AIS-Clus: A Bio-Inspired Method for Textual Data Stream Clustering

Amal Abid*, Salma Jamoussi[†] and Abdelmajid Ben Hamadou[‡]

*Multimedia, Information Systems and Advanced
Computing Laboratory (MIRACL) Digital Research Center of Sfax
CRNS, Sfax University, Sfax Technopark, Sfax, Tunisia
abidamal90@gmail.com

[†]salma.jamoussi@isims.usf.tn

[‡]abdelmajid.benhamadou@isimsf.rnu.tn

Received 30 December 2018

Revised 15 March 2019

Accepted 17 March 2019

Published 26 April 2019

The spread of real-time applications has led to a huge amount of data shared between users. This vast volume of data rapidly evolving over time is referred to as data stream. Clustering and processing such data poses many challenges to the data mining community. Indeed, traditional data mining techniques become unfeasible to mine such a continuous flow of data where characteristics, features, and concepts are rapidly changing over time. This paper presents a novel method for data stream clustering. In this context, major challenges of data stream processing are addressed, namely, infinite length, concept drift, novelty detection, and feature evolution. To handle these issues, the proposed method uses the Artificial Immune System (AIS) meta-heuristic. The latter has been widely used for data mining tasks and it owns the property of adaptability required by data stream clustering algorithms. Our method, called AIS-Clus, is able to detect novel concepts using the performance of the learning process of the AIS meta-heuristic. Furthermore, AIS-Clus has the ability to adapt its model to handle concept drift and feature evolution for textual data streams. Experimental results have been performed on textual datasets where efficient and promising results are obtained.

Keywords: Data stream clustering; concept drift; novelty detection; concept evolution; artificial immune system.

1. Introduction

In recent years, the emergence of new technologies and real-time applications has brought an enormous volume of data shared between worldwide users anytime and

*Corresponding author.

This is an Open Access article published by World Scientific Publishing Company. It is distributed under the terms of the Creative Commons Attribution 4.0 (CC-BY) License. Further distribution of this work is permitted, provided the original work is properly cited.

anywhere. The fast and continuous flow of information generated from real-time applications is known as data stream. Data stream can be defined as an ordered sequence of data items flowing continuously at a high-speed rate. Many sources, such as social networks, mobile and Web applications, and telecommunication services, may generate these streams of data. This real-time data is omnipresent in our daily life and may contain a lot of knowledge and hidden information. Nowadays, many companies are exploiting those flows of data to increase their sales by matching their products to the expectations and interests of potential buyers.

Data mining techniques are used to automatically mine and extract interesting and hidden information from the very large generated datasets. Despite the efficiency of data mining techniques in static domains, processing such continuous streams of data entails many additional constraints for querying and extracting knowledge. Even though several works have been proposed to tackle those problems, extracting hidden knowledge from infinite flows of data remains an important and difficult task which requires robust and efficient algorithms. This has given rise to a new domain called data stream mining.

Because of their unbounded size and high-speed rate, mining data streams require online algorithms that analyze data only once. Among the common tasks concerned with the data stream mining field we find classification,¹ association rules,² and clustering.³ Currently, the latter is the most commonly used data mining technique. Most of the recent researches are focusing on clustering data stream since classes of data are unknown, data are evolving rapidly, and new classes may appear instantaneously.

The study of data stream has generated in recent years a lot of activities and today it still presents many challenges to the scientific community.^{4,5} In addition to the clustering problem of continuous flows of data, the evolving nature of data stream is the most important issue to encounter when handling this kind of data. In fact, our dynamic environment is naturally changing over time, and that raises the significant variations in the data distribution processed by real-time applications. This phenomenon is known as concept drift. A typical example of concept drift is a customer's buying preferences, which may change in a specific period of time due to fashion, economy, or other contexts. A clustering algorithm for data streams should then effectively consider those changes and adapt its model as time flies.

Another challenging problem posed by data streams is novelty detection. This was considered as one of the most challenging issues in the field of data stream mining. It has therefore attracted the attention of many researchers in the machine learning community.^{6,7} Novelty detection in data stream mining aims to identify novel or unknown concepts from ever-growing flows of data. For instance, novelty detection in micro-blogging services refers to the emergence of new hot topics and the withdrawal of other topics in a specific period of time.

Moreover, flows of data are produced in different forms, e.g. text, images, and videos. In particular, many unstructured data feeds generated from the World Wide Web are considered as the most commonly shared and produced type of data

between Web users. Text data stream poses additional challenges to data stream processing due to its dynamic evolving nature and thus its unknown feature space. This is becoming the center of interest of many researchers who have focused on tuning and changing data mining techniques to address the problem of feature evolution in a huge volume of text.

This work substantially extends an earlier conference paper by the authors.⁸ In this paper, we aim to design a novel method for data stream clustering in the context of text data domain. For this purpose, we have guided our research by using a bio-inspired algorithm. This choice is argued by the powerful learning mechanism that natural organisms exhibit. These organisms are able to tackle complex problems thanks to their inherent learning capacity. Among bio-inspired algorithms, we have focused on the Artificial Immune System (AIS) meta-heuristic. AIS is an optimization method that takes the benefits and properties of natural immune systems, such as diversity, self-adaptation, and dynamic learning. AIS has powerful adapting capabilities that keep it robust even in changing environments. Since this method owns the property of ease of adaptability to dynamic environments, which is required for data stream learning algorithms, we decided to use it for the data stream clustering problem.

The remainder of this paper is organized as follows. In Sec. 2, we present related works in the data stream mining field. In Sec. 3, we give an overview of the AIS meta-heuristic. In Sec. 4, we expose the proposed algorithm for text data stream clustering. In Sec. 5, we discuss the experimental results. Ultimately, in Sec. 6, we present conclusions and future works.

2. Related Work

With the ever-growing amount of data, traditional data mining techniques are no longer technically feasible. While old techniques are designed for mining static data, algorithms for data stream mining should be both efficient and adaptive to manage a large amount of data with continuous unknown changes in its distribution. A broad spectrum of data stream methods and algorithms is provided in Ref. 9 where data mining techniques are adapted to handle data stream issues. For instance, it is no longer possible to process data by using multiple scans. Data must be analyzed online and one time. To limit the use of memory, data summaries about previously seen data are computed and stored. Therefore, the novel arriving data stream should be manipulated in real-time and then discarded. In addition, data may evolve over time yielding to the phenomena of concept drift.

Dealing with the above-mentioned constraints, several techniques and algorithms are proposed to mine and analyze data streams. As previously stated, one of the widely used techniques in the data stream mining researches is clustering. Clustering is an unsupervised learning task dedicated to group similar objects together and dissimilar objects into different clusters. Comprehensive studies on clustering techniques have been discussed in Ref. 10.

2.1. Data stream clustering

Clustering data stream addresses the problem of how to group similar points periodically with considering the evolving nature of clusters when the distribution of data stream changes.¹² As data stream is rapidly evolving over time, clustering algorithms should be as fast as possible to manage the high speed of arriving objects. In Ref. 11, author discussed the requirements of clustering data streams. Indeed, clustering algorithms should fulfill some characteristics, such as the compactness of representation, the fast incremental processing of new data points, and the fast identification of outliers in noisy data.

In order to extract hidden information from large amounts of continuously generated data, numerous traditional algorithms are extended to cope with data stream issues. References 13 and 14 are based on the K -means algorithm.

In Ref. 13, CluStream is proposed as a clustering framework based on two components: online and offline. The online component periodically stores summaries of clusters. This representation of the temporal extension of the cluster feature vector is known as micro-cluster. Whenever a novel point arrives, a Euclidean distance is computed and the closest micro-cluster will absorb that point. If there is no selected cluster to store the novel data point, a new micro-cluster is then created. The second component, the offline component, provides to the user a quick understanding of clusters. This phase is based on summaries of clusters provided by the online step and some inputs of the user. It provides actually found clusters in different timestamps. CluStream has been tested on real datasets and significant results are obtained in terms of accuracy. However, the concept drift and the novelty detection issues are not treated by this algorithm.

In Ref. 14, StreamKM++ is proposed as a new streaming algorithm based on the K -means++ algorithm.¹⁵ Standard K -means algorithm initializes the initial set of cluster centers randomly whereas K -means++ chooses initial centers that are probably close to the optimal solution. The specific way of choosing these centers is based on their probabilities. As a consequence, K -means++ gives better results than standard K -means. In fact, StreamKM++ is based on two steps: the merging and reducing steps. To reduce the high-dimensional data, a new data structure called Coreset Tree is used to reduce $2m$ objects to m objects. To do so, a Coreset Tree is represented by a number of points. This set of points is divided into homogeneous groups using hierarchical divisive clustering. A number of clusters are then obtained which represent the leaf nodes of the tree. The Coreset construction is mainly based on the K -means++ algorithm. After that, leaf nodes are represented by summary information, such as the number of points and the sum of squared distances. For the merging step, another data structure called bucket set is defined and it consists of L buffers. Each bucket can store a fixed number of m objects. When a new data point arrives, the first buffer will be examined, if it is full all its data will be removed and stored in a secondary bucket. Objects in the second bucket are then merged with the new arrivals. Then a Coreset Tree is constructed to reduce the number of stored

data. After having processed the whole input stream, the K -means++ algorithm is applied to obtain the final clustering solution.

The problem of clustering data stream has been also studied with density-based algorithms since they provide an efficient tool for capturing dynamic changes in data stream and detecting outliers.^{16,17} Due to the fact that it is impossible for clustering algorithms to store all streams of data, the micro-cluster technique is commonly used for effectively representing the most important information of clusters.¹⁸

In Ref. 17, authors proposed a new algorithm, called DenStream, for clustering evolving data streams. In this work, the damped window model is used and a weight is assigned to each data point to evaluate its importance over time. The proposed algorithm has two steps, the online step that maintains micro-clusters and the offline step that generates final clusters. Given a set of points, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm¹⁹ is applied to accomplish the online process. This step performs clustering on the basis of two types of micro-clusters. The first one, core micro-cluster (c-micro-cluster), is dedicated to summarize clusters with arbitrary shape²⁰ by associating the closest points together to form a micro-cluster represented by its radius, its center, Euclidean distance between its points, etc. The density of c-micro-clusters is constrained by a small value of radius so the number of c-micro-clusters will be much larger than the number of natural clusters. The second type, potential micro-cluster (p-micro-cluster), is a c-micro-cluster which consists of grouping similar objects together at a given timestamp, i.e. summarizing potential clusters. When a new point arrives, it will be merged with the nearest p-micro-cluster or c-micro-cluster according to the radius. The weight of p-micro-cluster will decrease gradually if no point is merged in, and thus it will be deleted from the memory to promote the creation of novel p-micro-cluster. Indeed, p-micro-cluster may become c-micro-cluster if its weight is above a certain threshold. If the novel arriving point does not fit to any c-micro-cluster or p-micro-cluster, an outlier micro-cluster (o-micro-cluster) is then created containing outlier instances. However, the weight of o-micro-cluster is periodically checked: if it is above a certain threshold, it will grow and become p-micro-cluster otherwise it is kept in the outlier buffer.

After detecting the density area of data stream, the offline part will be applied. This part is based on a variant of DBSCAN to generate final clusters. These final clusters are formed by searching the density-connected c-micro-clusters which are considered as virtual points for clustering. Nevertheless, DenStream presents some problems since it promotes p-micro-clusters instead of o-micro-clusters. Actually, the new arriving data point is first merged in p-micro-clusters. Otherwise, if it does not fit with any p-micro-clusters, the new point is merged into the closest o-micro-cluster. As a consequence, o-micro-clusters can be easily deleted. This strategy omits to detect and handle a challenging issue in data stream which is concept evolution. In addition to that problem, DenStream requires adjusting many parameters to have good clustering results.

Another density-based algorithm, known as D-stream, is proposed in Ref. 20. The proposed framework allows manipulating the evolving clusters without the need of user's adjusting parameters. Like Ref. 17, D-stream consists of an online component and an offline component. Whenever a new data point arrives, the online component places it into a corresponding density grid and continuously updates the characteristic vector of the density grid. On the other hand, the offline component initially generates new clusters and dynamically adjusts them.

Another work proposed in Ref. 21 deals with outlier detection in data stream mining. Authors proposed a novel algorithm called memory-efficient incremental local outlier detection (MiLOF) that detects local outliers in a system with limited memory. To address this problem, the authors proposed a novel adaptive clustering approach that builds summaries of data points in order to keep all necessary information with limited memory space.

2.2. Concept drift

Among challenging phenomena in data stream learning is concept drift. The latter occurs when new observations appear while old ones become irrelevant. This change happens over time due to the changing characteristics of our dynamic environment. To cope with this issue, applications have to include a change detection strategy in their learning process in order to identify concept drift. Reference 22 presents a description of existing algorithms for the concept drift challenge. Another intriguing problem is how to make difference between noise and change in the data distribution. Indeed, algorithms for change detection should be able to filter out noise as it provides no relevant information.²³

In a nonstationary environment, the concept of data may drift over time, so robust techniques are required to handle this challenging issue. When the distribution of data changes, the classification models should be updated to fit the novel data. For that, three main approaches are proposed in the literature. A first approach is a window-based approach which builds a classification model based on a slice of data instances in a given time frame. The framework proposed in Ref. 24 deals with the sliding window approach to manage the concept drift issue.

Initially, a classification phase is performed to classify arriving documents. Then for the labeling purpose, the most useful instances are selected using an active learning-based approach.²⁵ Relying on those examples, the classifier is updated by the most recent concepts.

The second approach proposed in the literature is ensemble methods. Instead of training a single classifier, many researchers have investigated the technique of combining the predictions of multiple classifiers. Upon receiving a new data point, it will be classified by taking a vote from combined predictions.

The method proposed in Ref. 26 deals with the ensemble classifier technique. It uses the traditional mining classifier (decision tree) which is updated automatically to reflect the most recent concepts in data streams. The first step is to assign a weight

for each training instance based on its posterior probability. Then for each chunk of a new data stream, a decision tree is built. After that, a classification step is performed and a weight is assigned to each decision tree. Consequently, concept drift is managed by replacing the existing model having minimal weight by the newly constructed model. Moreover, to detect novel class, each incoming instance is classified and a class label is associated on the basis of a voting strategy. If the novel data point does not belong to any cluster, it will be stored in a separate dataset for further analysis. The novel class is then detected by examining the cohesion between non-classified instances.

The third type of techniques is the weight-based approach which assigns a distinctive weight for each training instance.²⁷ The assigned weight is generally based on certain factors such as the time for storing the data. So, this weight allows to promote the contributing instances and discarding outdated ones. Thus, its learning process is similar to that of window-based approaches.

In Ref. 28, author proposed a weighted approach to handle concept drift in data streams. The proposed approach assigns a weight for each training example according to its appearance over time. Thus, the time-based forgetting function promotes and makes the last examples more significant for the learning algorithms than the old ones. As time flows, the importance of examples decreases. So, observations with old age are forgotten and replaced by newer ones.

2.3. Novelty detection

In many real-time applications, robust applications are required to detect changes in data. Any fault or failure in some such systems may lead to undesirable results and may cause disasters. It is obviously important to identify unforeseen or abnormal phenomena as soon as they appear to avoid unwanted results. Furthermore, novelty detection constitutes an active research area which has received considerable attention. This challenging topic provides an effective learning paradigm for many emerging applications. In a clustering context, the novelty detection seeks to detect novel classes. A significant amount of work was focused on novelty detection as a challenging task for the data stream mining. Recent comprehensive studies on novelty detection techniques are presented in Refs. 6 and 29.

In Ref. 30, an efficient technique is proposed for automatically detecting novel classes. In this work, data stream is divided into equal-sized chunks. Each novel data is either associated to the closest cluster or declared as an outlier if it falls outside the decision boundary. Then the buffer of outliers is examined to check if there is enough cohesion between outlier instances to form a novel class. The proposed technique follows the ensemble classifier approach to reflect the most recent concepts in the data stream. To do so, a classification model is trained on a novel arriving data chunk and the existing ensemble is updated by this newly constructed model, if necessary.

In the context of social networks streams, Ref. 31 addresses the problem of clustering and event detection in social networks. As a first step, a fixed number of

clusters are defined with their corresponding summaries. New data streams are continuously absorbed by the corresponding clusters, and new events are detected based on the growth rate of clusters.

A recent work⁷ has addressed the problem of multi-class detection. As a first step, the proposed algorithm, called MINAS, builds a decision model from labeled data. Then, for the streaming phase, new instances that fit with existing classes are correctly classified and unknown examples are stored separately for further analysis. Those groups of unidentified examples serve in updating the existing model with novel classes detection.

2.4. Feature evolution

The feature evolution generally occurs in text streams where new features emerge as time flies. In static data mining, during the training phase, a static feature vector is constructed from initially introduced data. Unlike static data, streaming data is evolving over time and a fixed feature vector is almost unavailable.

Feature evolution is identified as a consequence of the three main challenging characteristics of data stream, i.e. the infinite length, the concept drift, and the concept evolution. Indeed, due to the unbounded size of data stream, a dynamic feature vector is required as new features may appear over time. In addition, the concept drift leads to a change in the distribution of data and thus these changing characteristics can be identified by the presence of novel features. This evolution can happen in two ways:

- Adding novel attributes to the underlying data, e.g. new words appear in the case of text data. Then old features may have faded out and replaced by new ones.
- New values appear for existing attributes, this case is particularly designed for numerical data.

The work presented in Ref. 32 treats the main problem of text stream classification, which is the dynamic feature space. At the first stage, a feature selection method is applied to select the N best features. This is done by computing the number of times where a word appears in documents. In the second stage, an incremental learning algorithm is performed for the classification purpose. A fading factor is associated with each word to indicate its importance in the feature vector. As a new document arrives it is checked whether it contains any new words. The fading factor will be either incremented or is initialized to zero if novel words are added to the feature vector.

Considering the case of Twitter micro-blogging services, Ref. 33 proposes a new technique for detecting emerging topics. A vector of terms with their frequencies will be constructed from users' tweets. Each term has a life cycle to assess its importance during a specific time interval. Finally, the set of emerging topics is identified due to a graph containing links between all relevant and emerging terms. The first step aims to extract relevant content. To do that, the authors proposed to associate a weight

for each term in the tweet and thus a Twitter vector is constructed. Then the usage of each keyword is evaluated through a statistical analysis of its life cycle. The temporal information is critical to evaluate the importance of the term (effectiveness contribution of a term).

Identifying emerging terms leads to retrieval of emerging topics, and that by analyzing the semantic relation between these terms. The co-occurrence information is used to extract the semantic relationships between relevant keywords. In Ref. 34, authors proposed to tackle the problem of concept evolution and feature evolution by using the lossless homogenizing technique. This technique uses the union of features from both the initial feature set of historical data and the novel incoming data. Through this technique, the proposed method can effectively preserve and promote the appearance of novel features.

3. The AIS Meta-Heuristic

Swarm intelligence refers to the natural behavior of individual agents in achieving complex activities. This group of agents (bacteria, bees, fish, ants, etc.) cooperates with each other and with their environment in order to find the best solution. This natural interaction leads to no centralized behavior. Many researchers have mimicked the intelligent behavior of those individuals and have created intelligent systems based on what happens in our natural environment. Examples of such systems include Ant Colony Optimization (ACO),³⁵ Particle Swarm Intelligence,³⁶ Artificial Immune System,³⁷ Genetic Algorithms,³⁸ etc.

The Artificial Immune System is among those bio-inspired methodologies that have shown considerable results. It is a novel computational intelligence technique which simulates the immune features of the Biological Immune System.

The Biological Immune System is an elaborated defense mechanism, which has evolved over millions of years. This system is a multi-layered system having evolutionary mechanisms to protect the body against foreign invaders.

The Biological Immune System is composed of a set of cells, organs, and molecules that cooperate together to defend and protect the organism from infection. The immune system consists of lymphocytes which are special types of white blood cells (B-cells) that are responsible for detecting and destroying pathogens. In the presence of a pathogen (or antigen), B-cells secrete antibodies on their surface to recognize and bind for this new invader (antigen). Produced antibodies will undergo cloning and mutation processes. The cloning process consists of producing similar antibodies while mutation consists of changing the antibody characteristics to better fit the introduced antigen. Those steps help to rapidly recognize and identify the foreign invader.

To tackle real-world problems, the Biological Immune System has inspired computer scientists to simulate the concepts of the natural immune system on real applications.³⁹ This has led to the emergence of a new branch of computational intelligence known as the AIS meta-heuristic.³⁸

The AIS is mainly based on three theories. The Immune Network Theory⁴⁰ has been widely used for the data clustering. The negative selection algorithm is initially introduced in Ref. 41 as a learning algorithm to protect computers from malicious intruders, such as viruses. Finally, the clonal selection principle is used to perform

Fig. 1. Clonal selection principle.

machine learning and pattern recognition tasks. These days, the most used model is the clonal selection principle. This model describes the process of simulation and proliferation of B-cells in order to defend a malicious attacker (see Fig. 1).

AIS represents an efficient technique to cope with the evolving nature of data streams since it is inspired by Immune System that treats this kind of properties within foreign substances. Also, the Biological Immune System allows distinguishing between self- and nonself-cells, thereby AIS provides an effective distinction between noisy data and has the ability to detect novel classes. Due to its powerful ability of learning, its decentralized property, and its self-adaptation in managing the dynamic environment, AIS was successfully applied in many clustering problems.^{42,43} AIS is a meta-heuristic that can be efficient for manipulating the data stream problems too.

In Ref. 44, the authors propose a new clustering method (TECNO-STREAMS) based on the Artificial Immune System. This new method addresses the problem of mining evolving user profiles in noisy datasets. Instead of using classical B-cell, Dynamic Weighted B-cells (D-W-B-cells) are introduced. D-W-B-cells serve in determining an influence zone of the training data. Initially, the K -means algorithm is performed to cluster the initially introduced population. Each time a new antigen is presented to a D-W-B-cell the influence zone will decrease with the distance of the antigen. So, the more recent data will have a higher influence zone. Also, a time factor will be associated with each D-W-B-cell to indicate its contribution in a specific time interval. Whenever a new antigen is presented, if it succeeds in activating a set of D-W-B-cells, their ages are refreshed. Then all D-W-B-cells will undergo clone and mutation processes. If the number of newly generated population exceeds the size of the original population, the most contributed D-W-B-cells are kept and others are removed. The set of D-W-B-cells is scanned periodically by performing the K -means algorithm to track the dynamics of evolving clusters.

4. The Proposed Method

As discussed in Sec. 2, data stream poses many challenges, such as infinite length, concept drift, and feature and concept evolution. So far, most of the researches have focused on the first two challenges and have ignored the concept and feature evolution issues. Mining data stream requires one-pass algorithms, and thus traditional algorithms are not feasible to perform that task. In addition, algorithms should be able to detect any change in the streaming data. In the context of text stream mining, additional difficulties are encountered since text manipulation requires additional memory space and requires robust algorithms to manage its rapidly changing characteristics.

In this paper, we propose an algorithm for text stream clustering. In order to deal with data stream issues, we propose to integrate Artificial Immune System meta-heuristic. To handle concept drift issue, we propose to use a forgotten mechanism

that permits to keep newer concepts and forget the concepts with long period of inactivity.

Due to the powerful adaptability of AIS, we have proposed the AIS-Clus algorithm to handle challenging issues in textual data streams on the basis of AIS concepts.

4.1. Problem definition

A data stream is an infinite sequence of elements. It can be represented as $\{X_1, \dots, X_{\text{now}}\}$ where X_i is the i th instance presented in a specific time. Each data point X_i is an n -dimensional attribute vector represented as $\{A_{1i}, A_{2i}, \dots, A_{ni}\}$.

Initially all data points share the same feature vector. In the streaming scenario, novel features can appear and others can disappear, thereby each novel instance will have a specific feature vector. The new incoming data is divided into equal-sized chunks where each batch contains a fixed number N of instances.

In this work, the concepts of Artificial Immune System are used as an efficient mechanism for the self-adaptation of existing clusters with the novel introduced data. In AIS, B-cells are responsible for the production of antibodies to defend new invaders. Considering this biological observation, we define valid clusters as B-cells. As data stream is characterized by its flow of continuously generated data, this is similar to the biological concept in Immune System where new invaders enter in to our body. Thereby, new incoming data is considered as antigens which are foreign to existing B-cells (clusters).

Affinity between an antigen and a cluster (collection of similar points) represents the cohesion between those two substances. In the proposed framework, we define the affinity measure as a scoring function based on the entropy variation and a similarity metric.

The process of the proposed framework is described as follows.

For each incoming data X_{new} , a scoring function is calculated in order to examine the affinity values to each cluster. Once determining the closest clusters, the Artificial Immune System is involved to determine which cluster will absorb this novel invader. For this purpose, we apply both the clonal selection principle and the negative selection mechanism. The first model, the clonal selection, consists in the production of antibodies to best match the new invader. Those antibodies are created from existing B-cells (clusters) to undergo a mutation process. This model seeks to handle concept drift and feature evolution challenges. The negative selection mechanism enables detection of noisy data. In fact, instances declared as outliers may belong to novel classes. Thereby, the negative selection process is an efficient approach for detecting novel classes.

The proposed framework consists in clustering streaming data and considering the four main issues previously defined:

- (1) The infinite length is managed by dividing novel arrived data into equal-sized chunks; every chunk is processed online and one time. For each chunk, we keep

only valid instances belonging to existing clusters or forming novel concept. Noisy data will be eliminated from the memory since it does not provide any useful information.

- (2) To deal with the concept drift issue, a life cycle is associated to each term to indicate its contribution over time. By this fact, some terms will lose their importance as time flows so they will be eliminated and replaced by new appearing features. Because of that, clusters which are inactive for a long period of time are removed and replaced by novel appearing classes. Hence, we keep the most recent concepts in the learning process.
- (3) Concept evolution issue is managed by associating higher fitness value to novel appearing features. For each chunk of data, we apply a density-based clustering algorithm in order to detect novel classes.
- (4) Finally, the feature evolution challenge is handled by considering the feature space dynamics. Whenever a new data instance arrives, the set of its novel features is included in the original feature space. In order to ensure greater importance of more recent concepts, a survival factor is assigned to each term. This strategy aims to manage the high-dimensional size of feature vector by eliminating useless features and promoting novel ones.

Most existing data stream classification techniques consider only the detection of one novel class at a given time. Our framework, on the other hand, is able to detect multiple classes simultaneously.

4.2. AIS-Clus: The proposed method

The proposed framework aims to cluster text data stream using the artificial immune network meta-heuristic.

4.2.1. Text preprocessing

Text is characterized by its high-dimensional nature and sparsity of data. It represents unstructured format, such as emails, articles, and social networks. Text mining aims to find useful information from this unstructured format. A preprocessing step is crucial to reduce the huge size of textual data. The first step in text preprocessing is the filtering process. This preprocessing step leads to more compact and structured data. Accordingly, in our case, this step is required for an effective text mining technique.

Then, in order to improve the representation of documents, all punctuation marks should be removed, as well as tabs and other nontext characters, and replaced by single white space. Stop words removal technique consists in filtering out common terms of natural language. This step alleviates the task of preprocessing text data. The set of remaining words is called the dictionary of a document collection. In addition to stop words removal, frequently occurring words may represent little information content and should be removed from the dictionary. In addition, words like URLs and reply and repost syntax (@username) are removed.

In the proposed framework, the English language is adopted. Using LangDetect⁴⁵ module, non-English language words are filtered out and considered as nonrelevant. In order to reduce the number of words in the dictionary, a selection strategy is required to select the most informative words. In the proposed framework, we define a survival factor for each term to keep the most important words.

Each document is represented as a vector in m -dimensional space. More formally, a document D is represented as $\{X_1, X_2, \dots, X_m\}$ where X_i represents the i th feature. In our proposed framework, each feature X_i is defined as a tuple (Word, SURV, Wight).

Each tuple component is defined as follows:

- A “Word” is a string.
- A survival factor “SURV” $\in [0, \infty]$ calculates the word period of life.
- A weight “Weight” is a normalized value that indicates the importance of that term in the document. The weight value is calculated as follows:

$$\text{Weight}(X_i) = \frac{\text{FREQ}(X_i)}{m}, \quad (1)$$

where FREQ represents the number of times the word appears in the document and m is the number of words in D .

For clustering text data, we consider a set of text documents. In our framework, data are represented by a set of instances.

Each instance is defined as a tuple (ID, $\text{Vec}_{\text{feature}}$, $\text{Vec}_{\text{outlier}}$). Each component is defined as follows:

- ID which is a unique number to identify the instance.
- $\text{Vec}_{\text{feature}}$ is a vector of features which are present in that instance.
- $\text{Vec}_{\text{outlier}}$ is a vector of features which do not belong to the $\text{Vec}_{\text{feature}}$ vector.

The proposed algorithm, AIS-Clus, is a two-step method. The first step scans the static data to build an initial model. The second step performs the streaming scenario to cluster continuous flow of data and detect novel classes.

4.2.2. Initial clustering

Data stream is characterized by the unknown number of clusters. Clusters are dynamic with time, some of them will disappear, others will grow, and novel clusters appear. Another gap is that clusters are density-shaped. Our algorithm consists in creating clusters from text streams. First, the input data will be scanned and a clustering algorithm will be performed to construct homogenous groups of data. In this work, we adopt a density-based algorithm. The initial clustering is performed on the basis of the DBSCAN¹⁹ algorithm. DBSCAN is essentially based on two parameters: the neighborhoods parameter that defines a region of a given radius (eps) and the number of objects (MinPts) it contains.

Density-based algorithms seem efficient in the context of text stream clustering since text data contains plenty of noise because of its huge number of features. Likewise, clustering such data stream is characterized by its arbitrary-shaped clusters, and thus density-based algorithms are efficient to handle this issue by detecting arbitrary-shaped clusters. Data stream evolves over time, novel concepts may appear, and others disappear, so there is no assumption about the number of clusters.

In the context of text mining, the cosine similarity is the most commonly used for high-dimensional spaces. As we aim to cluster textual data, we depend on the cosine metric to calculate how much two messages are closer from each other.

Given two documents D_1 and D_2 having, respectively, the two vectors of weights $W_1 = \{W_{11}, \dots, W_{1n}\}$ and $W_2 = \{W_{21}, \dots, W_{2n}\}$ where W_{in} denotes the weight for a feature n in a document i . The similarity between D_1 and D_2 is calculated as follows:

$$\text{Cosine}(D_1, D_2) = \frac{\sum_{i=1}^n W_{i1} \times W_{i2}}{\sqrt{\sum_{i=1}^n W_{i1}^2 \times \sum_{i=1}^n W_{i2}^2}}. \quad (2)$$

After performing the clustering of available historical data and obtaining homogeneous clusters, summary information is associated to each cluster to provide a global description about it.

4.2.3. Storing the summary cluster information

The summarization technique plays an important role in the data stream mining and it is widely used to reduce the volume of manipulated information and the memory space. The summary statistics provide a global and sufficient description of a given cluster. So, a cluster “Clus” is represented as $\text{Clus} = (\text{Clus}_{\text{feature}}, \text{Clus}_{\text{outlier}}, \text{Centroid}, \text{Radius})$.

Definition 1. $\text{Clus}_{\text{feature}}$ is a vector of features which are words occurring in points of the cluster.

Definition 2. $\text{Clus}_{\text{outlier}}$ is a vector of features which do not belong to $\text{Clus}_{\text{feature}}$.

These summaries are used in order to calculate scores for novel arrived instances in the test data. The cluster having the strong cohesion with the novel data will absorb it. Else, it is declared as outlier and will be used for further analysis.

4.2.4. Clustering streaming data

Data stream clustering is defined as the clustering of data that flows continuously. Clustering such data requires online and one-pass algorithms. Our proposed method divides streaming data into equal-sized chunks. Each chunk is processed online and one time.

The following steps describe our main strategy for clustering continuously arriving data.

(1) Scoring function

In our work, we aim to minimize a scoring function to evaluate the performance of a given solution. This function is used to decide whether a novel instance is an outlier or belongs to one of the existing clusters.

Whenever a novel instance arrives, a score metric (defined as affinity function) will be calculated to quantify the compactness of this new point with each existing cluster.

To calculate the affinity between a point P and a cluster Clus_i the following formula is used:

$$\text{Affinity}(P, \text{Clus}_i) = \alpha \times e^{-\gamma \times \text{Cosine}(P, \text{Clus}_i)} + \beta \times V, \quad (3)$$

where α and β are two weighting factors. In order to make a consideration to the similarity information, we use the exponential distribution. As we aim to obtain a small value of $\text{Affinity}(P, \text{Clus}_i)$, the exponential function $e^{-\gamma \times \text{Cosine}(P, \text{Clus}_i)}$ is low when the $\text{Cosine}(P, \text{Clus}_i)$ is high and *vice versa*.

V is an entropy-based formula calculated as follows:

$$V = \text{Entropy}(\text{Clus}_{\text{after}}) - \text{Entropy}(\text{Clus}_{\text{before}}), \quad (4)$$

where $\text{Clus}_{\text{after}}$ is the cluster after adding the new point P and $\text{Clus}_{\text{before}}$ is the cluster before it changes. If $V > 0$ the added point disturbs the cluster characteristics otherwise the novel added point reduces the cluster entropy and then it fits with that cluster.

Given a random variable X , the entropy value $\text{Entropy}(X)$ measures the uncertainty of this variable. In our framework, we use the entropy measure in order to evaluate the orderliness of a given cluster before and after putting a novel point in it.

Suppose the vector $\text{Clus}_{\text{feature}} = (X_1, X_2, \dots, X_n)$ where $X_j, 1 \leq j \leq n$, is a feature word.

The information entropy $\text{Entropy}(\text{Clus}_i)$ of Clus_i is calculated as

$$\text{Entropy}(\text{Clus}_i) = \text{Entropy}(\text{Clus}_{\text{feature}}) = \sum_{j=1}^n H(X_j), \quad (5)$$

where the entropy $H(X_j)$ is defined as

$$H(X_j) = - \sum_{i=1}^d p_i \log_2(p_i), \quad (6)$$

where p_i defines the probability of an event $X_j = v$ where v belongs to the possible values of X_j . Whenever a foreign point enters into a cluster where most of its data are similar, it will disturb its homogeneity leading to an increase of the entropy value.

As we adopt features of AIS, the scoring formula represents the affinity value between B-cells and foreign invaders. Affinity measure evaluates the degree of interaction of antibody-antigen.

After calculating the affinity value with all clusters (B-cells), they will undergo a selection step to be ready for cloning and mutation.

(2) Cloning process

The process of cloning consists in producing similar copies from the original clusters. Those copies will be used during the mutation process. The number of clones to make from each cluster depends on the affinity value between P and Clus_i :

$$\text{NumClones}(\text{Clus}_i, P) = \frac{(\text{NumComWords}(\text{Clus}_i, P) + \rho)}{2}. \quad (7)$$

As mentioned in formula (7), the minimum number of clones is one as the weighting factor ρ is set to 2. This can be explained by the fact that we aim to explore all the space of existing clusters. The $\text{NumComWords}(\text{Clus}_i, P)$ refers to the number of joints features between novel point P and the cluster Clus_i .

(3) Mutation process

In genetics, mutation consists in changing characteristics of different organisms. This operation allows clones to diversify their repertory and to evolve by changing their structure. In our algorithm, the mutation process aims to make existing clusters more similar to novel incoming data.

The clonal selection principle produces cloned cells ready to undergo a mutation process. In our proposed approach, the mutation process consists in adding a number of features to the $\text{Clus}_{\text{feature}}$ vector. The mutation rate is the number of words to add to a given cluster. It is defined as

$$\text{WordsToAdd}(\text{Clus}_i, P) = \eta * \text{NumComWords}(\text{Clus}_i, P). \quad (8)$$

New added features are selected from $\text{Clus}_{\text{outlier}}$ vector according to their weights. To promote best features to be chosen, the roulette-wheel selection procedure is applied.

After performing the mutation process, clones evolve and change their structure to best fit introduced antigens. Closest clones to the antigen are selected to replace their parents. Clones which are far from antigen are removed.

(4) Merging B-cells

Due to their cooperative behavior, natural organisms attempt to find optimal solutions and exhibit a powerful learning process. In natural immune system, B-cells cooperate together in order to defend the new invader. As we aim to find a global optimal solution, we simulate this cooperative behavior by calculating an affinity measure between B-cells to merge similar ones. For this purpose, we calculate the summary of each cluster. Then the affinity between the centroids of two B-cells is calculated based on the Cosine function:

$$\text{Merging}(\text{Clus}_i, \text{Clus}_j) = \text{Cosine}(\text{Centroid}_{\text{Clus}_i}, \text{Centroid}_{\text{Clus}_j}), \quad (9)$$

where $\text{Centroid}_{\text{Clus}_i}$ and $\text{Centroid}_{\text{Clus}_j}$ are, respectively, the two centroids of clusters Clus_i and Clus_j .

4.3. Handling data stream challenges using AIS-Clus

This Subsection describes how our proposed algorithm AIS-Clus can beneficially handle the challenges of data stream.

4.3.1. Concept drift detection in text streams

Concept drift is a challenging issue in data stream mining. In text stream mining, a type of concept drift that may appear concerns the changing of the importance of features. In a given period of time, some features become hot (higher importance) whereas other features fade out (lower importance) and disappear. To handle concept drift in text data stream, we propose a survival factor that evaluates the importance of terms in each cluster. For example, clusters can share the same terms but each one has a different survival factor value. This allows evaluating the contribution of terms in different clusters at each period of time. This strategy simulates AIS by considering each term as a surviving organism.

At the beginning, all clusters will have a number of features. The survival factor SURV is set to one for the words of the cluster (e.g. features of points belonging to that cluster), otherwise, this factor is set to zero if the word does not belong to the cluster. So, whenever a new instance arrives, the life cycle of terms in the cluster is prolonged if they are present in the novel instance and SURV will be incremented by one. However, a feature dies when there is no nourishment for a long period of time. Indeed, whenever the vector $Clus_{feature}$ of a given cluster exceeds a predefined threshold, SURV of all features is decremented by one. Consequently, strong features are kept and weak ones are removed.

Due to the powerful adaptability of AIS, our approach handles the concept drift problem on the basis of AIS concepts. Based on the mutation process, characteristics of B-cells (clusters) change over time by newly introduced features. Thereby, features of B-cells which are inactive for a long period of time will disappear. Figure 2 illustrates the life cycle of a given term.

Fig. 2. Life cycle of a term.

As mentioned in Fig. 2, the first step is the appearance of the term with a survival factor equal to zero. The nourishment step consists of the activation of the term. In fact, when the term encounters similar other terms (e.g. when introducing an antigen), its survival factor SURV will be incremented. At a given period of time, the term becomes mature. Then as time flows, it loses its importance and becomes inactive.

Finally, the term will be removed and replaced by other more contributed terms. So, the survival factor SURV is used to slowly decrease a term's contribution.

4.3.2. Novelty detection in text streams

Novelty detection can be seen as the task of differentiating between existing data and novel incoming data. Artificial Immune System can prove significant results since it has powerful mechanisms to detect nonself-organisms.

As novelty detection problem in textual data clustering attempts to find novel clusters of data, our proposed algorithm is based on the clonal selection principle to recognize, to identify, and to learn about foreign invaders.

For each incoming instance we calculate its similarity with existing classes through the scoring formula defined in Eq. (3). The existing clusters will proliferate and mutate to best fit the novel data. The closest cluster will absorb that data. But, if the score is above a pre-defined threshold, that data point is declared as outlier and will be stored in a buffer. The outlier buffer is scanned periodically to detect if novel classes appear. This is done by applying the DBSCAN algorithm. Since a density-based algorithm does not require a predefined number of clusters, it presents a powerful mechanism to detect multi-novel classes. Figure 3(a) describes dataset without novel classes whereas Fig. 3(b) describes dataset with novel detected classes.

Fig. 3. Novel class detection.

Fig. 4. Feature evolution process.

4.3.3. Feature evolution in text streams

In our framework, textual datasets are used in order to perform the experiment of AIS-Clus for text stream clustering. Consequently, a module for feature evolution is critical in our work. Text data is characterized by its huge feature vector. In particular, text data stream represents additional issues since its feature vector is dynamic over time.

In this work, we adopt the lossless conversion.⁴⁶ As a first step, our initial feature vector is extracted from historical data using the preprocessing step. For each novel instance, the novel presented features will be added to the original feature vector. The feature evolution process used in our method is illustrated in Fig. 4.

As mentioned in Fig. 4, the first step tests if the incoming data (antigen) fits a given cluster (B-cell). Then the closest cluster absorbs the new data. The last step is the integration of the antigen features into the B-cell. By this fact, novel features will be introduced with historical data leading to the feature evolution process.

5. Experimental Results

In the previous section, we provided a theoretical description of the proposed method. In this section, we introduce the experiments and the corresponding results to prove the efficiency of AIS-Clus algorithm.

5.1. Datasets

In order to test the efficiency of our algorithm, we adopt Twitter datasets as they are best modeled as data stream. The first data is the Sanders corpus^a consisting of 5,513 hand-classified tweets with four topics (Apple, Google, Microsoft, and Twitter). After performing a preprocessing step, 3,452 tweets are obtained for our clustering purpose. In order to consider the streaming process, we divide the data as Shows in Table 1.

^a<http://www.sananalytics.com/lab>.

Table 1. Sanders dataset.

	No. of tweets	Topic	No. of tweets per topic
Initial data	1,200	Apple	600
		Google	600
Streaming data	1,765	Apple	243
		Google	139
		Microsoft	791
		Twitter	592

Initially, we have processed 1,200 instances belonging to “Apple” and “Google” classes. For the streaming scenario, we consider some instances belonging to existing classes and the other ones are considered belonging to novel classes which are “Microsoft” and “Twitter”.

The second dataset is the Health Care Reform (HCR)^b containing 2,516 tweets with eight topics (Health Care Reform, Obama, Democrats, Republicans, Tea Party, Conservatives, Liberals, and Stupak; see Table 2).

For the HCR dataset in the static phase, we have processed 309 instances belonging to “Republicans” and “Stupak” classes. Then, for the streaming scenario, 482 instances are processed to prove the efficiency of our algorithm in detecting the six novel classes.

Generally, Twitter datasets include many types of noise and different types of languages. This led us to a preprocessing step to remove all non-English tweets. Then, we used a stop words removal technique to remove words such as “and”, “if”, “he”, etc. Likewise, we filtered high-frequency and low-frequency words.

To test the scalability of our algorithm on different languages other than English, we have selected as dataset an Arabic corpus tackling the problem of event detection. The dataset EveTAR^c (Table 3) is used to describe different terrorist events from flows of data generated on Twitter.

Table 2. Health Care Reform dataset.

	No. of tweets	Topic	No. of tweets per topic
Initial data	309	Republicans	263
		Stupak	46
Streaming data	482	Democrats	100
		Conservatives	75
		Obama	83
		Tea Party	34
		HCR	151
		Liberals	38

^b<https://bitbucket.org/speriosu/updown/downloads>.

^c<http://qufaculty.qu.edu.qa/telsayed/evetar>.

Table 3. EveTAR dataset.

	No. of tweets	Event	No. of tweets per event
Initial data	1,362	Event 1	590
		Event 2	386
Streaming data	416	Event 3	416

This dataset contains more than the described events in Table 3. Due to time limitation and to be able to compute both Recall and Precision of event detection, we only consider those three events. So, for our experiments, we choose two events: Event 1 describing the suicide bombing in the city of Ab in Yemen, and Event 2 concerns another attack of air strikes in Pakistan. Event 3, which is considered as newly appearing event, concerns the Charlie Hebdo Attack.

5.2. Evaluation

In order to evaluate the quality of a learning system, the following evaluation metrics are typically used: *F*-measure (harmonic mean of Precision and Recall), Purity (rate of correctly classified instances in each cluster), and Accuracy (rate of overall correctly classified instances). Those measures can be used for the static and the stream tests. Therefore, we evaluate the clustering quality by considering the average quality of found clusters in each chunk. In addition, we test the overall misclassification error ERR and the percentage of novel class instances misclassified as existing classes M_{new} .

Given a set of training data instances, the first step is to group these items in to different homogenous groups. For this purpose, we apply the DBSCAN algorithm. However, this density-based algorithm requires two parameters: the eps and MinPts.

The eps parameter defines the minimum number of points required to form a dense region. The eps plays a significant role to control the number of formed clusters. By varying the eps value we obtain results shown in Fig. 5 according to four cases (MinPts = 5, 10, 15, and 20).

Fig. 5. Effect of the variation of the parameter eps.

We can notice that better results are obtained when ϵ is in the range of $[0.5, 0.7]$. On the other hand, the F -measure decreases when the values of ϵ are greater than 0.7. Additionally, when the minimum number of points, MinPts, required for forming a dense region is too small, then increasing the value of ϵ separates clusters and thus the number of clusters increases. The increasing number of clusters lead value to a decrease in the F -measure value at $\epsilon = 0.6$ because the number of unclassified instances increases and the Recall value decreases accordingly.

Concerning the number of clusters, when ϵ is too small, a large part of the data will be in the same cluster and a small number of clusters is obtained. This is because we use the cosine as a similarity measure. On the contrary, higher ϵ values lead to higher number of clusters. After a fine-tuning of this parameter, ϵ , we find that the best value is 0.58 which gives two classes with higher F -measure value equal to 0.91.

In order to fix the appropriate parameters to run AIS-Clus, we vary different parameters to test the sensitivity of AIS-Clus towards them. For the static phase, we have fixed our parameters as follows: $\epsilon = 0.58$ and MinPts = 16. For the stream processing step, using the Artificial Immune System meta-heuristic requires fixing additional parameters which in turn affect the quality of the final results. A number of parameters need to be adjusted to get the expected clustering result. So, after performing a number of experiments, we have fixed the cloning and the mutation rates, respectively, to $\rho = 2$ and $\eta = 2$, and for outlier detection $\delta = 0.12$.

To test the performance of our algorithm, we vary the batches with step sizes of 50, 100, 200, 400, 600, and 800 (Table 4).

As mentioned in Table 4, the best results are obtained when the size of a chunk is equal to 200. Indeed, for each chunk we apply a DBSCAN algorithm to detect novel classes. The novel arriving instances were correctly classified in their novel detected class. However, when the size of chunk is too large, the rate of misclassified novel instances increases since the novel class is not yet detected. So, higher size of chunks delays the detection of novel classes, and decreases the F -measure and the Accuracy values. Thus, novel arriving instances which belong to novel class are either classified in existing classes or considered as outliers for further analysis.

A reasonable size of chunk is required to obtain good clustering results. Based on the results obtained in Table 4, we have fixed the size of chunk to 200 as better results are obtained using that value.

Table 4. Obtained results with different chunk sizes.

Chunk	F -measure	Accuracy	#Class	Time/chunk (s)
50	0.82	0.57	3	31
100	0.81	0.63	4	86
200	0.86	0.78	4	304
400	0.76	0.67	5	1,011
600	0.62	0.62	3	1,534
800	0.67	0.67	6	5,855

Fig. 6. Sanders dataset: Evolution of F -measure, Accuracy, Recall, and Precision values according to the number of instances.

We carried a number of experiments in order to fix the appropriate parameters of AIS-Clus. In this subsection, we carry out some experiments to demonstrate the performance of the presented method for text stream clustering. Figure 6 illustrates the F -measure, Accuracy, Recall, and Precision values obtained with AIS-Clus for Sanders dataset.

Our algorithm detects the novel arrived class “Microsoft” when 400 instances are processed. Likewise for “Twitter” it is detected at chunk 1,400 (Fig. 6) and then novel class instances belonging to “Twitter” are added to it. However, there is a decrease of the Recall curve at instance 1,400. This is because our system delays a bit in detecting the novel class. So, when it appears, its Recall value is relatively low. Then, as time flows, its Recall value will increase.

In order to evaluate the robustness of AIS-Clus in identifying relevant instances, we use the Precision measure. In fact, there is some decline in the Precision curve. This is due to false classifications where novel class instances are classified as existing ones.

For the F -measure, it is strongly related to the Recall and Precision, so we can ensure that F -measure gets high value if Precision and recall are high too. In Fig. 6 there are two minimal values which refer, respectively, to (chunk = 400 and chunk = 1,400) coinciding with the appearance of novel classes. We can also notice that there is a little decline in the F -measure curve justified by the Precision decline in the same period.

In the presence of true labels, we can test the behavior of AIS-Clus in identifying the correct results as the number of streaming instances evolves. As mentioned in Fig. 6 the number of data instances increases while the accuracy of our algorithm slowly decreases. This can be explained by the presence of some novel instances which are classified as existing ones. For example, when novel instances arrive (chunk = 400 and chunk = 1,400) the accuracy slowly decreases. We see at the end

of the learning curve in Fig. 6 that Accuracy apparently presents good results. As a conclusion, we can say that the Accuracy results for AIS-Clus are not very sensitive as the number of instances increases, even when the data distribution changes.

To apply AIS-Clus on the HCR dataset, we perform an adjusting step for specific parameters. The size of chunk will be set to 50 since the dataset contains a relatively small number of instances. In addition, MinPts is reduced to 12 instead of 16. This is because topics of the HCR dataset contain smaller number of instances than Sanders dataset.

We test the clustering results using AIS-Clus on the HCR dataset.

Figure 7 illustrates the variations of F -measure, Accuracy, Recall, and Precision on the HCR dataset. We can notice that F -measure, as well as Recall curve, fluctuates as time flows. However, the five following instances (50, 150, 200, 300, and 350) indicate the moments of detection of novel classes. In fact, we are witnessing a decrease in the F -measure and Recall values at these moments. This is because at the beginning AIS-Clus detects a novel class with a low Recall value (for example when 50 data instances proceed, the F -measure decreases from 0.83 to 0.80), then this value is improved as novel instances are introduced in their appropriate classes. On the contrary, Precision is significantly higher for HCR dataset until the instance 350 where some novel classes are misclassified as existing ones.

For the Accuracy, Fig. 7 shows that the values are stable until reaching the instance 450 when they start to decrease. In fact, when we use a chunk size equal to 50 the last novel appearing class “Liberals” which has 38 instances is not detected yet by AIS-Clus.

5.3. Novel detected classes

In order to examine the performance of our algorithm in detecting novel classes, we compare the times where the novel class starts to emerge and when our system really detects this novel concept.

Fig. 7. HCR dataset: Evolution of F -measure, Accuracy, Recall, and Precision values according to the number of instances.

Fig. 8. Novel class detection results on Sanders.

As mentioned in Fig. 8, initially AIS-Clus has two classes from the static phase. When 382 novel instances have been processed, novel topic “Microsoft” starts to spread its concept. After a while (chunk 400), AIS-Clus detects that class and then novel class instances will be classified in that novel class. Likewise, the topic “Twitter” appears when 1,173 stream instances have been processed. In chunk 1,400, our algorithm detects this novel class and forms a novel cluster.

5.4. Handling concept drift and feature evolution

To cope with concept drift in text data streams, we focus on a weighting term strategy. By simulating the biological metaphor of AIS, a term can be considered as a living organism having a life cycle where terms are born, grow up, and die. Thus, the contribution of terms changes over time depending on their nourishment. This means that their frequencies increase whenever novel similar terms appear. Accordingly, by tracking the occurring changes in the life cycle of terms, we can easily detect the concept drift. Besides, to track the evolving nature in text data (feature evolution), we consider the $\text{Clus}_{\text{feature}}$ feature vector which defines hot terms inside the corresponding cluster. In fact, to cope with the high volume of text data, this feature vector is updated whenever its size exceeds a certain threshold. Consequently, this method keeps the newer features; limits the memory and the runtime of our algorithm.

The following experiment concerns the concept drift and feature evolution issues on the cluster “Apple” which is detected in the initial phase. In order to examine the performance of our method in detecting concept drift, we perform a set of experiments and we give the obtained results in Fig. 9.

Figures 9(a) and 9(b) give the hot terms after updating the vector $\text{Vec}_{\text{feature}}$ of the cluster “Apple”. As shown in these figures, the importance of terms is evaluated over time by considering cases where an update of the feature vector occurs [Fig. 9(a) refers to the first update and Fig. 9(b) refers to the second update]. For example,

Fig. 9. Tracking concept drift and feature evolution in the Sanders dataset.

the contribution of term “Apple” varies from ($SURV = 55$) in Fig. 9(a) to ($SURV = 225$) in Fig. 9(b). Likewise, some terms become more contributing such as “iPhone” while other terms like “Twitter” fade out and will dispartate as time flows. Also, in Fig. 9(a), last words such as “ShopColumbus” and “iPhoneS4” are considered as “drop terms” since they have negative survival factor ($SURV < 225$). This negative value can be explained by the fact that these words are inactive for a long period. Thus, they should be removed to promote novel appearing words. In addition, we can notice that novel features appear, such as “ios5”. This proves that our approach can effectively detect evolving features in text data streams.

Another issue that appears in data stream applications is the presence of noise. In order to test the robustness of the proposed model, we vary the noise rate and we observe the behavior of our algorithm. Figure 10 shows the results of our experiments with different noise percentages (0% noise, 20% noise, and 30% noise).

As mentioned in Fig. 10, there is a meaningless decrease in the F -measure value. Also, the presence of noise causes a meaningless increase in the ERR and M_{new} values. The number of detected classes remains the same for all levels of noise. We can deduct that even when the noise level increases, our model can achieve very high clustering quality.

Fig. 10. Noise variations.

To summarize, Table 5 shows the obtained results in the overall classification, where we give the F -measure evaluation, M_{new} , the novel detected classes, and the missed classes on Sanders, and HCR EveTAR datasets.

In Table 5, we can observe that our algorithm can effectively detect novel classes with good efficiency (F -measure = 0.86 for Sanders, F -measure = 0.79 for HCR, and F -measure = 0.89 for EveTAR). Concerning the number of novel detected classes, AIS-Clus can exactly detect novel classes for Sanders and EveTAR whereas for HCR there is one missed novel class. For the novel class instances classified as existing ones, we can notice from the M_{new} values that our algorithm can effectively detect and classify novel class instances in their corresponding classes with low error rates. However, for EveTAR, M_{new} is relatively high with regard to the other datasets. This can be explained by the reason that our algorithm misclassified novel instances in already existing classes. This misclassification influences the Precisions of Event 1 and Event 2 highlighted in Fig. 11.

5.5. Event detection

To cope with event detection in social networks, we experimented our algorithm with EveTAR to show its robustness in detecting new events. Results are summarized in Fig. 11.

As described in Fig. 11, we can notice that AIS-Clus succeeds in detecting the new class as it appears. At the first stage, for the static phase, our method perfectly

Table 5. Summary of the results.

Dataset	F -measure	M_{new}	Novel classes	Missed classes
Sanders	0.86	0.06	2	0
HCR	0.79	0.05	5	1
EveTAR	0.89	0.10	1	0

Fig. 11. Precision and Recall on EveTAR.

distinguishes between the two classes with high Recall and Precision values. Then, the chunk 200 indicates the moment of the scan of the buffer. This scan leads to the detection of the novel introduced class with also a high Precision value (0.87). The Recall value is moderately good (0.63). However, as time flies, the Recall of the new event is improved to become 0.72. Also, for the Precision it maintains higher values even with the introduction of new tweets. However, as explained in Table 5, the Precisions of Event 1 and Event 2 decrease slightly in the Chunk 200. Then, at Chunk 400 they preserve the same Precision values. These results demonstrate that AIS-Clus can be used to detect events in social text streams even for a different language.

5.6. Comparison of results

In order to compare the performance of our algorithm with the existing state-of-the-art methods, we used MOA.⁴⁸ MOA is an open-source framework for data stream mining. We used CluStream and DenStream algorithms implemented in MOA for our comparison.

The settings for the algorithm parameters were almost default. Table 6 summarizes the clustering quality of each algorithm on Sanders and HCR datasets in comparison with AIS-Clus.

Table 6. Performance comparison of AIS-Clus with CluStream and DenStream.

Dataset	<i>F</i> -measure	CluStream	DenStream	AIS-Clus
Sanders	Recall	0.55	0.12	0.79
	Precision	1	1	0.97
	No. of clusters	100	32	4
HCR	Recall	0.71	0.2	0.67
	Precision	1	1	0.92
	No. of clusters	30	6	7

In our comparative evaluation, we have used the GPrecision and GRecall measures implemented in MOA. The GPrecision measure is the proportion of nonnoise data points returned by the clustering divided by the total number of objects covered by the clustering.

GRecall divides the proportion of nonnoise data points by the number of all nonnoise data points returned by the clustering. So, those measures consider all classes as a “General” class. Using those measures can explicate clearly the obtained results. However, it is not reasonable to obtain high number of clusters with high Recall and Precision measures whereas in our real dataset only a much smaller number of clusters are present. For example, CluStream detects 100 clusters in Sanders dataset with high Recall and Precision values whereas only four clusters are present in Sanders. According to the obtained results in Table 6, the AIS-Clus algorithm often outperformed the DenStream algorithm on Sanders and HCR datasets. However in some cases, the Precision of other algorithms is higher than AIS-Clus. This is because CluStream and DenStream produce more classes than our algorithm. So their Precision is significantly higher. Also, we observe in the evaluation that AIS-Clus also outperforms CluStream and DenStream in detecting novel classes with relatively high Recall and Precision values. Lower Recall values for DenStream algorithm can be explained by the fact that it does not classify a large number of instances and considers them as outliers. Also, the main reason behind the poor performance of CluStream is that the number of clusters is initially predefined by the user. So, CluStream is not adequate for novelty detection problem. We can mention that although stream clustering is an active field of research, no stream clustering evaluation tool exists that offers a suite of implemented stream clustering algorithms and evaluation measures.

As a conclusion, we can notice that the performed experimental evaluation of AIS-Clus on textual datasets demonstrates the effectiveness and efficiency of our algorithm. In addition, our algorithm has demonstrated its robustness in detecting novel appearing classes.

5.7. Discussion

We have presented different experiments performed on the AIS-Clus algorithm. Since AIS-Clus is based on the DBSCAN algorithm, it requires adjusting the distance around an object (eps) and the minimum number of neighbors to include (MinPts). In order to avoid the poor performance caused by improper seed initialization, we have varied the eps parameter to find its best value. Indeed, a small value of eps is chosen because AIS-Clus uses the cosine measure to determine the similarity between different points. Our algorithm for text data stream has efficiently handled the main issues of data stream clustering namely concept drift, feature evolution, and novelty detection. As it was demonstrated, AIS-Clus algorithm succeeds in detecting novel classes using the concepts of the AIS. The key to detect novel classes is a good threshold for outlier detection. However, higher values of δ will increase the number

of clusters. So, after a fine-tuning of this parameter, experiments demonstrate that our algorithm is robust in considering the rate of incoming data and handling noisy data. In addition to a good threshold, the size of a chunk impacts the efficiency of the data stream clustering algorithm. The chunk sizes for Sanders and HCR are, respectively, defined as 200 and 50. We observe in the evaluation that AIS-Clus detects novel classes with relatively high Recall and Precision values. However, the decline of the Recall and F -measure values can be explained by the fact that AIS-Clus does not classify a number of instances and considers them as outliers. This issue is rapidly managed by the detection of novel class, so novel instances will be included in their appropriate classes.

In addition, relying on the concept of the surviving organisms, performed experiments have demonstrated that our algorithm has managed the evolving nature of data stream and capturing drifting concepts. Adopting the survival factor for each term has effectively handled feature evolution challenge where novel features appear and others disappear as time flows.

We can mention that although stream clustering is an active field of research, no stream clustering evaluation tool exists that offers a suite of implemented stream clustering algorithms and evaluation measures.

6. Conclusion

In this paper, we have proposed a clustering algorithm AIS-Clus for text data stream. The proposed algorithm is based on the Artificial Immune System bio-inspired meta-heuristic. AIS-Clus is an adaptive algorithm that uses the strength of both the clonal selection and the negative selection theories. Those two theories succeeded in adapting existing clusters for novel arriving streams and detecting new occurring concepts. In order to overcome the inherent challenge of novelty detection in flows of data, the cloning process performs an exploration of the searching space to seek for candidate solutions. Then, the mutation mechanism is used to allow a diversification of the candidate solutions and thus a very quick response to the antigens (novel arriving instances). Furthermore, AIS-Clus is a density-based algorithm that succeeded in effectively managing the multi-class detection in the streaming scenario. While most of the existing data stream mining techniques consider only the detection of one novel class at a given time, our framework is able to detect multiple classes simultaneously. The results show that our algorithm adapts quickly and detects novel concepts with a high efficiency rate. In addition to novelty detection issue, the coincidence of the living organism's theory with the problem of concept drift and feature evolution has proved its effectiveness in the context of textual streams.

For evaluation purposes, we have conducted our experiments on Twitter datasets. In our case, tweets are given in a chronological order to consider streaming scenario. Obtained results demonstrate that AIS-Clus succeeded in managing data stream issues, namely, novelty detection, concept drift, and feature evolution.

As a future work, we aim to extend the AIS-Clus to identify evolving user's interests in social streams. Tracking the evolving nature of user's interests remains an open research area and, nowadays, it gains much more attentions within several domains. In particular, identifying user's interest in Cyber-Security domain raises the most attention since it finds out the persons who are likely to become terrorists and a threat to public security.

Acknowledgments

This publication was made possible by NPRP Grant No. 9-175-1-033 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. D. Mena-Torres and J. S. Aguilar-Ruiz, A similarity-based approach for data stream classification, *Expert Syst. Appl.* **41** (2014) 4224–4234.
2. L. Su, H. Lui and Z. Song, A new classification algorithm for data stream, *Int. J. Mod. Edu. Comput. Sci.* **3** (2011) 32–39.
3. V. Kavitha and M. Punithavalli, Clustering time series data stream: A literature survey, *Int. J. Comput. Sci. Inf. Secur.* **8** (2010) 289–294.
4. B. Tidke and R. Mehta, A comprehensive review and open challenges of stream Big Data, in *Soft Computing: Theories and Applications*, Advances in Intelligent System and Computing, Vol. 584 (Springer, 2018), pp. 89–99.
5. M. M. Gabor, Advances in data stream mining, *WIREs Data Min. Knowl. Discov.* **2** (2012) 79–85.
6. A. F. P. Marco, A. C. David, C. Lei and T. Lionel, A review of novelty detection, *Signal Process.* **99** (2014) 215–249.
7. E. R. Faria, I. J. Gonçalves, A. C. de Carvalho and J. Gama, Novelty detection in data streams, *Artif. Intell. Rev.* **45** (2016) 235–269.
8. A. Abid, S. Jamoussi and A. B. Hamadou, Handling concept drift and feature evolution in textual data stream using the artificial immune system, in *Proc. Int. Conf. Computational Collective Intelligence* (2018).
9. C. C. Aggarwal, *Data Streams: Models and Algorithms*, Advances in Database Systems, Vol. 31 (Springer-Verlag, Berlin, 2007).
10. P. Berkhin, A survey of clustering data mining techniques, in *Grouping Multidimensional Data*, eds. J. Kogan, C. Nicholas and M. Teboulle (Springer, Berlin, 2006), pp. 25–71.
11. D. Barbard, Requirements for clustering data streams, *ACM SIGKDD Explor. Newsl.* **3** (2002) 23–27.
12. M. Mousavi, A. A. Bakar and M. Vakilian, Data stream clustering algorithms: A review, *Int. J. Adv. Soft Comput. Appl.* **7** (2015) 1–15.
13. C. C. Aggarwal, J. Han, J. Wang and P. S. Yu, A framework for clustering evolving data streams, in *Proc. 29th Int. Conf. Very Large Data Bases*, Vol. 29 (VLDB Endowment, 2003), pp. 81–92.
14. R. A. Marcel, L. Christiane, M. Marcus, R. Christoph, S. Christian and S. Kamil, StreamKM++: A clustering algorithm for data streams, *ACM J. Exper. Algorithmics* **17** (2012) 2.

15. D. Arthur and S. Vassilvitskii, K-means⁺⁺: The advantages of careful seeding, in *Proc. Eighteenth Annu. ACM-SIAM Symp. Discrete Algorithms* (2007).
16. A. Amineh, W. T. Ying and S. Hadi, On density-based data streams clustering algorithms: A survey, *J. Comput. Sci. Technol.* **29** (2014) 116–141.
17. C. Feng, E. Martin, Q. Weining and Z. Aoying, Density-based clustering over an evolving data stream with noise, in *Proc. 2006 SIAM Conf. Data Mining* (2006).
18. M. Aljibawi, M. Z. A. Nazri and Z. Othman, A survey on clustering density based data stream algorithms, *Int. J. Eng. Technol.* **7** (2018) 147–153.
19. M. Ester, H. P. Kriegel, J. Sander and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proc. Int. Conf. Knowledge Discovery and Data Mining* (1996).
20. C. Yixin and T. Li, Density-based clustering for real-time stream data, in *Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining* (2007).
21. M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan and X. Zhang, Fast memory efficient local outlier detection in data streams, *IEEE Trans. Knowl. Data Eng.* **28** (2016) 3246–3260.
22. G. Joao, I. Zliobaite, A. Bifet, P. Mykola and B. Abdelhamid, A survey on concept drift adaptation, *ACM Comput. Surv.* **46** (2013) 44.
23. I. Zliobaite, M. Pechenizkiy and J. Gama, An overview of concept drift applications, in *Big Data Analysis: New Algorithms for a New Society*, eds. N. Japkowicz and J. Stefanowski (Springer International Publishing, Cham, 2016), pp. 91–114.
24. P. Lindstrom, S. J. Delany and B. Mac Namee, Handling concept drift in a text data stream constrained by high labelling cost, in *Proc. Florida Artificial Intelligence Research Society Conf.* (2010).
25. C. David, A. Les and L. Richard, Improving generalization with active learning, *Mach. Learn.* **15** (1994) 201–221.
26. M. F. Dewan, Z. Li, H. Alamgir, M. R. Chowdhury, S. Rebecca, S. Graham and D. Keshav, An adaptive ensemble classifier for mining concept drifting data streams, *Expert Syst. Appl.* **40** (2013) 5895–5906.
27. J. Z. K. Jeremy and M. A. Maloof, Dynamic weighted majority: A new ensemble method for tracking concept drift, in *Proc. Third IEEE Int. Conf. (Data Mining, 2003)*.
28. I. Koychev, Gradual forgetting for adaptation to concept drift, in *Proc. ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning* (2000).
29. D. Miljkovic, Review of novelty detection methods, in *Proc. 33rd Int. Convention MIPRO* (2010).
30. M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han and B. Thuraisingham, Addressing concept evolution in concept-drifting data streams, in *Proc. 2010 IEEE 10th Int. Conf. (Data Mining (ICDM), 2010)*.
31. C. C. Aggarwal and K. Subbian, Event detection in social streams, in *Proc. 2012 SIAM Int. Conf. Data Mining* (2012).
32. I. Katakis, G. Tsoumakas and I. Vlahavas, Dynamic feature space and incremental feature selection for the classification of textual data streams, in *Proc. ECML/PKDD-2006: Int. Workshop Knowledge Discovery from Data Streams* (2006).
33. C. Mario, D. C. Luigi and S. Claudio, Emerging topic detection on Twitter based on temporal and social terms evaluation, in *Proc. Tenth Int. Workshop Multimedia Data Mining* (2010).
34. M. M. Masud, Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, A. Srivastava and N. C. Oza, Classification and adaptive novel class detection of feature-evolving data streams, *IEEE Trans. Knowl. Data Eng.* **25** (2013) 1484–1497.

35. A. Colorni, M. Dorigo and V. Maniezzo, Distributed optimization by ant colonies, in *Proc. European Conf. Artificial Life* (1991).
36. K. James and E. C. Russell, Particle swarm optimization, in *Proc. IEEE Int. Conf. Neural Networks* (1995).
37. J. D. Farmer, N. H. Packard and A. S. Perelson, The immune system, adaptation and machine learning, *Physica D* **22** (1986) 187–204.
38. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence* (MIT Press, Cambridge, 1992).
39. U. Aickelin and D. Dasgupta, Advances in artificial immune systems, *IEEE Comput. Intell. Mag.* **1** (2006) 40–49.
40. N. Jerne, Towards a network theory of the immune system, *Ann. Immunol.* **125** (1974) 373–389.
41. S. Forrest, A. S. Perelson, L. Allen and R. Cherukuri, Self-nonsself discrimination in a computer, in *Proc. 1994 IEEE Symp. Research in Security and Privacy* (1994).
42. K. Ren-Jieh, C. Sow-Hsin, C. Wei-Chung and T. Chih-Yang, Integration of artificial immune network and *K*-means for cluster analysis, *Appl. Artif. Intell.* **40** (2013) 541–557.
43. N. Tang and V. R. Vemuri, An artificial immune system approach to document clustering, in *Proc. 2005 ACM Symp. Applied Computing* (ACM, New York, 2005).
44. O. Nasraoui, C. C. Uribe, C. R. Coronel and F. Gonzalez, TECNO-STREAMS: Tracking evolving clusters in noisy data streams with a scalable immune system learning model, in *Proc. Third IEEE Int. Conf. Data Mining* (2003).
45. S. Nakatani, Language detection library for Java (2010), <https://github.com/optimaize/language-detector>.
46. L. Gong, G. Zeng and S. Zhang, Text stream clustering algorithm based on adaptive feature selection, *Expert Syst. Appl.* **38** (2011) 1393–1399.
47. E. R. de Faria and A. C. P. de Leon Ferreira Carvalho and J. Gama, MINAS: Multiclass learning algorithm for novelty detection in data streams, *Data Min. Knowl. Discov.* **30** (2016) 640–680.
48. A. Bifet, G. Holmes, R. Kirkby and B. Pfahringer, MOA: Massive online analysis, *J. Mach. Learn. Res.* **11** (2010) 1601–1604.