Methods to Investigate Concept Drift in Big Data Streams

Chapter · February 2018

DOI: 10.1007/978-981-10-6680-1_3

Nidhi Krail
Panjab University
5 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Veenu Mangat
44 PUBLICATIONS **314** CITATIONS
SEE PROFILE

Vishal Gupta
Panipat Institute Of Engineering & Technology
69 PUBLICATIONS **549** CITATIONS
SEE PROFILE

Methods to Investigate Concept Drift in Big Data Streams

Nidhi, Veenu Mangat, Vishal Gupta and Renu Vig

Abstract The explosion of information from various social networking sites, Web clickstream, information retrieval, customers' records, users' reviews, business transactions, network event logs, etc. Results in generating a continuous deluge of data at different rates, called streaming data. Organizing, indexing, analyzing, or mining hidden knowledge from such a data deluge becomes a critical functionality for a broad range of content analysis tasks that includes emerging topic detection, interesting content identification, user interest profiling, and real-time Web search. But managing such 'Big Data' becomes even more challenging when streaming data is taken for analyzing and producing results in real time. The streaming data may include numeric, categorical, or mixed value. Most of the current research has been done on numeric data streams by exploiting the statistical properties of the numeric data. But now categorical/textual data streams have also gained researchers' interest due to the high availability of data in textual format on the Internet. Applying classification for managing data streams is an unrealistic approach as not every incoming data has a class label. So, in such a case, for managing unlabeled data streams, a clustering technique is applied. One property that can affect the results of any clustering algorithm is concept drift. Therefore, detecting and managing concept drift over a period imposes a great challenge to better cluster analysis. This chapter provides an in-depth critique of various algorithms that have been introduced to handle concept drift in a real environment. A framework for examining concept drift in big data streams is also proposed.

Nidhi (✉) · V. Mangat · V. Gupta · R. Vig
UIET, Panjab University, Chandigarh, India
e-mail: nidhi789@pu.ac.in

V. Mangat
e-mail: vmangat@pu.ac.in

V. Gupta
e-mail: vishal@pu.ac.in

R. Vig
e-mail: renuvig@hotmail.com

Keywords Big data · Social networks · Social media · Clustering
Concept drifting

1 Introduction

The primary task of social networking sites like Facebook, Twitter, Snapchat, Instagram is to communicate and interact with a vast audience. Nowadays such sites play a significant role in disseminating information ranging from the entertainment industry, new medical science achievements, brand promotion, awareness among people, etc. Initially, this role was played by blogging sites, but reading and creating such lengthy blogs is a time-consuming task. Therefore, micro-blogging was introduced. Micro-blogging describes content in a concise and a meaningful way. Twitter is the most notable example of a micro-blogging site and is one of the greatest revolutions in social media. Here the user can post a single message up to 140 characters only, and these messages are well known as tweets [1]. According to the Twitter Web site (http://www.twitter.com), the broad coverage of this social network is confirmed by having 255 million monthly active users that post 500 million tweets per day [2]. An enormous amount of data that is being generated and shared across these micro-blogging sites serves as an excellent source of big data streams for analysis. Data shared on such sites is not confined to only textual data but also consists of photos, videos, gifs, URLs, etc., as content. Clustering is a widely used technique to organize such unlabeled data into homogenous groups. But implementation of clustering techniques for streaming data is very different from those for static data (i.e., data that does not change in the clustering process). The reasons for this are twofold. Firstly, it is hard to store an entire data stream once as data is arriving continuously, and secondly, due to the massive volume of streaming data, scanning data multiple times as it comes is not possible. Another major issue with streaming data is that the concepts behind the data evolve with time; therefore, discovering hidden concepts in data streams with time imposes an enormous challenge to cluster analysis [3, 4]. Therefore, identification and handling concept drift in 'Big Data' streams is a current area of interest [5]. For this, a learning system is required to detect and analyze the concept drift while producing results in a real time, i.e., results need to be temporally relevant and timely [3, 6]. In future, this work will enable linking concepts semantically to analyze topic convergence or divergence [7–9] and predicting coevolving events, if any, in the data streams. The work will also help in refining the results of clustering algorithm by reducing the clusters' count either by merging the clusters if there is any semantic relationship between clusters, or by declaring the clusters as outliers. The applications of this study are in the detection of events, product recommendation systems, campaign promotion, customer segmentation, etc. Many clustering algorithms that have been researched so far are mostly implemented for numeric data by exploiting the mathematical properties, e.g., by calculating the distance between data objects to form a cluster. Therefore, an effective clustering algorithm is required for textual streaming data while keeping semantic aspect in mind as very less amount of work has been done in this direction as compared to clustering static data and numeric data streams.

1.1 Social Media Analysis

Social media technologies are growing at a rapid rate, they are now considered to be a mainstream communication tool for much of the global population. Social media is a broad and continually evolving term but refers to Internet-based platforms, which enable users to connect, interact, and share information, ideas, and other content. They have become increasingly popular and numerous, with an estimated 1.5 billion users according to recent statistics (widely used examples include Facebook, Twitter, Snapchat, Instagram, and LinkedIn) with millions of users incorporating them into their daily routines. Recently, social networking sites have also been found to be utilized by the government for agenda setting, policy making, and to communicate new initiatives [4]. There are billions of active users on the social networking sites that are generating dynamic data in an enormous amount at different rates.

Twitter is one of the most well-known social media platforms, being characterized by providing a micro-blogging service where users can post text-based messages of up to 140 characters, known as tweets, mimicking the SMS (Short Message Service) messages [5, 6].

Twitter Entities: Entities provide metadata and additional contextual information about content posted on Twitter. Entities are always linked to the content they describe. In API v1.1, entities are returned wherever Tweets are found in the API [7] (Table 1).

1.2 Applications of Social Media Analysis

1. **Event Detection**: As content changes frequently with every second as new data arrives, it becomes necessary to detect the event of the data and to analyze topic convergence or divergence if any [8–10]. It also helps in predicting coevolving events.

Table 1 Twitter entities

Entity name	Description
Hashtag	For indexing information (#name)
User_Mention	For indexing users (@name)
Media	Consists photos, animated gifs, videos, etc.
URLs	URLs present in the tweets

2. **Recommendation Systems**: Such a system provides suggestions for users to buy a particular product by analyzing the user's profile, user's history, and information sharing [11].
3. **Campaign Promotion**: It deals with influencing people's behaviors/opinions/decisions about a particular thing or concept. It is thus important to discover such campaigns, their promoter accounts, and how the campaigns are organized and executed as it can uncover the dynamics of Internet marketing [12].
4. **Customer Segmentation**: By examining users' behavior, liking/disliking, communities they follow, customer values, products/stocks they purchased, etc., customers can be segregated into different groups [13, 14].
5. **Awareness Programme**: This enables promotion of health care facilities among public [15].

1.3 Introduction to Streaming Data

The rate, at which data is being generated nowadays with offline and online users, is increasing exponentially. Data that is being generated continuously, temporally ordered, fast changing, and massive in size is called streaming data. Real-time examples of streaming data from different fields are following:

1. **Business/E-commerce**: Online transactions of the users, stock market; online advertisements; data generated from mobile apps.
2. **Social Media**: Textual data from user's posts on Twitter, Facebook, etc.; sharing images or videos on social media; multimedia data.
3. **Medical Science**: Doctors make use of large amounts of time-sensitive data for serving patients, including results of lab tests, pathology reports, X-rays, and digital imaging.
4. **Telecommunication and Networks**: It includes data from sensors or cameras for traffic analysis; clickstream data; network monitoring of packets or security; GPS data; data from satellites; server logs.

1.4 General Architecture of Clustering Streaming Data

Most of the algorithms for processing streaming data work on the assumption that the class labels of the arriving data are available. However, this assumption is very impractical, especially in the case of social media analysis where the data or content depends heavily on millions of users. There can be some association or relation between the contents depending upon users' interestingness, or there can be no association. In such a case, labeling each incoming data is very expensive and time consuming; it also requires a lot of skilled labors to do so.

Therefore, for processing and analyzing streaming data without class labels, clustering techniques are used. But the approach for clustering streaming data is different from clustering static data (i.e., data that does not change in the clustering process) as the entire data stream is not available at a time. Therefore, clustering of streaming data includes two phases: offline and online processing of streaming data [16].

Online Phase: In this component, initial processing of streaming data is done in chunks, which produces micro-clusters by storing summaries/clusters of each processed data chunk. It also includes incremental computation and maintenance of the micro-clusters to keep the micro-clusters updated.

Offline Phase: This component produces macro-clusters by analyzing the micro-clusters on various users' constraints for example number of clusters, a query related to a particular time or topic, etc.

1.5 *Introduction to Concept Drift*

In a given time window, each data sample in the data stream is assigned to an individual cluster that further accounts for a concept. As new data records arrive, two cases can be possible: (1) the existing clusters either update themselves to accommodate new arriving data (if there is any similarity between the two concepts); or, (2) a new cluster is created (if two concepts are completely different than each other). In the second case, the difference in the concepts of existing and incoming data is what is known as concept drift.

1.5.1 Types of Concept Drift

1. Sudden (ABRUPT) Concept Drift: Sudden concept drift occurs when there is a dramatic change in the concepts. As shown in Fig. 1, at time *t*1, there are only two concepts "A" and "B," as time *t*2 occurs, only "B" and "C" concepts are prevailing. Concept "A" is entirely replaced by concept "C".

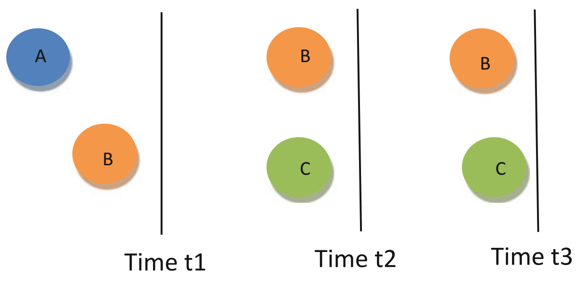

Fig. 1 Sudden concept drift

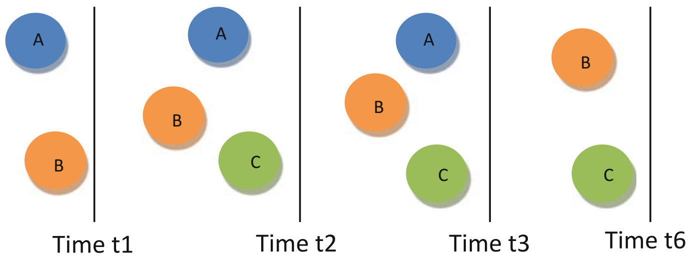

Fig. 2 Incremental concept drift

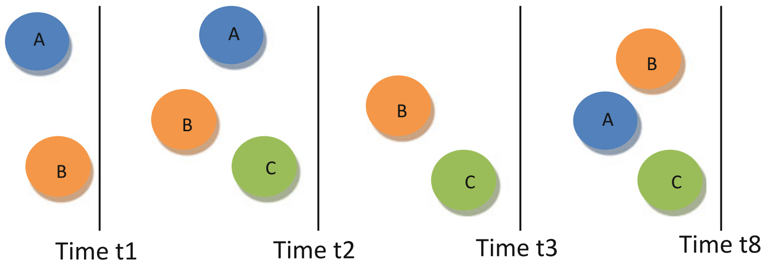

Fig. 3 Gradual concept drift

2. Gradual Concept Drift: In this concept drift, change in a concept occurs either gradually or incrementally over time. In incremental concept drift, concept "A" vanished from the site at time t6 and never occured again (as shown in Fig. 2), whereas in gradual concept drift, concept "A" is switching back and forth over time. As shown in Fig. 3, concept "A" is not present from time t3 to t7 but recurs at time t8.

But the major difficulty in dealing with concepts is identifying outliers in clusters as one may misinterpret outlier as concept drift by analyzing adjacent windows. At that time, it becomes necessary to distinguish between a concept drift and an outlier by analyzing all the succeeding and preceding windows.

1.5.2 Need to Detect Concept Drift in Streaming Data

Concept drift is the most undesirable yet prevalent property of streaming data as data streams are very unpredictable. Due to concept drift, the performance of mining techniques such as classification or clustering degrades as chances of misclassification increase. Therefore, it becomes necessary to identify such drifts in the data to get efficient results with accuracy.

2 Literature Review

In this chapter, the literature review is organized into three broad categories:

1. Study of clustering algorithms on categorical data streams
2. Study of detecting concept drift using unsupervised learning approach
3. Study of research done on social media analysis regarding concept drift.

2.1 Clustering Categorical Streaming Data

Undoubtedly, many clustering algorithms studied so far are mostly implemented for numerical data by exploiting various mathematical properties for calculating the distance between data objects to form a cluster. But, in a real environment, data consists of either categorical data or both numeric and textual data, e.g., Web clickstream data, information retrieval data, patients' records, customers' records. The field of clustering categorical streaming data has received very less attention as compared to applying clustering on static data and numerical data streams. In the following table, the conventional clustering algorithms for categorical/textual streaming data are mentioned with their key features. The latest research on clustering categorical data streams is also reviewed in this section (Table 2).

In [18], Wu proposed an algorithm for detecting outliers in categorical data streams. First, closed frequent patterns are discovered in each sliding window using HCI-MTree. Then, using weighted closed frequent pattern outlier factor

Table 2 Conventional clustering algorithms for categorical/textual streaming data [17]

Algorithms	Features
K-Modes; Fuzzy K-Modes	• Extended version of k-Means • Based on frequency method
CACTUS	• Consist of three phases: Summary (to compute candidate clusters); validation and clustering (to get actual clusters by validating the clusters formed)
ROCK	• Adaption of agglomerative hierarchical clustering • Each tuple is assigned separately to individual clusters; then clusters are merged by closeness between the tuples which is calculated by the sum of links between pairs of tuples
Squeezer	• Tuples are scanned one by one and are assigned/rejected to/by clusters on the basis of similarity
COOLCAT	• Entropy-based algorithm; clusters of similar entropy • Efficient for streaming data as it is based on incremental approach
CLOPE [8]	• Based on increasing height-to-width ratio of the cluster histogram • Efficient for large, high-dimensional transactional data, but requires little domain knowledge about dataset to control the count on clusters

(WCFPOF), outliers are detected which are stored in query indexed structure (QIS). To handle concept drift, the older categorical data are periodically replaced by newer ones. The accuracy achieved by the algorithm is equaled to Apriori algorithm.

Sora et al. [19] introduced a practical approach for clustering categorical data streams called FLoMSqueezer, an enhanced version of Squeezer algorithm (a clustering algorithm for categorical dataset). Squeezer provides high-quality clusters, but its performance degrades if the size of the histogram increases. The histogram is used to capture the distribution of values in the dataset. Therefore, by controlling the size of the histogram, the same method can be applied to categorical data streams, resulting in a new approach called FLoMSqueezer. The results show that proposed algorithm outperforms Squeezer algorithm in average clustering error and execution time parameters. The proposed algorithm also produces quality clusters with limited memory in data stream environment.

To reduce the dimensionality of the categorical dataset, entropy-based relevance index is computed for each attribute in the cluster to calculate its relevancy in the cluster [20].

Qin et al. [21] proposed a new information theory-based algorithm for clustering categorical data by computing mean gain ratio (MGR) of attributes and then selecting equivalence class on the chosen clustering attributes using entropy of the clusters. The proposed algorithm is compared with other information theory-based algorithms such as MMR (a rough set hierarchal algorithm), K-ANMI (K-means and mutual Information) and COOLCAT (Entropy based) on accuracy, adjusted Rand index (ARI) performance metrics. The results show that MMR achieves highest accuracy (0.931 for zoo dataset) and higher ARI (0.96 for zoo and cancer dataset) than other algorithms. The average running time of MGR is 6.67 s which is 21, 11.09, and 133.159 s for MMR, COOLCAT, and K-ANMI, respectively. In future work, MMR algorithm can be combined with K-ANMI for improving the performance of clusters.

Lenco et al. [22] introduced change detection algorithm for evolving categorical data streams in which summaries are extracted from the batches, and then Chebyshev's Inequality is used for detecting a change in those batches. The proposed algorithm is compared with [23] in which concept drift is detected by computing the difference between the two concepts using rough set theory; this may also help in measuring the velocity at which change is happening in the concepts. The proposed algorithm achieves an average accuracy of 75% for different batch sizes, i.e., 50, 100, 500, and 100. Lenco et al. [23] algorithm can detect the changes which were not identified. A high point of this approach is its ability to exploit historical information for decision making by adjusting the size of history window.

Li et al. [24] presented an integrated model for clustering categorical data streams that help in data labeling, change detection, and cluster evolving analysis. In this, three dissimilarity measures are proposed that are based on incremental entropy (for computing dissimilarity between point-cluster and cluster-cluster) and sample standard deviation (for computing dissimilarity between two cluster distributions). The algorithm is made to run on different sliding window sizes to analyze its efficiency on various datasets: zoo, soybean, dermatology, and DNA. The experimental results demonstrate that the labeling accuracy of the proposed algorithm is higher than [25–27] (other data labeling methods). For measuring concept detection performance, precision, and recall are calculated as follows:

$$\text{Precision} = \frac{\text{no\_of\_concept\_drifting\_windows\_correctly\_detected}}{\text{total\_no\_of\_concept\_drift\_windows\_detected}}$$

$$\text{Recall} = \frac{\text{no\_of\_concept\_drifting\_windows\_correctly\_detected}}{\text{actual\_no\_of\_concept\_drift\_windows\_present}}$$

The change detection algorithm of [24] is superior to [25], and somehow if the performance degrades then, it is due to the use of k-modes algorithm in the proposed work.

Talistu et al. [28] suggested a new approach to cluster data streams in a distributed manner which reduces the burden of a centralized processor to process the whole data streams by itself. In this approach, first local online summaries are generated at each local location using hierarchical clustering technique; then these summaries are distributed to every other node present in the system to get a global view of the data using gossip protocol. This collection and summarization of the data streams may help in analyzing the patterns over time and also in tracking cluster evolution, if any. After distributing local online summaries, nodes may get many duplicate summaries which can give a false interpretation of the data streams. Therefore, to avoid such false results, the merge-and-reduce algorithm is applied to remove the replicated summaries. This algorithm can also be used at hierarchical clustering phase to get the smaller sets of representative data points. For performance analysis, the average squared error (ASE) is computed for evaluating the quality of the clusters and scalability is analyzed by calculating the effect on cluster quality as numbers of nodes in the system increases. Gossip protocol does not guarantee delivering local online summaries among nodes; this could be a limitation when some nodes in the system increase plus the merge-and-reduce algorithm is not efficient in detecting and deleting the duplicate summaries at each node. The experimental results demonstrate that the value of ASE increases and scalability performance of the system decreases with the increase in the number of nodes.

Xhafa et al. [29] discussed one of the issues of processing big data streams, i.e., streaming consistency. Streaming consistency is the desirable property for some of the applications where the time of occurring events plays a significant role in generating results; in that scenario, it becomes necessary to preserve the order of the events in the output as in the input streams. In the paper, Yahoo!S4 is implemented for processing real-time data streams generating from FlightRadar24. The implementation showed that the average time for updating the flight status is 4 s which is a very practical result but still several inconsistencies are observed in the streaming due to the heterogeneous nature of the clusters.

A novel approach based on consensus or trade-off is introduced by Erdi and Gil [30] to aggregate atomic similarity semantic measures using fuzzy logic (consensus) and then applying a high degree of trade-off to get the overall score. To aggregate the similarity measures between two concepts, aggregation function (if-else rule) is developed. The significant result of 0.85 is achieved by the proposed method than other standard measures. A second experiment is conducted to measure the degree of similarity between the terms by exploiting the search logs of Google search engine using following measures: midrange, quadratic mean, arithmetic mean, maximum, minimum median, and normalized Google distance. In this case also, the proposed algorithm outperforms other measures by achieving 0.64 accuracy.

Rehioui et al. [31] provided an improved version of density-based technique (DENCLUE-IM) for clustering big data streams. The hill climbing step of standard DENCLUE had high execution time and produced poor clustering results. Therefore, in the improved version, this step is modified from calculating density attractor at each step to finding an equivalent variable to density attractor that will represent all points, hence result in reducing the computations and increasing the clustering efficiency. To evaluate the effectiveness of the proposed method on the following datasets: page blocks, spambase and cloud services, Dunn index (intra-cluster dissimilarity), Davies–Bouldin index (inter-cluster dissimilarity), execution time, and clustering accuracy are taken as performance measures. Following results are obtained by the proposed algorithm (Table 3).

For cloud services dataset, clustering accuracy cannot be computed as labeling data is not available. The proposed algorithm provides acceptable results as compared to DENCLUE but remarkable results are achieved regarding execution time. The proposed algorithm is taking 12 times less execution time than DENCLUE for page block dataset; also for classifying cloud services, DENCLUE takes approx. 32 h where DENCLUE-IM takes only 28 min.

Laohakiat et al. [32] developed a clustering algorithm for high-dimensional data by incorporating dimension reduction step in the clustering framework. The algorithm performs dimension reduction at the arrival of each datum by finding local subspace using unsupervised linear discriminant analysis (LDA) method where classes are replaced by clusters and each class mean is replaced by the center of each cluster. Normalized mutual information matrix (NMI), Rand index (RI), adjusted Rand index, (AR) and Hubert's index (HI) are taken as performance indices. KDD CUP 99 and forest cover type datasets are used to examine the performance of the algorithm in comparison with other streaming algorithms named DenStream, HDDStream, and HPStream. The window size of 10,000 data instances

Table 3 Results of DENCLUE-IM [31]

	Page blocks	Spambase	Cloud services
Dunn index	0.693	0.831	0.899
Davies–Bouldin index	0.412	1.041	1.262
Execution time (in minutes)	5.479	27.54	1657.508
Clustering accuracy	0.911	0.701	–

and 2000 data instances is taken for KDD CUP 99 dataset and forest cover type dataset, respectively. The experimental results show that for KDD CUP 99, 0.86, 0.87, 0.97, and 0.94 values are obtained for NMI, RI, AR, and HI, respectively, which is much higher than the performance indices values of other clustering algorithms taken for comparison. Similarly, for forest cover type dataset, proposed algorithm yields better performance than DenStream, HDDStream, and HPStream. Even the runtime of the algorithm is relatively lower than other clustering algorithms. For future work, this work can also be extended for categorical clustering attributes.

The literature review of clustering categorical data streams indicates that almost all the clustering algorithms exploit only one property to cluster the categorical data streams, i.e., computing the distance between the contents of two consecutive windows of the data stream to measure the similarity between two adjacent windows. No algorithm has utilized or discussed the semantic relation between the contents of windows to cluster the evolving data streams. The semantic relations, if considered, can help in reducing clusters' count in the result, hence reducing the execution time, computational cost, memory usage and may lead to better prediction of the content of next incoming data instances (Table 4).

2.2 Detecting Concept Drift in Categorical/Textual Data Streams

To handle concept drift in any data stream with efficiency, classification is the most common approach. But the primary requirement of the classification task is class labels which are not always available in the streaming data. And assigning class labels manually to the incoming data streams is very time consuming and requires a lot of skilled workforce. Therefore, in this section, different clustering or semi-supervised techniques are studied to detect concept drift depending on whether the researcher has knowledge about the domain of the incoming data or is completely unaware of their contents.

Barddal et al. [33] focused on a different kind of concept drift called feature drift occurring in the streaming data. Feature drifts are relevant to the learning algorithm as they result in dimensionality reduction and the decision can also be made based on these features. The experiment is conducted on spam corpus dataset, and the performance of the proposed method is evaluated regarding classifiers' accuracy, the runtime of the algorithm, and memory usage. The proposed landmark-based drift detection algorithm runs on different chunks created from the streaming data, these chunks are analyzed to determine relevant feature subsets present in chunks using discriminative factor, and then finally classifier is trained using these subsets. The results reveal that the proposed method produces interesting features subset with fast computation and less memory usage as compared to conventional algorithms such as information gain, correlation, and gain ratio.

Table 4 Summary of clustering categorical/textual data streams

Author	Primary task	Methodology	Dataset	Performance measure	Window size	Results
QunHui and ShiLong [18]	Detecting outliers from categorical data streams	Discovering closed frequent patterns in sliding window then measuring weighted factor	KDD CUP 99 network intrusion detection data set	Precision, running time of algorithm	50 K	Algorithm is sensitive to data size; precision achieved equals to Apriori algorithm
Sora et al. [19]	To overcome the scalability problem of Squeezer clustering algorithm	Sampling is introduced into the update histogram module	Mushroom Dataset (22 Attributes, 8124 tuples)	Execution time; accuracy	Varying window size	Proposed algorithm outperforms Squeezer algorithm in efficiency and execution time
Joel and Abel [20]	Clustering categorical data using entropy-based k-modes (EBK) algorithm	Measure the relevancy of each attribute in the dataset to reduce dimensionality by entropy-based relevance index	Vote (435 tuples, 17 attributes, two classes); mushroom (8124 tuples, 23 attributes, two classes); breast cancer (286 tuples, ten attributes, two classes); Soyabean (683 tuples, 36 attributes, 19 classes); genetic promoters (106 tuples, 58 attributes, two classes)	Accuracy, f-measure. Adjusted Rand index	No window size; static data	The accuracy results of EBK ranges from 70–88% for all the datasets. Even in f-measure and adjusted Rand Index metrics, the proposed algorithm outperforms the standard K-modes algorithms and its versions
Qin et al. [21]	Clustering categorical data	Using mean gain ratio and entropy	Zoo, congressional votes (Votes), Wisconsin breast cancer (breast cancer), mushroom, balance scale, car evaluation, chess, Hayes-Roth, and nursery	Accuracy, adjusted Rand index (ARI)	No sliding window; static data	Running time less than other three algorithms: MMR (a rough set hierarchal algorithm), K-ANMI (K-means and Mutual Information) and COOLCAT (Entropy based); MMR achieves highest 0.96 ARI

(continued)

Table 4 (continued)

Author	Primary task	Methodology	Dataset	Performance measure	Window size	Results
Lenco et al. [22]	Detecting change in evolving categorical data streams	Extracting summaries and applying Chebyshev's Inequality	Electricity, Forest, Airlines, KDD99	Accuracy; percentage of change detected	Batch size: 50, 100, 500, 1000	Achieves average accuracy of 75%; also able to detect changes, not detected by [23]
Li et al. [24]	Data labeling, change detection, analyzing evolving clusters	Incremental entropy and standard deviation	Soybean, zoo, dermatology and DNA	Precision, recall (for measuring concept detection performance) and accuracy, time (for evaluating clustering efficiency)	Varying sliding window sizes	Superior to [25–27] in detecting concept drift and generating clusters
Talistu et al. [28]	Clustering data streams	Distributed processing, gossip protocol merge-and-reduce algorithm to manage summaries in online phase	2-D dataset consisting 1,920,000 data objects	Average squared error, scalability	Distributed approach	ASE increases and scalability performance decreases with increase in the number of nodes
Xhafa et al. [29]	Streaming consistency	Clustering approach	FlightRadar24	Average time to update flight status	Clusters	Average time of 4 s is achieved for updating flight status

Fast evolutionary algorithm for clustering (FEAC) of data streams is proposed by Silva et al. [34] that aims at relaxing the assumption of prior knowledge of a number of clusters to be used in the clustering algorithm. In the suggested method, the number of clusters is estimated from the first incoming data using silhouette width criterion, then these clusters are incrementally updated, discarding the previous clusters as new data arrives; or change is reflected in any cluster by Page-Hinkley method. KDD CUP 99 intrusion detection dataset, forest cover type, and localization data for person activity are taken for experimental results in chunks of 2000 data instances. The algorithm can capture the changes present in both intra-class and inter-class characteristics by providing best trade-off between accuracy and runtime. In future, this work can be extended to predict cluster changes.

Liu and Zio [35] suggested a new algorithm based on feature selection vector to detect recurring drift in the streaming data. The major drawback with the existing algorithms is the replacement of the old patterns with new ones in the updating procedure. As a result, when that same old pattern reoccurs, the model needs to relearn again and hence computational time increases. To overcome this drawback, online ensemble based on feature vector (OE-FV) is introduced to store all the past patterns and updating the submodels by their weights. The algorithm adaptively updates the ensemble with the arrival of new data and thus can learn new patterns on time. Unlike, other window-based or chunk-based approach where the model has to wait for sufficient amount of data to be available. The results prove that the proposed algorithm decreases the count of changes patterns, leading to a reduction in computational burden.

Bai et al. [36] introduced a new clustering algorithm for categorical data streams and a detection index for drifting concept. Here, the clusters are merged by computing the difference between the old clusters and new ones. If the difference is greater than the threshold value, then a new cluster is formed. Otherwise, clusters are merged. At the end of this iterative step, the final clusters are analyzed to capture the evolution in the trends with time. For detecting drifts, following two facts are considered between the last and the new cluster: (1) distribution variation (for computing cluster representative) and (2) certainty variation (to capture change direction). The performance indices used to evaluate the proposed algorithm are accuracy, effectiveness in detecting drifts, and scalability. The datasets used for testing are letters, DNA, nursery, and KDD CUP 99. The results indicate that the proposed algorithm produces valid clusters irrespective of the data distributions in comparison with Chen et al. [37] and Cao et al. [38] algorithm. The average precision, recall, and accuracy value of the proposed algorithm for all the three datasets are above 85%. To evaluate the results of detective index, precision, recall, and Euclidean distance are computed. Even, in this case, the proposed algorithm produced less number of clusters while detecting drifts in the data streams. The computational time of the proposed algorithm is much higher than other two algorithms because, in other two algorithms, clustering problem is seen as data labeling step that needs a single iteration whereas the proposed algorithm considers the clustering problem as an iterative learning, requiring many iterations for better results.

For detecting both local and global changes in the transactional data streams, Koh [39] presented an algorithm where transactions falling in a given window are represented as a tree. Spearman correlation is computed to check the homogeneity between the current and previous window. If the distance calculated is greater than the threshold value then global drift is detected. In a similar fashion, when the difference in support of any particular item in the current window and the previous window comes out to be greater than the threshold value, then local drift is detected. True detections, false alarms, and detection delay are computed to measure the performance of the algorithm on following datasets: Kosarak, Airlines, and BMS-POS. The results show that the proposed algorithm gives high true positive rate and very less false alarm rate for determining structural changes in the dataset, even with noise.

A novel ensemble model called dynamic clustering forest that employs clustering trees (CTs) for clustering textual streams was proposed in [40]. In training phase, first clustering trees (CTs) of incoming chunks of textual streams are built, then discriminative CTs are selected by comparing the accuracy weight of each CT with defined threshold value. Next, for classifying test instances, the creditability of each discriminative CT is computed, and finally, with the help of both the weights, i.e., accuracy and credibility weights, the test set is classified. Creditability is calculated by computing the similarity between the test instance and centroid of each CT. To evaluate the performance, the proposed algorithm is compared with accuracy weighed ensemble (AWE) and accuracy updated ensemble (AUE) algorithms. For the spam stream dataset, DCF achieves around 1–5% improvement in classification accuracy. DCF running time is also low indicating its applicability to real-world datasets.

Haque et al. [41] introduced a semi-supervised algorithm based on existing SAND algorithm (k-NN type) for detecting concept drift and concept evolution in data streams by computing classifier confidence score, i.e., association and purity. The confidence score is also used to select limited labeled data instances for labeling the current chunk. A difference in the confidence scores of two windows represents the occurrence of a concept drift. The change detection algorithm is invoked only when the value of confidence score is less than the threshold value, resulting in a reduction in execution time. The algorithm is tested on three real-time datasets: forest cover (FC), power supply (PS), and physical activity monitoring (PAM). The performance metrics used are misclassification error (error %), novel class instances misclassified as existing class (M-new), existing class instances misclassified as new classes (F-new), and F-score. The error % of the proposed algorithm is 3.55, 3.85, and 0.01 for FC, PS, and PAM, respectively. For FC dataset, the M-new, F-new, and F2 achieved by the proposed algorithm are 13.55, 2.13, and 0.71, respectively.

An unsupervised, distribution independent, online incremental-based algorithm is introduced for detecting concept drift in unlabeled data streams in Sethi et al. [42]. The critical blindspot cardinality (CBC) is analyzed for detecting drifts in the data streams. For each of the incoming data samples, CBC is computed that indicates whether the input sample belongs to a CBC region or not. If the drift is detected, then experts are asked to label the samples to retrain the classifier. The average accuracy achieved by the proposed methodology when tested on four real-world datasets is 92.05%. The algorithm can reduce false alarms by capturing only those changes that can affect the performance of the classifiers.

By exploiting the temporal relationship among data, four unsupervised concept drift detection algorithms are introduced in [43], namely (1) stable clustering concept drift detection (SCCDD), (2) cross recurrence concept drift detection (CRCDD), (3) SCCDD-decomposition, and (4) CRCDD-decomposition. Third and fourth algorithms are the extended versions of first and second algorithms, respectively, in which preprocessing of the data is done before applying the algorithm. In the first experiment, dendrograms of two successive windows are created and then analyzed using Gromov–Hausdorff distance to detect concept drift. In the second experiment, a matrix of recurrence is created from the data of two successive windows, and then this matrix is analyzed to detect recurrent patterns by computing the longest diagonal in the matrix. The results of each proposed algorithm are compared with conventional concept drift detection methods, i.e., Page-Hinkley test and cumulative sum. The following metrics are used to measure the performance: (i) missed detection rate (MDR); (ii) mean time to detection (MTD); (iii) mean time between false alarms (MTFA). Following are the experimental results of the algorithms (Table 5).

The results show that SCCDD performs remarkably better than other algorithms for detecting concept drifts in the data streams, even in the presence of noise.

Lughofer and Mouchaweh [44] work on merging and splitting clusters by computing Mahalanobis distance between the clusters. If the distance is greater than the threshold value, then a new cluster is created. Otherwise, parameters of existing clusters are updated. The algorithm works efficiently for high-dimensional data but has the limitation of over-clustering for complex clusters with arbitrary shape and inability to handle concept drifts.

A mechanism is proposed by Yang and Fong [45] to handle concept drifts by introducing the optimized node-splitting method in the learning tree. Hoeffding bound with weighted Naïve Bayes classifier (to handle biased data) is used to construct an incremental tree for incoming data streams. The experimental results show that the proposed mechanism requires less memory as the final decision tree is compact in size.

Table 5 Results of paper [43]

	MDR	MTD	MTFA
PHT	1.000	0.000	0.989
CUSUM	0.850	0.002	0.213
SCCDD	0.250	0.006	0.319
CRCDD	0.050	0.018	1.000
SCCDDd	0.300	0.001	0.548
CRCDDd	0.100	0.009	0.600

Wu et al. [46] proposed a semi-supervised classification algorithm for unlabeled data streams. First, an incremental decision tree is generated using arriving data streams. In parallel, unlabeled data instances are labeled using K-modes algorithm and concept drift is detected by measuring the deviation between previous and current concept clusters. The proposed algorithm outperforms other semi-supervised algorithms CVFDT and BagBest regarding classification accuracy by 30%. The algorithm is also flexible to a degree of noise in the presence of the major unlabeled data.

Approaches for detecting concept drift are very diverse as can be concluded from this section. Every researcher is exploring a new methodology to handle concept drift according to their dataset either by finding the similarity between the concepts semantically or computing the distance between two windows. Hence, proving that there is still a wide scope of developing an efficient algorithm for not only analyzing concept drift but also describing the reason for occurrence of the concept drift.

2.3 Research Done for Detecting Concept Drift on Social Media

In social media analysis, two cases can be possible (1) an outlier in a particular window can become a trend in all the succeeding windows; (2) a trend in a current window can be evolved or vanished in succeeding windows. These two cases can help in producing results that either predict popular topics by analyzing current messages or contents of social media or analyze topics that may revive or evolve itself over the time. For obtaining such results, concept drift needs to be detected and handled efficiently. This section includes the literature review of work done on social media data for detecting concept drift.

In [47], re-tweets count of a particular message is taken as the measure to capture the popularity on Twitter data. But this approach ignores the temporal correlation between the messages.

A formal popular multimedia detection algorithm in social media is described in [48] by tackling two classification problems. First, is predicting whether a particular message M of time T will be re-shared by the users in time $T + 1$, this is called *re-share classification.* And second is multiclass classification problem, called *popularity score classification*; in this, the prediction is made by calculating the popularity score of social multimedia. The features used in the prediction task are (i) information diffusion feature (this includes the information of a set of users who post a particular message consisting of multimedia M in time t; (ii) multimedia meta-information (extracting meta-information of multimedia shared in time t from the source). Euclidean and overlap metric is used to measure the distance between numerical values and categorical values, respectively. The proposed algorithm performs remarkably better than SVM and J48 in terms of accuracy and F-score for the classification task.

In order to predict future networks, two approaches are proposed in [49]: (1) unweighted approach and (2) weighted approach. In unweighted approach, if a pair of nodes is connected to each other in the current window, then there will be a higher probability of connecting common nodes in the future networks. But this probability decreases with the increase in the neighbors of the network. The results show that the prediction performance of the algorithm for the static network is much better than evolving networks. In weighted approach, weights are assigned to the nodes by analyzing the dominance of human behavior, the decrease in the dominance will also reduce the weights allocated to the nodes. This approach also performs well for the static network.

In this paper [50], a novel method is presented to extract and visualize events from social media streams. The concept map is created where nodes are represented as sender and words found in the tweets post. To enhance the concept map, events are time labeled, and semantic similarity is computed between the nodes using Lin similarity. The method can extract the concept drift occurs in the events.

Miller et al. [51] implemented a combination of StreamKM++ and DenStream clustering algorithm for detecting spam in Twitter streams and also studied the effect of StreamKM++ and DenStream clustering algorithms individually on Twitter streams. The results prove that StreamKM++ achieves 99% recall and a 6.4% false positive rate and DenStream produces 99% recall and a 2.8% false positive rate, whereas the proposed algorithm, i.e., a combination of these algorithms reaches 100% recall and a 2.2% false positive rate.

To identify topic-specific post in Twitter streams, adjusted information gain (AIG) index is proposed and compared with other existing term scoring indexes such as IDF, Dice, Jaccard, and TF in [52]. The results show that 8–40% of the area under ROC curve that signifies true positive rates versus false positive rates when AIG is used as term scoring index.

Malik et al. [53] introduce two models: (i) to group related tweets into automatically generated topics (ii) an interactive visualization tool called TopicFlow to display similar topics in a specific time and identifying topic emergence or convergence or divergence over a given period. The proposed algorithm works as follows: (1) partition incoming tweets into bins, (2) run an unsupervised clustering algorithm called latent Dirichlet allocation (LDA) on each bin, (3) align topic into four cases: emerging, ending, continuing, and standalone, and (4) displaying results in visualization tool. The evaluation shows that TopicFlow is very useful in capturing the frequency of a particular topic over the given period and also provide details on demand to users through hovering over the extracted topic.

Context free method or variable length Markov model [54] is a model to predict the next symbol in a stream by observing the set in the preceding symbols, instead of relying on distance measure methods. This method is also effective for mixed data types.

A distributed and incremental temporal model for extracting topic from massive micro-blogs data streams called bursty event detection (BEE+) is proposed in [55], and the same is implemented on spark engine for real-world applications. BEE+ preserves the latent semantic indices by processing the post-stream incrementally to track the topic drifting of events over time. The algorithm is compared with TwitterMonitor (TM) to detect bursty events. Weibo dataset of 6,360,125 posts is taken for the experiment. The results show that for top-10 results, TM and BEE+ achieve 50 and 70% precision, respectively. TM also not able to group the keywords based on the temporal information of events whereas BEE+ does and hence can detect maximum concept drift in the streams.

For solving infrequent and semantic gap problem in the data streams, an algorithm is developed in which DBpedia is used as a knowledge base to link the data semantically. And for resolving word disambiguation problem, for each tweet, a graph is constructed based on the graph centrality theory. The proposed algorithm is compared with spotlight, a publicly available NEL tool. The proposed algorithm and spotlight achieve 60 and 51% F-score, respectively [56].

An improvised version of SFPM (Soft Frequent Pattern Mining) to recognize real-time social events in Twitter streams is introduced in [57]. In this paper, the new term selection algorithm is also developed that selects not only relevant keywords from the current window but also identifies the emerging terms from the current window to handle concept drift. Experiments are conducted on the posts related to FIFA World Cup 2014. The proposed framework outperforms in all the performance indices such as topic recall, keyword precision, and keyword recall, in comparison with standard SFPM version.

In paper [58], DBpedia is used as a knowledge base to link twitter entities based on graph centrality with the assumption that entities present in the topic-specific tweets are related to each other. The results show that the proposed algorithm achieves 69.5% F-score in comparison to TAGME model that attains only 46.8% F-score. In a similar way, to classify tweets related to specific events, semantic relations between the Twitter entities are exploited by again using DBpedia Ontology as a knowledge base. The algorithm [59] works in three main steps: (i) entity detection from the tweets; (ii) extracting relevant features related to the specific event; (iii) calculating entity score and class similarity to avoid word ambiguities.

3 Proposed Framework for Detecting Concept Drift on Social Media

From the literature review, following two significant research gaps are identified:

- More efficient clustering algorithm is required for analyzing concept drift on big data streams by determining the strength of association between data streams and correlating them.
- There is a need to explore other means of trusted sources of data and various content-based features to find the actual context of data, to identify ambiguities or synonyms in the data streams.

3.1 Proposed Framework

The proposed framework for detecting and handling concept drift in streaming data is divided into the following phases:

Phase 1: Data Collection
Collecting data from various online sources such as Twitter, Web sites, news articles.

Phase 2: Data Clustering and Labeling

2.1 Dividing data streams into windows.
2.2 Applying appropriate clustering algorithm on a window to group the data into clusters.
2.3 Labeling clusters to get knowledge about the concept hidden in that cluster.
2.4 Steps 2.2 and 2.3 are repeated for each window of the streaming data.

Phase 3: Detection of Concept Drift
Clusters of two adjacent windows are then analyzed for the following tasks: (i) to identify concept drift; (ii) to find any relationship between the concepts of two windows; (iii) to analyze coevolving events, if any.

For analyzing clusters, graph edge weight technique can be implementing by using content-based features of the streaming data, entities of social networking sites, and different data resources such as Web sites, new articles for making a decision.

Phase 4: Evaluation Phase
The performance of the algorithm can be evaluated by various performance metrics such as accuracy, precision, recall and execution time, classification, or clustering error.

4 Conclusion

The primary problem faced while clustering streaming data is the concept drift. The state-of-the-art clustering techniques and concept drift detection techniques mainly work on exploiting statistical features of data that often produce too many clusters in a final output. However, the clusters' count can be reduced by merging clusters that are semantically similar.

This paper highlights the recent research on clustering categorical data streams and concept drift detection techniques. From the literature review, it can be concluded that clustering algorithms which handle concept drift are often lacking in some major performance parameters such as computational cost, execution/running time of the algorithm, the quick response to the query, varied data density. Even, for exploiting semantic similarity between the clusters, the assurance of any knowledge-based resources such as Web sites, new articles or proper utilization of social networking sites entities is also required to validate the results.

Hence, for analyzing and mining any hidden knowledge from the streaming data efficiently and effectively, it becomes necessary to develop a system that can tackle the problem of concept drift using conceptual features. Several applications of such a system can be: identifying new emerging trends, top news, thematic categorization, top-influencers, queries suggestion, etc.

References

1. Zhang, B., Qin, S., Wang, W., Wang, D., & Xue, L. (2016). Data stream clustering based on fuzzy C-mean algorithm and entropy theory. *Journal of Signal Processing, 126,* 111–116.
2. Lifna, C., & Vijaylakshmi, M. (2015). Identifying concept drifts in Twitter streams. In *International Conference on Advanced Computing Technologies and Applications* (pp. 86–94).
3. Xhafa, F., Naranjo, V., Barolli, L., & Takizawa, M. (2015). On Streaming Consistency of Big Data Stream Processing in Heterogenous Clusters. In *18th IEEE International Conference on Network-Based Information Systems* (pp. 476–482).
4. Schnitzler, K., Davies, N., Ross, F., & Harris, R. (2016). Using Twitter™ to drive research impact: A discussion of strategies, opportunities and challenges. *International Journal of Nursing Studies, 59,* 15–26.
5. Costa, J., Silva, C., Antunes, M., & Ribeiro, B. (2014). Concept Drift Awareness in Twitter Streams. In *13th IEEE International Conference on Machine Learning and Applications* (pp. 294–299).
6. Wang, Y., Liu, J., Huang, Y., & Feng, X. (2016). Using Hashtag graph-based topic model to connect semantically-related words without co-occurrence in microblogs. *IEEE Transactions on Knowledge and Data Engineering, 28*(7), 1919–1933.
7. Entities—Twitter Developers. (2017). In Dev.twitter.com. https://dev.twitter.com/overview/api/entities. Accessed May 8, 2017.
8. Eskandari, S., & Javidi, M. (2016). Online streaming feature selection using rough sets. *International Journal of Approximate Reasoning, 69,* 35–57.
9. Li, J., Tai, Z., Zhang, R., Yu, W., & Liu, L. (2014). Online bursty event detection from microblog. In *IEEE/ACM 7th International Conference on Utility and Cloud Computing* (pp. 865–870).
10. Adedoyin-Olowe, M., Gaber, M., Dancausa, C., Stahl, F., & Gomes, J. (2016). A rule dynamics approach to event detection in Twitter with its application to sports and politics. *Expert Systems with Applications, 55,* 351–360.
11. Villanueva, D., González-Carrasco, I., López-Cuadrado, J., & Lado, N. (2016). SMORE: Towards a semantic modeling for knowledge representation on social media. *Science of Computer Programming, 121,* 16–33.
12. Li, H. (2014). Detecting campaign promoters on Twitter using Markov random fields. In *IEEE International Conference on Data Mining* (pp. 290–299).
13. Kuo, R., Mei, C., Zulvia, F., & Tsai, C. (2016). An application of a meta-heuristic algorithm-based clustering ensemble method to APP customer segmentation. *Neurocomputing, 205,* 116–129.
14. Wang, B., Miao, Y., Zhao, H., Jin, J., & Chen, Y. (2016). A biclustering-based method for market segmentation using customer pain points. *Journal of Engineering Applications of Artificial Intelligence, 47,* 101–109.

15. Giannitsioti, E., Athanasia, S., Plachouras, D., Kanellaki, S., Bobota, F., Tzepetzi, G., et al. (2016). Impact of patients' professional and educational status on perception of an antibiotic policy campaign: A pilot study at a university hospital. *Journal of Global Antimicrobial Resistance, 6,* 123–127.
16. Han, J., Kamber, M., & Pei, J. (2011). *Data mining* (3rd ed.). Amsterdam: Elsevier/Morgan Kaufmann.
17. He, Z., Xu, X., & Deng, S. (2011). Clustering categorical data streams. *Journal of Computational Methods in Sciences and Engineering, 11*(4), 185–192.
18. Wu, Q., & Ma, S. (2011). Detecting outliers in sliding window over categorical data streams. In *Eighth International Conference on Fuzzy Systems and Knowledge Discovery* (pp. 1663–1667).
19. Sora, M., Roy, S., & Singh, I. (2011). FLoMSqueezer: An effective approach for clustering categorical data stream. *International Journal of Computer Science Issues, 8*(6), 1.
20. Carbonera, J., & Abel, M. (2014). An entropy-based subspace clustering algorithm for categorical data. In *IEEE 26th International Conference on Tools with Artificial Intelligence* (pp. 272–277).
21. Qin, H., Ma, X., Herawan, T., & Zain, J. (2014). MGR: An information theory based hierarchical divisive clustering algorithm for categorical data. *Knowledge-Based Systems, 67,* 401–411. https://doi.org/10.1016/j.knosys.2014.03.013.
22. Lenco, D., Bifet, A., Pfahringer, B., & Poncelet, P. (2014). Change detection in categorical evolving data streams. In *29th Annual ACM Symposium on Applied Computing* (pp. 792–797).
23. Cao, F., & Huang, J. Z. (2013). A concept-drifting detection algorithm for categorical evolving data. In J. Pei, V. S. Tseng, L. Cao, H. Motoda & G. Xu (Eds.), *Advances in knowledge discovery and data mining*. *PAKDD* 2013. *Lecture notes in computer science* (vol. 7819). Berlin, Heidelberg: Springer.
24. Li, Y., Li, D., Wang, S., & Zhai, Y. (2014). Incremental entropy-based clustering on categorical data streams with concept drift. *Knowledge-Based Systems, 59,* 33–47. https://doi.org/10.1016/j.knosys.2014.02.004.
25. Chen, H. L., Chen, M. S., & Lin, S. C. (2009). Catching the trend: A framework for clustering concept-drifting categorical data. *IEEE Transactions on Knowledge Data Engineering, 21*(5), 652–665.
26. Cao, F., Liang, J., Bai, L., Zhao, X., & Dang, C. (2010). A framework for clustering categoricaltime-evolving data. *IEEE Transactions on Fuzzy System, 18*(5), 872–882.
27. Cao, F., & Liang, J. (2011). A data labeling method for clustering categorical data. *Expert Systems with Applications, 38,* 2381–2385. https://doi.org/10.1016/j.eswa.2010.08.026.
28. Talistu, M., Moh, T. S., & Moh, M. (2015). Gossip-based spectral clustering of distributed data streams. In *International Conference on High Performance Computing and Simulation* (pp. 325–333). https://doi.org/10.1109/HPCSim.2015.7237058.
29. Xhafa, F., Naranjo, V., Barolli, L., & Takizawa, M. (2015). On streaming consistency of big data stream processing in heterogenous clusters. In *18th International Conference on Network-Based Information Systems* (pp. 476–482).
30. Martinez-Gil, J. (2016). CoTO: A novel approach for fuzzy aggregation of semantic similarity measures. *Cognitive Systems Research, 40,* 8–17. https://doi.org/10.1016/j.cogsys.2016.01.001.
31. Rehioui, H., Idrissi, A., Abourezq, M., & Zegrari F. (2016). DENCLUE-IM: A new approach for big data clustering. In *7th International Conference on Ambient Systems, Networks and Technologies* (pp. 560–567).
32. Laohakiat, S., Phimoltares, S., & Lursinsap, C. (2017). A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction. *Information Sciences, 381,* 104–123. https://doi.org/10.1016/j.ins.2016.11.018.
33. Barddal, J., Gomes, H., Enembreck, F., & Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software, 127,* 278–294. https://doi.org/10.1016/j.jss.2016.07.005.

34. Andrade Silva, J., Hruschka, E., & Gama, J. (2017). An evolutionary algorithm for clustering data streams with a variable number of clusters. *Expert Systems with Applications, 67,* 228–238. https://doi.org/10.1016/j.eswa.2016.09.020.
35. Liu, J., & Zio, E. (2016). A SVR-based ensemble approach for drifting data streams with recurring patterns. *Applied Soft Computing, 47,* 553–564. https://doi.org/10.1016/j.asoc.2016.06.030.
36. Bai, L., Cheng, X., Liang, J., & Shen, H. (2016). An optimization model for clustering categorical data streams with drifting concepts. *IEEE Transactions on Knowledge and Data Engineering, 28,* 2871–2883. https://doi.org/10.1109/tkde.2016.2594068.
37. Chen, H.-L., Chen, M.-S., & Lin, S.-C. (2009). Catching the trend: A framework for clustering concept-drifting categorical data. *IEEE Transactions on Knowledge and Data Engineering, 21,* 652–665. https://doi.org/10.1109/tkde.2008.192.
38. Cao, F., Liang, J., Bai, L., Zhao, X., & Dang, C. (2010). A framework for clustering categorical time-evolving data. *IEEE Transactions on Fuzzy Systems, 18,* 872–882. https://doi.org/10.1109/tfuzz.2010.2050891.
39. Koh, Y. S. (2016). CD-TDS: Change detection in transactional data streams for frequent pattern mining. In *International Joint Conference on Neural Networks* (pp. 1554–1561). https://doi.org/10.1109/IJCNN.2016.7727383.
40. Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R., & Liu, F. (2016). Dynamic clustering forest: An ensemble framework to efficiently classify textual data stream with concept drift. *Information Sciences, 357,* 125–143. https://doi.org/10.1016/j.ins.2016.03.043.
41. Haque, A., Khan, L., Baron, M., Thuraisingham, B., & Aggarwal, C. (2016). Efficient handling of concept drift and concept evolution over Stream Data. In *IEEE 32nd International Conference on Data Engineering* (pp. 481–492).
42. Sethi, T. S., Kantardzic, M., & Arabmakki, E. (2016). Monitoring classification blindspots to detect drifts from unlabeled data. In *IEEE 17th International Conference on Information Reuse and Integration* (pp. 142–151).
43. da Costa, F., Rios, R., & de Mello, R. (2016). Using dynamical systems tools to detect concept drift in data streams. *Expert Systems with Applications, 60,* 39–50. https://doi.org/10.1016/j.eswa.2016.04.026.
44. Lughofer, E., & Mouchaweh, M. S. (2015). Autonomous data stream clustering implementing split-and-merge concepts—Towards a plug-and-play approach. *Information Sciences, 304,* 54–79. https://doi.org/10.1016/j.ins.2015.01.010.
45. Yang, H., & Fong, S. (2015). Countering the concept-drift problems in big data by an incrementally optimized stream mining model. *Journal of Systems and Software, 102,* 158–166. https://doi.org/10.1016/j.jss.2014.07.010.
46. Wu, X., Li, P., & Hu, X. (2012). Learning from concept drifting data streams with unlabeled data. *Neurocomputing, 92,* 145–155. https://doi.org/10.1016/j.neucom.2011.08.041.
47. Hong, L., Dan, O., & Davison, B. D. (2011). Predicting popular messages in Twitter. In *ACM International Conference on World Wide Web*(WWW).
48. Li, C., Shan, M., Jheng, S., & Chou, K. (2016). Exploiting concept drift to predict popularity of social multimedia in microblogs. *Information Sciences, 339,* 310–331. https://doi.org/10.1016/j.ins.2016.01.009.
49. Shang, K., Yan, W., & Small, M. (2016). Evolving networks—Using past structure to predict the future. *Physica A: Statistical Mechanics and its Applications, 455,* 120–135. https://doi.org/10.1016/j.physa.2016.02.067.
50. Lipizzi, C., Dessavre, D., Iandoli, L., & Marquez, J. (2016). Social media conversation monitoring: Visualize information contents of Twitter messages using conversational metrics. *Procedia Computer Science, 80,* 2216–2220. https://doi.org/10.1016/j.procs.2016.05.384.
51. Miller, Z., Dickinson, B., Deitrick, W., Hu, W., & Wang, A. (2014). Twitter spammer detection using data stream clustering. *Information Sciences, 260,* 64–73. https://doi.org/10.1016/j.ins.2013.11.016.

52. Karunasekera, S., Harwood, A., Samarawickrama, S., Ramamohanrao, K., & Robins, G. (2014). Topic-specific post identification in microblog streams. In *IEEE International Conference on Big Data* (pp. 7–13). https://doi.org/10.1109/BigData.2014.7004416.
53. Malik, S., Smith, A., Hawes, T., Papadatos, P., Li, J., Dunne, C., et al. (2013). TopicFlow: Visualizing topic alignment of Twitter data over time. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 720–726). https://doi.org/10.1145/2492517.2492639.
54. Jiang, W., & Brice, P. (2009). Data stream clustering and modeling using context-trees. In *6th IEEE International Conference on Service Systems and Service Management* (pp. 932–937).
55. Li, Wen J., Tai, Z., Zhang, R., & Yu, W. (2015). Bursty event detection from microblog: A distributed and incremental approach. *Concurrency and Computation: Practice and Experience, 28*(11), 3115–3130.
56. Kalloubi, F., Nfaoui, E. H., & Beqqali, O. El. (2014). Named entity linking in microblog posts using graph-based centrality scoring. In *9th International Conference on Intelligent Systems: Theories and Application* (pp. 501–506). https://doi.org/10.1109/SITA.2014.6847286.
57. Gaglio, S., Re, G., & Morana, M. (2015). Real-time detection of Twitter social events from the user's perspective. In *IEEE International Conference on Communications* (*ICC*) (pp. 1207–1212).
58. Kalloubi, F., Nfaoui, E., & Beqqali, O. (2014). Graph-based tweet entity linking using DBpedia. In *IEEE/ACS 11th International Conference on Computer Systems and Applications* (*AICCSA*) (pp. 501–506). https://doi.org/10.1109/AICCSA.2014.7073240.
59. Kumar, N., & Muruganantham, D. (2016). Disambiguating the Twitter stream entities and enhancing the search operation using DBpedia ontology. *International Journal of Information Technology and Web Engineering, 11*(2), 51–62. https://doi.org/10.4018/IJITWE.2016040104.