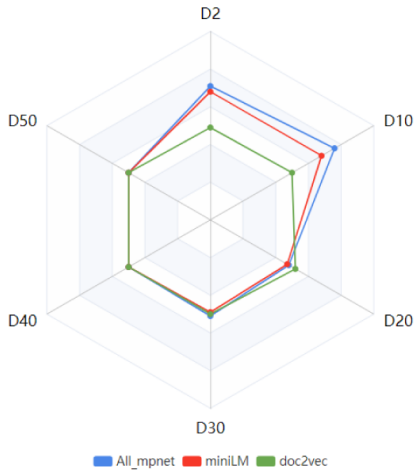


Fig. 4 Effect of the number of micro-clusters

Fig. 5 Radar chart of AGNews dataset

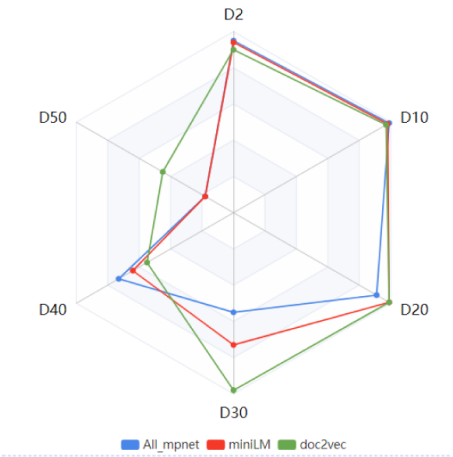


(a) Purity of AGNews.

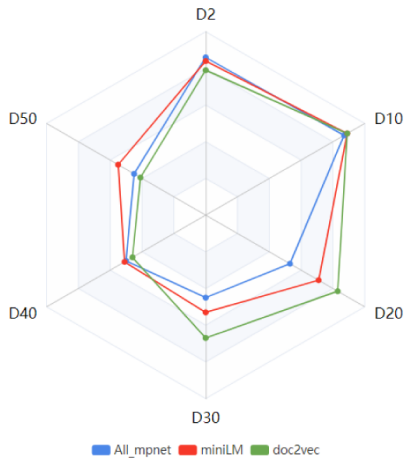


(b) Silhouette of AGNews

Fig. 6 Radar chart of BBC News dataset

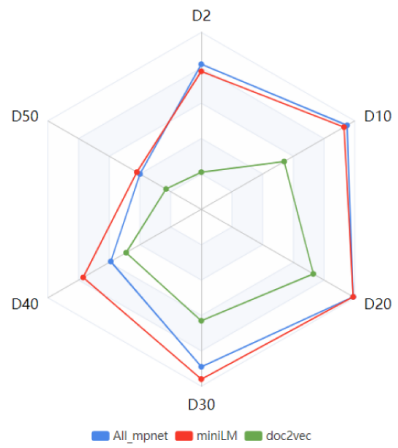


(a) Purity of BBC News

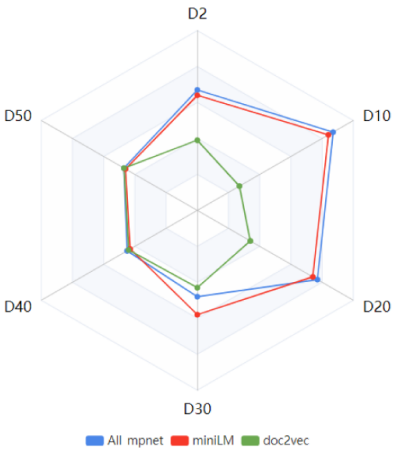


(b) Silhouette of BBC News

Fig. 7 Radar chart of DBPedia dataset

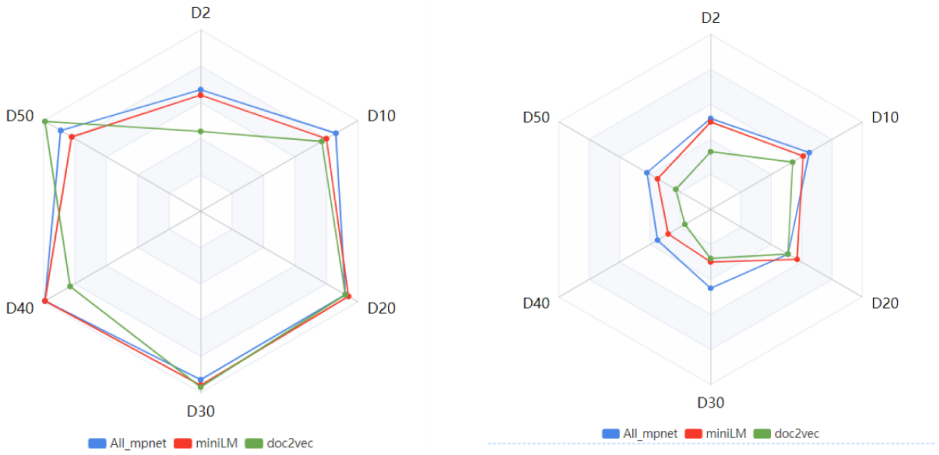


(a) Purity of DBpedia



(b) Silhouette of DBpedia

Fig. 8 Radar chart of Newsgroup dataset



(a) Purity of Newsgroup20

(b) Silhouette of Newsgroup20

BBC News and AGNews, with purity decreasing after 20 dimensions. However, for the Newsgroup20 dataset, we noticed that the purity fluctuates depending on the number of dimensions.

This can be explained by the nature of the dataset, which has a large number of classes, making high purity easier to achieve. We also observe that the results achieved using the all_mpnet model for text representation surpass those obtained with doc2vec and miniLM, both in terms of purity and silhouette scores.

Conclusion

In this paper, we proposed OMTStream, a novel approach for online clustering of massive text data streams that effectively addresses the challenges of high dimensionality, data sparsity, and evolving data distributions. By integrating text representation learning with an online incremental clustering model, OMTStream dynamically learns meaningful representations of streaming text while efficiently grouping similar textual data in a reduced-dimensional space. Our experimental validation on multiple textual datasets demonstrates that OMTStream outperforms classical methods in terms of Purity, SSQ, and Silhouette index, highlighting its superior clustering performance.

One of the key strengths of OMTStream is its ability to scale efficiently while maintaining high clustering quality. By leveraging semantic text representation, our method captures deeper contextual relationships in textual data, leading to more meaningful and coherent clusters compared to traditional methods. Additionally, the incremental clustering model enables OMTStream to adapt dynamically to changes in streaming data without requiring a full re-clustering process, ensuring that it remains effective even as data evolves. Another major advantage of OMTStream is its computational efficiency, achieved through dimensionality reduction, which reduces the impact of high-dimensional data on clustering performance.

Despite its advantages, OMTStream has certain limitations that present opportunities for future research. One of them is its sensitivity to hyperparameters in UMAP and CluStream, as the effectiveness of both dimensionality reduction and clustering largely depends on parameter tuning. Another limitation of our approach is the use of a single text representation method to model textual data. While BERT effectively captures contextual semantics, relying on a single representation method may not fully leverage the diverse aspects of text data.

Future Work

In future work, to enhance OMTStream's adaptability and robustness, we plan to explore a multi-view learning approach that incorporates multiple text representation techniques. Specifically, we intend to integrate BERT embeddings to capture rich semantic relationships, TF-IDF to reflect statistical term importance, and Latent Dirichlet Allocation (LDA) to extract thematic insights. These diverse representations will be combined using ensemble learning techniques to further improve clustering accuracy and robustness. Additionally, we aim to address diverse text characteristics, including varying text types such as short texts, long documents, and domain-specific data, as well as challenges related to data sparsity.

Appendix

See Tables 4, 5, 6 and 7.

Table 4 Stream clustering results of AGNews

H=200 K=80	OMTStream-allmpnet			OMTStream-miniLM			OMTStream-Doc2vec		
nb_dim	Pur	SSQ	Sil	Pur	SSQ	Sil	Pur	SSQ	Sil
D2	0.79	2.92	0.71	0.74	3.01	0.68	0.33	2.36	0.49
D10	0.91	12.99	0.76	0.92	9.67	0.68	0.60	22.80	0.50
D20	0.70	23.85	0.48	0.65	20.37	0.47	0.22	41.14	0.52
D30	0.42	35.86	0.51	0.37	50.04	0.49	0.03	69.74	0.50
D40	0.14	48.28	0.50	0.16	39.79	0.50	0.02	90.07	0.50
D50	0.08	55.95	0.50	0.08	44.48	0.50	0.01	102.33	0.50

Table 5 Stream clustering results of DBpedia

H=1000 K=280	OMTStream-allmpnet			OMTStream-miniLM			OMTStream-Doc2vec		
nb_dim	Pur	SSQ	Sil	Pur	SSQ	Sil	Pur	SSQ	Sil
D2	0.82	3.53	0.67	0.78	2.95	0.64	0.21	3.22	0.39
D10	0.95	16.42	0.87	0.93	13.70	0.84	0.54	25.28	0.27
D20	0.99	36.33	0.77	0.99	23.96	0.74	0.73	98.98	0.34
D30	0.89	37.35	0.48	0.96	40.82	0.58	0.63	166.24	0.43
D40	0.59	40.96	0.45	0.77	42.14	0.43	0.49	233.04	0.44
D50	0.40	48.85	0.47	0.42	58.29	0.46	0.23	274.10	0.47

Table 6 Stream clustering results of Newsgroup20

H=1000 K=400	OMTStream-allmpnet			OMTStream-miniLM			OMTStream-Doc2vec		
nb_dim	Pur	SSQ	Sil	Pur	SSQ	Sil	Pur	SSQ	Sil
D2	0.67	2.43	0.52	0.64	3.01	0.50	0.44	1.32	0.33
D10	0.86	20.51	0.65	0.80	30.56	0.61	0.77	33.01	0.54
D20	0.92	54.23	0.51	0.94	59.70	0.57	0.92	55.11	0.51
D30	0.93	95.15	0.45	0.96	113.86	0.30	0.97	108.30	0.28
D40	0.99	116.75	0.35	0.99	162.16	0.28	0.83	151.75	0.17
D50	0.89	143.41	0.42	0.82	187.77	0.35	0.99	181.84	0.23

Table 7 Stream clustering results of BBC News

H=200 K=100	OMTStream-allmpnet			OMTStream-miniLM			OMTStream-Doc2vec		
nb_dim	Pur	SSQ	Sil	Pur	SSQ	Sil	Pur	SSQ	Sil
D2	0.95	4.52	0.86	0.94	3.88	0.84	0.90	2.91	0.79
D10	0.99	30.80	0.87	0.98	23.44	0.89	0.97	16.83	0.89
D20	0.91	55.87	0.53	0.99	59.66	0.71	0.99	33.94	0.83
D30	0.55	98.86	0.45	0.73	79.19	0.53	0.98	56.42	0.67
D40	0.73	129.25	0.50	0.64	110.46	0.51	0.55	83.76	0.46
D50	0.18	159.97	0.45	0.18	145.50	0.55	0.45	96.08	0.41

Author Contributions Mrs Maha Ben-Fares: conceptualization, visualization, programming and experiences, result analysis, writing; Mr. Nistor Grozavu: conceptualization, supervision, formal analysis, writing-review and editing; Mrs Parisa Rastin and Mr. Pierre Holat: supervision.

Funding CIFRE Project.

Data Availability All the datasets used are open-source.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no Conflict of interest.

Research Involving Human and/or Animal Participants Not applicable.

Informed Consent Not applicable.

References

1. Ackermann MR, Märtens M, Raupach C, Swierkot K, Lammersen C, Sohler C. Streamkm++ a clustering algorithm for data streams. J Exp Algorithmics (JEA). 2012;17:1–2.
2. Aggarwal CC. A survey of stream clustering algorithms. In: Data clustering. Chapman and Hall/CRC; 2018. p. 231–58.
3. Aggarwal CC, Philip SY, Han J, Wang J. A framework for clustering evolving data streams. In: Proceedings 2003 VLDB conference. Elsevier; 2003. p. 81–92.
4. Aggarwal CC, Yu PS. A framework for clustering massive text and categorical data streams. In: Proceedings of the 2006 SIAM International Conference on Data Mining. SIAM; 2006. p. 479–483.
5. Barddal JP, Gomes HM, Enembreck F. Sncstream: A social network-based data stream clustering algorithm. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing; 2015. p. 935–940.
6. Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, Seidl T. Moa: Massive online analysis, a framework for stream classification and clustering. In: Proceedings of the First Workshop on Applications of Pattern Analysis. PMLR; 2010. p. 44–50
7. Cao F, Estert M, Qian W, Zhou A. Density-based clustering over an evolving data stream with noise. In: Proceedings of the 2006 SIAM International Conference on Data Mining. SIAM; 2006. p. 328–339.
8. Carnein M, Trautmann H. Optimizing data stream representation: an extensive survey on stream clustering algorithms. Bus Inf Syst Eng. 2019;61(3):277–97.
9. Dang XH, Lee V, Ng WK, Ciptadi A, Ong KL. An em-based algorithm for clustering data streams in sliding windows. In: International Conference on Database Systems for Advanced Applications. Springer; 2009. p. 230–235.
10. Greene D, Cunningham P. 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. p. 377–384
11. Guha S, Mishra N. Clustering data streams. In: Data stream management. NY: Springer; 2016. p. 169–87.
12. Jones KS. A statistical interpretation of term specificity and its application in retrieval. J Doc. 1972;28(1):11–21.
13. Kumar J, Shao J, Uddin S, Ali W. An online semantic-enhanced Dirichlet model for short text stream clustering. In: Proceedings of the 58th annual meeting of the association for computational linguistics; 2020. p. 766–776.
14. Lang K. 20 newsgroups dataset (empty), http://people.csail.mit.edu/jrennie/ 20Newsgroups/
15. Le Q, Mikolov T. Distributed representations of sentences and documents. In: International conference on machine learning. PMLR; 2014. p. 1188–1196
16. Luhn HP. A statistical approach to mechanized encoding and searching of literary information. IBM J Res Dev. 1957;1(4):309–17.
17. Nguyen HL, Woon YK, Ng WK. A survey on data stream clustering and classification. Knowl Inf Syst. 2015;45(3):535–69.
18. Reimers N, Gurevych I. 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 2019. https://arxiv.org/abs/1908.10084
19. Ren J, Ma R. Density-based data streams clustering over sliding windows. In: 2009 Sixth international conference on fuzzy systems and knowledge discovery, vol. 5. IEEE; 2009. p. 248–252
20. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65.
21. Ruiz C, Menasalvas E, Spiliopoulou M. C-denstream: Using domain knowledge on a data stream. In: Discovery Science: 12th International Conference, vol. 12, DS 2009, Porto, Portugal, October 3–5, 2009. Springer; 2009. p. 287–301
22. Silva JA, Faria ER, Barros RC, Hruschka ER, de Carvalho A, Gama J. Data stream clustering: a survey. ACM Comput Surv (CSUR). 2013;46(1):1–31.
23. Yin J, Chao D, Liu Z, Zhang W, Yu X, Wang J. Model-based clustering of short text streams. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining; 2018. p. 2634–2642.
24. Zhang T, Ramakrishnan R, Livny M. Birch: an efficient data clustering method for very large databases. ACM SIGMOD Record. 1996;25(2):103–14.
25. Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R, editors. Advances in neural information processing systems, vol. 28. Curran Associates, Inc; 2015.
26. Zhang X, Zhao JJ, LeCun Y. Character-level convolutional networks for text classification. In: NIPS; 2015.
27. Zhao Y, Karypis G. Criterion functions for document clustering: experiments and analysis. in Technical Report TR 01-40, Department of Computer Science, University of Minnesota; 2001.
28. Zhou A, Cao F, Qian W, Jin C. Tracking clusters in evolving data streams over sliding windows. Knowl Inf Syst. 2008;15(2):181–214.
29. Zubaroğlu A, Atalay V. Data stream clustering: a review. Artific Intell Rev. 2021;54(2):1201–36.