DEVELOPING NOVEL METHODS ON DATA STREAM CLASSIFICATION AND CLUSTERING FOR ACCURACY IMPROVEMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ENGIN MADEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

SEPTEMBER 2024

Approval of the thesis:

DEVELOPING NOVEL METHODS ON DATA STREAM CLASSIFICATION AND CLUSTERING FOR ACCURACY IMPROVEMENT

submitted by **ENGIN MADEN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Pınar Karagöz
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. İsmail Sengör Altıngövde
Computer Engineering, METU _____

Prof. Dr. Pınar Karagöz
Computer Engineering, METU _____

Prof. Dr. Mehmet Erkut Erdem
Computer Engineering, Hacettepe University _____

Prof. Dr. Ahmet Coşar
Computer Engineering, Ankara Medipol University _____

Prof. Dr. Ferda Nur Alpaslan
Computer Engineering, METU _____

Date:06.09.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Engin Maden

Signature :

ABSTRACT

DEVELOPING NOVEL METHODS ON DATA STREAM CLASSIFICATION AND CLUSTERING FOR ACCURACY IMPROVEMENT

Maden, Engin
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Pınar Karagöz

September 2024, 92 pages

The streaming data from different sources as social media, telecommunication network or credit card processing are accumulated and growing enormously. Thus, it has become more important to produce valuable information from such big data environments. There are specific characteristics of data streams such as continuous flow, high volume, rapid arrival and change of distribution. Due to these characteristics, there are limitations for processing data streams such as limited resource and time and the data can be scanned only once. At this point data stream mining emerges with the streaming version of traditional data mining operations such as clustering and classification.

In this study, data stream classification and short text stream clustering as a specific area of data stream clustering are worked on. Enhancements and novel methods are proposed and their performances are compared with the state of the art methods. For data stream classification, our proposed methods are named as m-kNN (Mean Extended kNN) and CSWB (Combined Sliding Window Based) classifier which is a combination of m-kNN and MC-NN (Micro Cluster Nearest Neighbour). Two new versions of CSWB are also presented, CSWB-e and CSWB-e2, such that our m-kNN

classifier is combined with K* (K-Star) and C4.5, and with K* (K-Star) and Naive Bayes, respectively. For the short text stream clustering, a method named T-GSC (A Two Level Graph Based Short Text Stream Clusterer) is proposed. A survey is also prepared about the current methods in short text stream clustering and classified them with respect to their clustering approaches.

Keywords: Data stream classification, sliding window, hybrid classifier, short text stream clustering, word relation network

ÖZ

VERİ AKIŞI SINIFLANDIRMA VE KÜMELEME ÜZERİNE DOĞRULUĞU ARTIRMAYA YÖNELİK YENİ YÖNTEMLER GELİŞTİRİLMESİ

Maden, Engin
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Pınar Karagöz

Eylül 2024 , 92 sayfa

Sosyal medya, telekomünikasyon ağı veya kredi kartı işlemleri gibi farklı kaynaklardan gelen veri akışı birikmekte ve hızla büyümektedir. Dolayısıyla bu kadar büyük veri ortamlarından değerli bilgilerin üretilmesi daha da önemli hale gelmiştir. Veri akışlarının sürekli veri gelmesi, yüksek hacim, verinin hızla ulaşması ve veri dağılımının değişebilmesi gibi bazı kendine özgü özellikleri bulunmaktadır. Bu özelliklerden dolayı veri akışlarının işlenmesinde sınırlı kaynak ve süre gibi bazı kısıtlar bulunmaktadır ve veriler yalnızca bir kez taranabilmektedir. Bu noktada veri akışı madenciliği, kümeleme ve sınıflandırma gibi geleneksel veri madenciliği işlemlerinin akış versiyonuyla ortaya çıkmaktadır.

Bu çalışmada, veri akışı kümelemesinin özel bir alanı olarak veri akışı sınıflandırması ve kısa metin akışı kümelemesi üzerinde çalışılmıştır. Bazı iyileştirmeler ile yeni yöntemler önerilmiş ve bunların performansı en gelişmiş yöntemlerle karşılaştırılmıştır. Veri akışı sınıflandırması için önerdiğimiz yöntemler, m-kNN (Mean Extended kNN) ve m-kNN ile MC-NN (Micro Cluster Nearest Neighbour) yöntemlerinin birleşimi olan CSWB (Combined Sliding Window Based) sınıflandırıcı olarak adlandırılmış-

tır. Ayrıca CSWB'nin iki yeni sürümü de sunulmaktadır: CSWB-e ve CSWB-e2. Bu yöntemlerde de m-kNN sınıflandırıcımız önce K* (K-Star) ve C4.5 ile sonra da K* (K-Star) ve Naive Bayes ile birleştirilmektedir. Kısa metin akışı kümelemesi için önerdiğimiz yönteme T-GSC (A Two Level Graph Based Short Text Stream Clusterer) adını verdik. Ayrıca kısa metin akışı kümelemesinde güncel yöntemler hakkında bir araştırma çalışması hazırlanmış ve bu yöntemleri kümeleme yaklaşımlarına göre sınıflandırılmıştır.

Anahtar Kelimeler: Veri akışı sınıflandırma, kayan pencere, hibrit sınıflandırıcı, kısa metin akışı kümeleme, kelime ilişki ağı

to all my loved ones...

ACKNOWLEDGMENTS

First of all, I would like to thank to my thesis advisor, Prof. Dr. Pınar Karagöz, for her guidance, support and motivation she provided throughout my research. Also, I would like to thank to Prof. Dr. İsmail Sengör Altıngövde and Prof. Dr. Mehmet Erkut Erdem for their suggestions, comments and contributions. I also very much like to thank to my family for their support and patience.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
CSWB	Combined Sliding Window Based
K*	K-Star
T-GSC	Two Phase Graph Based Short Text Stream Clusterer
m-kNN	Mean extended kNN
gEBoost	graph Ensemble Boosting
ELPM	Extreme Learning Machine
ADWIN	ADaptive WINdowing
PAW	Probabilistic Adaptive Window
VFDT	Very Fast Decision Tree
OBA	OzaBagADWIN
CSC	Cost Sensitive Classifier
MOA	Massive Online Analysis
kNN	K-Nearest-Neighbours
m-kNN	mean extended kNN
VHT	Vertical Hoeffding Tree
SAMOA	Scalable Advanced Massive Online Analysis
DP-BMM	Dirichlet Process Biterm Based Mixture Model
OSDM	Online Semantic Enhanced Dirichlet Model
LAST	Lifelong Learning Augmented Short Text Stream Clustering
LDA	Latent Dirichlet Allocation
HAR	Human Activity Recognition
D3CAS	Distributed Clustering Algorithm Applied To Short Text Stream Processing

CBTC	Correlated Bursty Term Clustering
CF	Cluster Feature
OSGM	Online Semantic Enhanced Graphical Model
NDML	Non-Parametric Dirichlet Model With Language Translation Component
DCT	Dynamic Clustering Topic Model
DCSS	Dynamic Clustering For Short Text Stream Based On Dirichlet Process
ICVE	Incremental Clustering With Vector Expansion
CA	Cluster Abstract
FCTGS	Fast Core Term Group Search
NDCG	Normalized Discounted Cumulative Gain
TREC	Text REtrieval Conference

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The amount of data that is obtained from different sources such as telecommunication, credit card usage and social media is getting higher and it has become important to extract valuable information from such data sources. There are several characteristics of data streams that can be summarized as follows:

- The flow of data is continuous.
- The volume of data is high.
- The arrival rate of data is rapid.
- The distribution of data may change over time.

Conventional data mining tasks such as classification and clustering can be applied on streaming data, as well. However, as given in [1], there are several limitations for a stream classifier or clusterer:

- During streaming, instances can be examined only one by one, each can be examined only once.
- The memory resource is limited in comparison to the amount of streaming data, hence it is not feasible to make batch processing.
- It is necessary to process data fast in order to respond (near) real time.

In order to work with infinite and continuous data, *sliding window* mechanism is a commonly used approach. *A window* can be defined as a set of stream elements within a certain time frame. Sliding windowing has two basic types: *time-based* and *count-based*. In count-based sliding windows, the borders of windows are specified with the counts of instances lying inside the windows. On the other hand, in time-based sliding window mechanisms, the borders are determined by using a time interval [2].

There are two main parts in our study as data stream clustering and classification. For data stream classification, an enhancement of kNN is proposed where it is applied in a sliding window approach and it is called m-kNN. As the second enhancement, a combined version of m-kNN, traditional kNN and Naive Bayes is applied in sliding window mechanism, and this method is called CSWB (Combined Sliding Window Based) classifier. Also, new versions of CSWB such as CSWB-e and CSWB-e2 are proposed where new classifiers are used together with m-kNN.

For the second part of our study which is about data stream clustering, short text stream clustering is worked on specifically. The latest techniques and algorithms in addition to the mostly referred ones about short text stream clustering are reviewed. These techniques are also classified into groups according to their clustering approaches. The literature is comprehensively reviewed and the algorithms published in the last few years are filtered. The state of the art algorithms are included which are most commonly referred ones in the literature. These algorithms are used as baseline by these methods and used to measure their clustering performance. Few number of batch processing based short text clustering methods from the literature are also included in this survey. These studies employ recent approach, mostly deep neural network based solutions, which carry potential to be adapted for short text streaming clustering as well. The characteristics of the proposed methods are described in detail and summarized as a table with respect to these groups. The datasets used by the analyzed methods for performance evaluation and the clustering quality measures employed by the authors are reviewed. The clustering performances reported by the studies are also organized and presented in a comparative way with respect to the datasets and the measures used. As a result of this part of study, a survey is prepared about current short text stream clustering techniques [3].

There are some challenges about short text stream clustering and the main ones are about the characteristics of streaming data where the data flow is so high and it can not be stored. Therefore, the methods for short text stream clustering can not pass over the data more than once. Also, the time to cluster the incoming instance is limited and the execution time of the proposed methods are also important.

Another challenge is that the number of clusters should not be predefined and it should be dynamic due to the nature of streaming data. In addition the the number of clusters, there may also be other parameters for the proposed models and the selection of these parameters can effect the results of clustering. Thus, it is also important to develop a model with minimum number of parameters required. Term ambiguity and the characteristics of short text documents such as the limitations of characters can be considered as other challenges about short text stream clustering.

After the survey study, short text stream clustering is worked on and the motivation of this work is to develop a graph based method for short text stream clustering utilizing neural model based embeddings to be able to capture the semantics of the content better. The challenges of short text stream clustering is considered while developing our method.

1.2 Proposed Methods and Models

The characteristics of data stream processing which are mentioned before lead to limitations such as limited time and resources to process the data or the data can be processed only once. To overcome these challenges, novel methods on data stream classification and clustering are proposed in this thesis work.

For data stream classification, an enhancement of kNN is proposed where it is applied in a sliding window approach. To deal with infinite data streams, windowing is commonly used in stream processing. A window can be defined as a set of stream elements within a certain time frame. There are two common types of sliding windows, which are time-based and count-based. In time-based sliding windows, a time interval is used for specifying the borders of the window while the count of instances specifies these borders in count-based ones [2]. In this paper, count-based sliding window is

used to process the data streams. Our method is called m-kNN (mean extended kNN) and in this method traditional kNN is applied in a sliding window. Additionally, one of the k-nearest neighbors is obtained among the centroids (the mean values of the features of the classes). The class centroid, which is the most similar to the incoming instance, is considered as the k^{th} nearest neighbor. As the second enhancement, a combined version of m-kNN, traditional kNN and Naive Bayes is applied in sliding window mechanism, and this method is called CSWB (Combined Sliding Window Based) classifier.

Two new versions of CSWB are proposed where new classifiers are used together with m-kNN. In CSWB-e, m-kNN is combined with K* (K-Star) and C4.5. In CSWB-e2, in addition to m-kNN, K* and Naive Bayes classifier are used. The motivation for including Naive Bayes in CSWB-e2 is based on our observation [4] that it contributed well to the performance of CSWB.

The first newly added classifier is K*, which is a nearest neighbor based classifier [5]. In K*, entropy is used as the distance measure. The approach to calculate the distance between two instances is inspired from information theory, and the main idea is that this distance can be viewed as the complexity of transforming one instance into another.

The second classifier that included in our hybrid model is C4.5 [6], which is a decision tree based classifier, and is the successor of the ID3 algorithm. The algorithm uses the concept of information entropy for node selection.

For each node of the decision tree, C4.5 selects the attribute that most effectively splits list of the training data samples. To this aim, C4.5 selects the attribute with the highest information gain. This process is applied recursively on the partitioned data.

As in CSWB, in CSWB-e and in CSWB-e2, two different voting schemata may be considered in the assembler. In [4], it was already shown that the voting under *current accuracy* model performed well. Hence in our experiments, voting is used with current accuracy schema among the participating classifiers.

For the second part of our study, short text stream clustering as a specific area for data stream clustering is worked on. Firstly, a deep literature review is made and

a survey is prepared about the recent methods in short text stream clustering. Then a graph based method for short text stream clustering is developed where a neural model based embeddings are used. At this point it is aimed to capture the semantics of the content better. In addition, a two level clustering approach is proposed to refine the clustering further. For the first phase, sentence embeddings are applied for the short text documents and for the second phase clustering is applied on the node embeddings obtained from the clusters of previous phase. Our method is called as *Two Phase Graph Based Short Text Stream Clusterer (T-GSC)*. The datasets used in this study and the source code of applications are available at GitHub[1].

1.3 Contributions and Novelties

This study involves developing new methods for traditional data stream mining tasks such as classification and clustering, short text stream clustering specifically. The contributions of our study in data stream classification can be summarized as follows:

- m-kNN is presented to associate the behaviour pattern coming from the past and current state of data.
- An ensemble classifier called CSWB is developed as the combination of our m-kNN with traditional kNN and Naive Bayes classifier.
- These two enhancements are evaluated on several data sets from different domains. The experiments reveal that the enhancements provide higher accuracy values for several of the data sets, and have potential for further improvement.
- The first new version of Combined Sliding Window based Classifier, *CSWB-e*, combines m-kNN with two other methods from the literature, K* and C4.5 algorithms, in a sliding window. The other new version, *CSWB-e2*, is composed of m-kNN, K* and Naive Bayes.
- The performance of *CSWB-e* and *CSWB-e2* are analyzed in comparison to the individual stream classifiers and the previous version, *CSWB*.
- The correctness of *m-kNN* is analyzed further on different datasets.

[1] https://github.com/enginmaden

For the second part of our study which is about short text stream clustering, the contributions can be summarized as follows:

- The most recent methods and algorithms on short text stream clustering which are published in the recent years are reviewed.
- The most widely referred methods are also covered in this survey in addition to these recent methods.
- The methods are classified with respect to their short text stream clustering approaches used in the studies.
- The proposed methods are comparatively analyzed according to the characteristics, datasets or clustering quality measures.
- In this study, a graph based short text stream clustering is proposed. There are only very few graph based approaches in literature for short text stream clustering. The proposed method aims to use the graph structure more effectively than the prior approaches.
- Using a two level clustering process in our method provides a finer grained grouping leading to increase in the accuracy of clustering.
- A hierarchical clustering method is used for clustering phases. This reduces the number of parameters required for our model.
- The proposed model combines the power of sentence embeddings and node embeddings for short text stream clustering.

1.4 The Outline of The Thesis

The rest of the thesis is organized as follows: Chapter 2 involves the background and related concepts on the subject. Enhancements are proposed methods for data stream classification are given in details with experiments and analysis in Chapter 3. In Chapter 4, the survey work explaining the recent methods on short text stream clustering and also an analysis with comparison is included. In addition, proposed

method for short text stream clustering is explained in Chapter 4. Finally, conclusions are given in Chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

A data stream can be defined as a continuous flow of data which is infinite. The characteristics of data stream can be given as follows:

- Data flows continuously.
- The amount of data is huge.
- The data may arrive rapidly.
- The distribution of data may evolve in time.

The traditional tasks of data mining such as classification and clustering cab be applied to data streams. However, there are some limitations due to the nature of data.

- The process of data can be executed only once.
- The memory which can be used to process data is limited.
- The data must be processed fast to generate a response in real time.

Concept drift is another term about data stream processing and it can be defined as the change of statistical features of data stream in time. For instance, in clustering if a new cluster is generated or disappeared this can be considered as a concept drift [7]. Four types of concept drift such as sudden, gradual, incremental or recurring concept drifts are defined in [8]. Concept drift is a specific area to study in data stream mining but the proposed methods in this thesis work do not directly aim to handle concept drift.

2.1 Data Stream Classification

Classification can be defined as predicting the class labels of instances in a dataset by using the features and data stream classification can be considered as the streaming variation of classification task where it must be executed by taking into account the limits of data stream process.

In literature, there is a variety of studies and algorithms developed to deal with data streams in classification, ranging from distance based nearest neighbor approaches, decision tree based solutions to assembler methods that aim to improve the classification performance by combining several methods. One of such assembler method is gEBoost (graph Ensemble Boosting), which classifies imbalanced graph streams with noise. This method partitions the graph stream into chunks where each chunk contains noisy graphs having imbalanced class distributions. A boosting algorithm for each chunk is proposed to combine the selection of discriminative sub-graph pattern. Combination of the chunks forms a unified framework as a learning model for graph classification [9].

Fuzzy methods can be also utilized in data stream classification. In [10], Silva et al. propose an extension for On-Demand classification algorithm developed by Aggarwal et al. [11]. They include the concepts of fuzzy sets theory and they aim to improve the adaptability of the classification process to changes in data stream.

Yang et al. present an ensemble Extreme Learning Machine (ELM) that includes a concept drift detection method. The authors utilize online sequence learning strategy in order to handle gradual concept drift. In their work, it is reported that their ELM algorithm improves the accuracy results for classification and also can adapt to new concepts in a shorter time [12].

In [13], Wozniak et al. propose a data stream classifier based on sliding windows. Their method contains a novel classifier training algorithm and it includes a *forgetting mechanism*. The authors choose *interesting* objects to be labeled by using active learning.

ADWIN (ADaptive WINdowing) also maintains a window for processing data streams

but the window has a variable size. ADWIN2 is the improved version for memory usage and time efficiency. In [14], it is proposed to make a combination of ADWIN2 and the Naive Bayes classifiers.

PAW (Probabilistic Adaptive Window) is another windowing based stream learning approach [15]. It contains older samples from dataset as well as the most recent ones. This leads a quick adaptation to new concept drifts and also it enables to maintain information about past concept drifts.

In [16], Bifet et al. propose STREAM DM-C++, which is a framework where decision trees for data streams area implemented in C++. This framework is easy to extend and it includes powerful ensemble methods.

Data stream classification methods can be applied to several domains. Among them, finance includes several interesting sub-problems. Credit card fraud detection or risk assessment for credit are well-known examples of financial application areas for data stream classification. Sousa et al. propose a new dynamic modeling framework which can be used in credit risk assessment. Their framework extend the credit scoring models and they apply it to a real-world data set of credit cards from a financial institution in Brazil [17].

In [18], Shi et al. consider identification of human factors in aviation incidents as another application area for the data stream classification methods. They use four data stream algorithms, Naive Bayes, Very Fast Decision Tree (VFDT), OzaBagADWIN (OBA) and Cost Sensitive Classifier (CSC), which are implemented in MOA (Massive Online Analysis) framework [19].

2.1.1 kNN (K-Nearest-Neighbours)

For the data stream classification part of our study, we proposed a method called mkNN (mean extended kNN) and in this method traditional kNN is applied in a sliding window.

kNN is a well-known method used in classification and regression in data mining. It has a non-parametric and supervised learning approach. In kNN, there is a parameter

k for the classification model and a distance metric such as *Euclidean Distance* is selected to determine these k nearest neighbours for the incoming instance. In kNN, the classification and selection of the label is made by using the previously classified instances in dataset.

Our methods for data stream classification, *m-kNN* and *CSWB*, are sliding window based methods and they are similar to PAW or ADWIN in this aspect. The main difference of our m-kNN method is that we select a nearest neighbor from the historical data and thus, we can associate the past and current behaviour of stream. On the other hand, CSWB an assembler for data stream classification. It differs from other ensemble method in the way singleton classifiers are combined. In CSWB, m-kNN is combined with other classifiers in the literature in a sliding window.

For the data stream classification part of our study, we proposed a method called m-kNN (mean extended kNN) and in this method traditional kNN is applied in a sliding window.

2.1.2 MC-NN (Micro-Cluster Nearest Neighbour)

For accuracy comparison, we implemented the streaming version of Micro-Cluster Nearest Neighbour (MC-NN) as a state-of-the-art streaming classifier in the literature [20].

MC-NN utilizes statistical summaries of the data stream. In this method, statistical summaries are processed incrementally and it aims to handle concept drifts as well. Additionally, MC-NN has a parallel version, in order to improve efficiency.

MC-NN uses micro-clusters in the nearest neighbor approach. There are two measures for each micro cluster in MC-NN as follows:

- **Error Count:** It is the count of misclassified instances for the micro cluster. It is initially 0 and incremented by 1 for incorrect classifications. Similarly, it is decremented by 1 for correct classifications.
- **Participation Percentage:** It is the degree of recency for the micro-cluster and calculated according to the timestamp of the instances in the micro-cluster

when they participate.

The flow of the MC-NN can be summarized as follows:

- Calculate the centroids of the current micro-clusters (Each micro-cluster contains instances belonging to the same class).
- Calculate the Euclidean distance between each centroid and the new distance.
  - Assign the instance to the nearest micro-cluster.
  - Check if the assigned value is equal to the actual class label of the instance.
- If the classification is correct, decrease the error count of the micro-cluster by 1.
- Otherwise, add the instance to the nearest Micro-Cluster that matches the instance's class label.
  - Increment the error count of both involved Micro-Clusters.
  - If the error count of any of these two micro-clusters exceeds the error threshold, calculate the variance value for each attribute and split the micro-cluster according to this attribute.
  - Calculate the participation percentage for each micro-cluster.
- Delete any micro-cluster having participation percentage lower than the performance threshold.

When the error count exceeds the error threshold, the variance of each feature is calculated. For the feature having the maximum variance, the mean value is calculated. According to this mean value, the cluster is split into two new micro clusters.

Recency of current micro clusters refers to whether a micro cluster is old, and hence should be deleted or not. It is determined through *participation percentage*, which is the ratio of the sum of timestamps of instances in micro cluster to the real triangular number of micro cluster. The *real triangular number* of a micro cluster is calculated as the difference between the triangular number for current timestamp and the initial

triangular number for the micro cluster. The initial triangular number is calculated with the timestamp value for the micro cluster when the micro cluster is generated and the first instance is assigned to this micro cluster [20]. If the participation percentage for a micro cluster is lower than a given threshold, the micro cluster is deleted.

The other method from literature we use for accuracy comparison is Vertical Hoeffding Tree (VHT). It is implemented in Apache SAMOA (Scalable Advanced Massive Online Analysis) framework [1]. VHT is a distributed classifier and it uses vertical parallelism. Vertical parallelism partitions the instances according to the attributes and enables parallel processing. The number of attributes in each partition is decided by the division of the number of total attributes and the number of partitions. The algorithm is executed in parallel on the attributes in each partition and the best local attribute for split operation is selected. Following this, the results of these parallel computations are combined in order to select the best global attribute to split, and to grow the tree [21].

2.2 Data Stream Clustering

2.2.1 General Approaches For Data Stream Clustering

In clustering, the idea is to divide the data into partitions called clusters. The most important point is to have a maximized similarity within a cluster, and to have a minimized the similarity between the clusters [22]. Data stream clustering is a type of clustering task where a data stream is processed in limited time and memory due to the limitations of data stream mining. Data stream clustering methods can be grouped into five groups as:

- **Partitioning Based Methods:** These methods partition data into fixed and previously given number of clusters. They have an iterative approach to cluster the data. StreamKM++ is an example of this kind of data stream clustering algorithms. In this method, a weighted sample of the data stream is computed and clustering is operated by using k-means++ [23]. Another example of partition-

[1] https://samoa.incubator.apache.org

ing methods is Adaptive Streaming k-means [24]. This algorithm is composed of two parts as initialization phase and continuous clustering phase. It is mainly developed to overcome the two typical problems for partitioning based methods. These problems can be given as follows:

  - An input parameter k is needed.
  - Adapting concept drift is difficult for partitioning based methods.

- **Hierarchical Methods:** They use a hierarchical tree of clusters in a divisive or agglomerative manner. BIRCH is an example of hierarchical methods and it is the abbreviation of Balanced Iterative Reducing and Clustering using Hierarchies[25]. BIRCH firstly generates a summary of the dataset and makes clustering on this summary instead of the whole dataset. The Online Divisive-Agglomerative Clustering (ODAC) is another hierarchical method and it enhances the ability about concept drift detection. ODAC has an agglomerative phase and utilizes a correlation-based dissimilarity measure between time series in a data stream.

- **Density Based Methods:** This type of methods aim to identify the identity profile of the incoming data stream. They utilize a summary of the streaming data and use micro clusters with feature vectors. These micro clusters are merged in time and final clusters are obtained. DenStream [26] is a density based method and there are different types of micro clusters such as dense or core micro clusters which are used to summarize the clusters with arbitrary shape and potential core micro clusters and outlier micro clusters. Another example is D-Stream which can be considered as a framework for clustering streams using a density based approach [27].

- **Grid Based Methods:** They generate a grid structure and calculate the densities for each cell . The data stream is clustered according to these densities and they obtain the cluster centroids. GCHDS (Grid Based Clustering Algorithm for High Dimensional Data Streams) [28] and DGClust (Distributed Grid Clustering) [29] are examples of grid based methods. GCHDS uses high dimensional data stream and analyzes the data distribution to produce the clusters where it can generate the form structure in a short time period. DGClust is

designed for sensor data and the clustering of data is operated at a fixed interval.

2.2.2 Short Text Stream Clustering

The content of section is adapted from our survey work [3]. Short text stream clustering is a specific area of data stream clustering. It can be used to discover trends in real-time or news recommendation. In addition to the above-mentioned limitations of data stream clustering, there are several other challenges in short text stream clustering such as the limited number of words in the content, the evolution of data and sparse representations. Social media applications such as Twitter (renamed as X[2]) and Facebook[3] are useful resources to apply short text stream operations on. In addition to these platforms, resources like Google News[4] or Stack Overflow[5] are the mostly utilized data sources for short text stream clustering.

There are some terms such as Dirichlet Process, LDA (Latent Dirichlet Allocation) and Dirichlet Process which are generally mentioned in short text stream clustering approaches.

Latent Dirichlet Allocation (LDA) [30] is the basis for these models and it has an assumption that text documents have a collection of words used to determine the topics. In LDA, each document can be assumed as a distribution of topics and each topic can be evaluated as a distribution of words.

Given k as the number of topics, α and β as hyper parameters for the model, as the first step, LDA loops through the documents and randomly assigns each word in the document to one of the topics. Then a sampling is operated and the sampled word is removed from the topic assignment. In addition, the count for the document and topic related with the sample is decremented and the followings are calculated:

- The ratio of documents assigned to a topic for a given word (by using Eq. 2.1)

[2] https://twitter.com/
[3] https://www.facebook.com
[4] news.google.com
[5] https://stackoverflow.com

$$P(t_k|d_i) = \frac{n_{ik} + \alpha}{N_i - 1 + k\alpha} \tag{2.1}$$

where d_i: i^{th} document, t_k: k^{th} topic, n_{ik}: number of words in i^{th} document and k^{th} topic, α: hyper parameter for the model and N_i: number of words in i^{th} document

- The ratio of words in a document assigned to topic (by using Eq. 2.2)

$$P(w_j|t_k) = \frac{m_{j,k} + \beta}{\Sigma_{j\epsilon V} m_j, k + V\beta} \tag{2.2}$$

where d_i: i^{th} document, t_k: k^{th} topic, $m_{j,k}$: assignment of word w_j to k^{th} document, V: vocabulary and β hyper parameter for the model.

- Finally, by using these values, conditional probability is calculated for the word w_j (by using Eq.2.3).

$$P(w_j|t_k, d_i) = P(t_k|d_i).P(w_j|t_k) \tag{2.3}$$

The topic having the maximum value for the word w_j is found and the word is assigned to this topic. These steps are repeated for all of the words in all documents for the given number of iterations.

Dirichlet Process. There are various LDA based methods proposed for short text stream clustering but they have the lack of ability to handle topic drift problem. Dirichlet Process based methods appear to solve this problem. Dirichlet Process is defined as a non-parametric stochastic process to model the data. It can be considered as a distribution over distributions. The name Dirichlet process comes from having Dirichlet distributed finite dimensional marginal distributions [31]. The process is to draw a sample from base distribution and each sample itself is also a distribution. The process of drawing sample is controlled by a hyper parameter α.

Cosine Similarity. Some of the short text stream clustering methods generate vector representations for short documents with the help of techniques such as word embeddings and they calculate the similarity values such as Cosine Similarity between these vectors and topics or clusters. *Cosine similarity* can be considered as the measure of similarity between two vectors of an inner product space. It is calculated by the cosine of the angle between two vectors and indicates the orientation for these two vectors. As a result, this value is used to determine the document similarities.

Given two vectors, A and B, including attributes, the cosine similarity, $\cos(\Theta)$, is calculated as given in Eq. 2.4.

$$\cos(A, B) = \frac{AB}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2}\sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.4}$$

Word Relation Network. Another group of short text stream clustering methods is word relation network based methods. They represent the current state of the stream as a *word relation network*. Here, given word relation network graph as $G < V, E >$, the words constitute set of nodes *V* weighted by its corpus-level term frequency. Also each edge *E* is weighted by its corpus-level term co-occurence frequency between the two nodes [32].

In the literature, there are several approaches used in short text stream clustering. Some of these methods use Dirichlet Process which can be defined as a non-parametric stochastic process to model the data, and it is a kind of distribution over distributions [3]. The methods in this group such as Dirichlet Process Biterm Based Mixture Model (DP-BMM) [33] and Online Semantic Enhanced Dirichlet Model (OSDM) [34], and other methods like Lifelong Learning Augmented Short Text Stream Clustering (LAST) [35] and MStream/MStreamF [36].

They utilize co-occurence of words or calculate the probabilities of assigning the document to an existing cluster or constructing a new cluster with the help of frequency of each word. Graphs are useful structures to model the relation between objects in a dataset. The degree of relation can also be represented with the edge weights in

graphs. This representative power of graphs can also be used for clustering operations. However, the number of studies utilizing graphs for short text stream clustering operations is limited in the literature. One of the graph based methods is EWNStream [37] and EWNStream+ [32] is an extension of it. These methods construct a bi-weighted word relation network where the graph is both node-weighted and edge-weighted. They use the frequencies and co-occurence of the words. EWNStream+ is an enhancement over EWNStream where it has a decay policy about the outdated instances and also uses a new closeness metric on the bi-weighted word relation network. While these methods report successful results in terms of accuracy, they do not adopt recent approaches such neural network based solutions into their clustering process.

The number of studies and proposed methods about short text stream clustering is increasing due to immense use of social media in recent years. In this section, the methods and algorithms proposed for short text stream clustering in recent years are described in detail. However, there are several earlier studies that form the basis for short text stream clustering and that have been referred to for comparison. Therefore, initially we present a summary of such studies.

One of these algorithms is a density based short text stream clustering method, Distributed Clustering Algorithm Applied To Short Text Stream Processing (D3CAS). As proof-of-concept study, it is also realized on Spark [38]. D3CAS is a density based clustering method and it uses micro clusters similar to the traditional data stream clustering methods such as DenStream [26] and CluStream [39]. D3CAS utilizes a 7-tuple representation for micro clusters. In this representation, information about the micro cluster, including weighted linear sum of the data, weighted quadratic sum of the data, weight of micro cluster, number of instances, the dimensionality of micro cluster, the timestamp when the micro cluster is generated and the timestamp when the micro cluster is last updated, are stored. D3CAS has an online and an offline phase. In the online phase, when a new instance arrives, it is included in a micro cluster if it is within the radius value. Otherwise a new micro cluster is generated. For the offline phase micro clusters are revised and combined by usiing DBSCAN [40] algorithm.

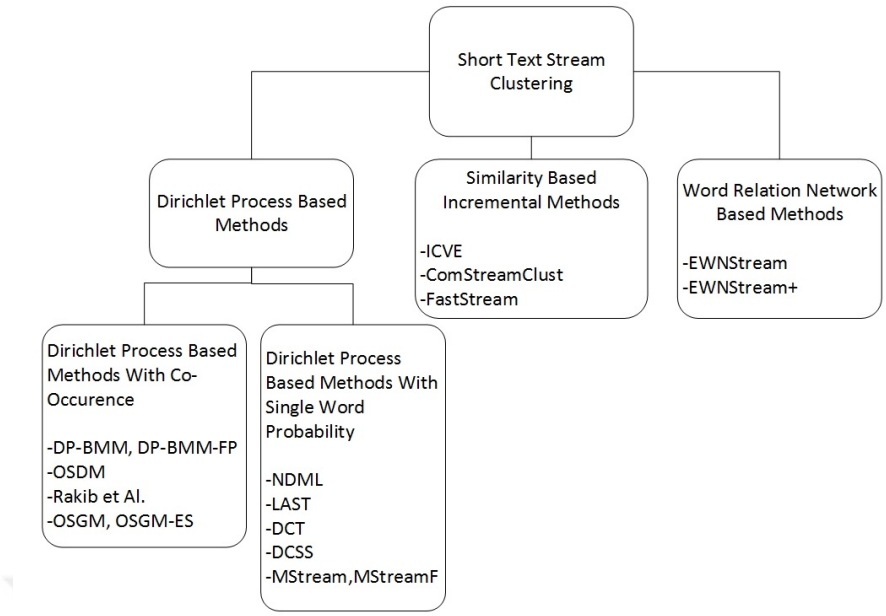

Figure 2.1: Short Text Stream Clustering Methods

Another proposed method in this context, Correlated Bursty Term Clustering (CBTC), uses bursty terms and their co-occurences to improve text stream clustering in [41]. Bursty terms and burst detection are one of the most interesting research areas about data stream clustering. In this method, firstly bursty term detection is performed with the help of Kleinberg's two state automaton model [42]. After that, a bursty term correlation graph is constructed. An edge weight is assigned between terms, if the time period of bursty terms are intersected and if the terms occur together in a document in this intersection period. By this way, bursty term groups are obtained in this correlation graph. The documents in stream containing at least one of these bursty terms and appearing in the period of these bursty terms are assigned to the corresponding group. Cluster representatives are generated with respect to these groups. In order to lower the number of cluster representatives to the desired parameter count, agglomerative clustering is used. Finally a spherical k-means clustering [43] is applied on the generated clusters by using the centroids of the clusters as the cluster representatives.

Short text stream clustering methods covered here are classified into three groups as Dirichlet Process based methods, similarity based and incremental methods and

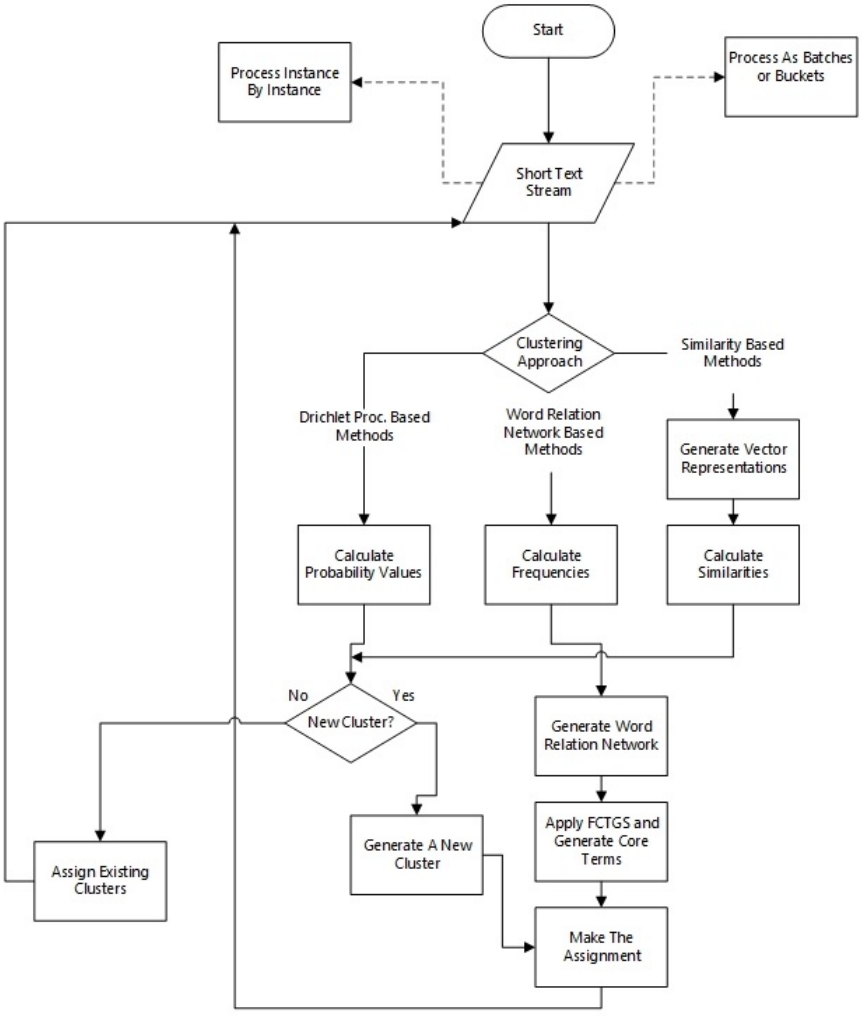

Figure 2.2: Short Text Stream Clustering Process

word relation network based methods. These groups are illustrated in Figure 2.1. The first group of methods calculate the probabilities about assigning the incoming text to current clusters or generating a new cluster. The second group is similarity based and incremental methods. These methods generate a vector representation for the short documents and calculate the similarity between the text and the current clusters. They utilize an incremental approach and use the similarity values such as *cosine similarity* to make assignments. The final group is word relation network based methods. These methods generate a word relation network with the help of frequency and co-occurence values to operate clustering. They use the words as vertices of the graph and frequencies as the labels for the vertices. In the graph, if both of the words

occur together an edge is constructed between their vertices and the frequency of co-occurence is given as the edge value. As a summary, the general flow of short text stream clustering can be illustrated as in Figure 2.2.

From the perspective of their approaches, these methods are summarized in Table 2.1. According to this table, half of these methods use term co-occurence in clustering, and majority of the methods process the incoming short text stream instance by instance whereas some of them handles the stream as batches of instances or buckets.

Table 2.1: Characteristics of Methods. Clustering approaches are abbreviated as Dirichlet Process based (D), Similarity Based Incremental (S), Word Relation Network Based (W)

Method	Uses Co-Occ.s	Co-Occ. Pattern	Proc. Appr. For Data Stream	Uses CF Vectors	Clustering Approach
DP-BMM [33]	Yes	Biterms	Instance by instance	Yes	D
DP-BMM-FP [33]	Yes	Biterms	As batches of instances	Yes	D
OSDM [34]	Yes	Co-occurence matrix	Instance by instance	Yes	D
Rakib et Al. [44]	Yes	Frequent Word Pairs	As batches of instances	Yes	D
NDML [45]	No	-	Chunks (time window or fixed size window)	Yes	D
LAST [35]	No	-	Instance by instance	Yes	D
MStream-MStreamF [36]	No	-	As batches of instances	Yes	D
DCT [46]	No	-	Instance by instance	No	D
DCSS [47]	No	-	Instance by instance	No	D
OSGM, OSGM-ES [48]	Yes	-	Co-occurence matrix	Yes	D
ICVE [49]	Yes	Co-occurence vectors	Buckets	No	S
ComStreamClust [50]	No	-	Instance by instance and time slot based (day/minute)	No	S
FastStream [51]	Yes	Unigrams, bigrams, biterms	Instance by instance	Yes	S
EWNStream [37]-EWNStream+ [32]	Yes	Co-occurence counts	As batches of instances	No (Cluster Abstract-CA is used)	W

2.2.2.1 Dirichlet Process based Methods

In this section, the recent short text stream clustering methods which uses Dirichlet Process based models are reviewed in detail. These methods are further divided into two sub-groups as *Dirichlet Process Based Methods with Co-occurence* and *Dirichlet Process Based Methods with Single Word Probability*. The methods in the first sub-group consider the co-occurence of the words in documents such as biterms. On the other hand, methods in the latter sub-group generate a new cluster for incoming instances or assign the instance to an existing cluster. These methods make this assignment by calculating the probabilities for these two situations and they handle each word individually. These methods do not utilize the co-occurences of the words.

The methods in first group which are **Dirichlet Process based methods with Co-occurence**, take the co-occurences of the words, such as biterms, into account. They

calculate the probabilities of generating a new cluster for an incoming instance or assigning it to one of the existing clusters in their models by using co-occurences. One of such short text stream clustering methods is Dirichlet Process Biterm Based Mixture Model (DP-BMM) [33], which aims to solve the sparsity problem for short text streams through *word-pair (biterm)* based approach. The biterms are word tuples where the order of terms is unimportant. The first step of DP-BMM is the extraction of biterms in short texts. Prior to this, stop-word removal and stemming operations are applied for the pre-processing of short texts. The basic assumption of DP-BMM method is that all of the biterms in a short text are assigned to the same topic. Generally, the number of words in short-texts in social media is so few that the total number of generated biterms stay within reasonable amounts. In DP-BMM, biterm clusters are represented with *Cluster Feature* (CF) vectors. The CF vectors are updated for adding a new document to a cluster or removing a document from the cluster. These operations are called addition/deletion property in [34] and addible/deletable property in [35]. Here, a CF vector contains the list of biterms, the number of documents and the number of words in a cluster. For an incoming short text, the probability to be assigned to an existing cluster or a new cluster is calculated. Here, the biterms are taken into account in these calculations. Finally, the cluster assignment is performed with respect to these calculated probabilities. DP-BMM has another version with *forgetting property*, called DP-BMM-FP. In this version, the short text stream is handled as batches. The maximum number of batches to be saved is given as an input parameter in [33]. If the total count of the batches exceeds this value, the outdated batches are removed from the model. During this removal operation, the CF vectors are updated. To sum up, DP-BMM uses a Dirichlet Process and suggests to solve sparsity problem by using biterms. It also handles the short text streams by adding a forgetting property to the model which aims to solve the topic drift problem.

Another Dirichlet Process based method in this group for short text stream clustering is Online Semantic Enhanced Dirichlet Model (OSDM) [34]. The word *online* in the name indicates that the algorithm processes the short text stream instance by instance. In the algorithm, for handling semantic information, the co-occurence of the words is taken into account. Kumar et al. mention that their method integrates semantic information to model based clustering and they claim that their method is the first

to cope with term ambiguity problem efficiently. In OSDM, semantic information is modeled through co-occurence matrix of the words in a cluster. They also use a decay factor to remove old and outdated clusters from their model. The method represents the clusters with CF vectors, which include the number of documents, the frequency of words, the co-occurence matrix for words, the number of words, the weight and the last update timestamp value for the cluster. OSDM uses Dirichlet Process model and uses the calculated probabilities to assign an incoming short text to an existing cluster or generate a new one. The short text is assigned to the cluster with the highest probability. For each timestamp, the outdated clusters are determined by using the decay factor and the decay function. The CF vectors are updated after the removal of the old clusters.

Another biterm based method is proposed in [44]. The main idea in this approach is working with frequent word pairs. This method processes the data stream in batches. As the first step, word pairs are extracted from short texts. The average frequency of the word-pairs are calculated and the ones which exceed the pre-defined threshold value are considered as *frequent word pairs*. The model assumes that a frequent word pair will not co-exist with another word pair in a cluster. This means that a document should have only one frequent word pair. The clusters are obtained in this way with respect to the frequent word pairs. The CF vectors contain word frequencies, number of documents and number of words in cluster. The list of frequent word pairs coming from the previous batch is merged with the current one and the documents containing more than one frequent word pair are discarded in this phase. In order to obtain more accurate results, the non-distinctive words are deleted from the model on the basis of their entropy values. After deleting such words, CF vectors are updated.

In the next phase, documents were not clustered in the previous step are clustered by using the MStream algorithm [36]. The method also applies outlier detection by a graph based approach. Here, in the conceptual graph, if both of any two words occur together in a document, an edge is formed connecting the vertices of these two words. The vertices in the smallest connected components are considered as outliers. Here, a connected component is a kind of sub-graph where each pair of its nodes are connected. This means there is a path from any node to any node of the connected component. After detecting the outliers, cosine similarity values are calculated with

these outliers and cluster centers are updated.

As the first step of re-assignment of outliers, the sum of mean and standard deviation of these similarity values are calculated. Then if the maximum similarity value for an existing cluster exceeds this sum, the outlier is assigned to that cluster. Otherwise, a new cluster is generated. Therefore the outliers are assigned to clusters with dynamic thresholds. Appearance of an outlier in data stream may indicate a concept drift. According to the method in [44], the aim of outlier detection and the re-assignment of these outliers is to handle concept drift problem more efficiently.

In the method, a cluster identifier, which is unique and increasing with the number of clusters, is saved for each cluster. When a new cluster is generated or an existing cluster is updated, a new cluster identifier is obtained and assigned to this cluster. Therefore, the most recent clusters have the highest identifiers. The outdated clusters are determined and removed by calculating the mean and standard deviation values for cluster identifiers and cluster sizes.

Given that μ_{id} is the mean of identifiers, σ_{id} is the standard deviation of identifiers, μ_{size} is the mean of cluster sizes, the σ_{size} is the standard deviation of cluster sizes; a cluster is deleted from the model if the following two conditions are satisfied;

- If the identifier of the cluster is less than μ_{id}-σ_{id}
- If the size of the cluster is less than μ_{size}-σ_{size}

Another recent Dirichlet Process based method in this group is proposed in [48]. The proposed method, Online Semantic Enhanced Graphical Model (OSGM), uses semantic information by considering word co-occurence, and detects evolving active topics. It handles term ambiguity problem without using any external resources such as word embeddings. The generated clusters are represented by a 7-tuple CF vector which includes word frequencies, total number of documents in clusters, word-to-word co-occurence matrix, total number of words, decay weight, last update timestamp and arriving timestamps of words. Decay weight value is used for deciding whether the old clusters are going to be removed from the model or merged with existing clusters. OSGM updates active clusters by using this decay value and also

removes words from clusters by calculating a recency value. For each incoming document in the stream, OSGM calculates the probabilities for assigning the document to one of the existing clusters or generating a new cluster. On the basis of probability values, the document assignment to cluster is fulfilled and the CF vector for the related cluster is updated. OSGM handles the term ambiguity problem by utilizing the co-occurence of terms in semantic term space. It calculates the probability of assigning an incoming document to an existing cluster by using co-occurences. Online Semantic Enhanced Graphical Model-Exclude Semantic Smoothing (OSGM-ES) is proposed as a variation of OSGM in [48] where inverse clustering frequency is not included while calculating the probability of an incoming document to an existing cluster. The last method in this group is called Topic Enhanced Dirichlet Model (TEDM) [52] and it also utilizes graph structures as distinct from the other methods in this group. TEDM uses co-occurences of the words while constructing the graph and produces topic phrases from this graph structure. It uses this phrases to calculate weight values for the words in documents. These values are used in the calculation of probabilities to assign an existing cluster or to generate a new cluster. TEDM also handles a decay weight value for each cluster and calculates the probabilities to merge the cluster with the other clusters. Then TEDM takes the maximum value for merge and compares it with the probability of standing alone. The cluster is merged with another one or continues to be stand alone with respect to these values. TEDM has also a reassignment procedure and calculates importance scores for short text documents in cluster.The short text documents having low importance scores are re-assigned to the clusters or new clusters are generated.

The Dirichlet process methods in second group which are **Dirichlet Process based Methods with Single Word Probability** handling each word individually. They calculate the probabilities for assigning a short text to an existing cluster or generating a new cluster by using the frequency values for each word. They do not consider the co-occurences of words such as biterms. The first method in this group is called Non-Parametric Dirichlet Model With Language Translation Component (NDML). NDML has a non-parametric Dirichlet Model which clusters unknown number of topics in a short text stream. The methods mentioned so far about short text stream

clustering work only with datasets in English. Following a different approach, NDML is a multi-lingual text stream clustering method. This model is composed of a non-parametric Dirichlet Process based component and a language translation component. As the first step of this method, data stream is divided into chunks. Here, this division can be in terms of time windows such as hours, days or weeks or the data can be divided in terms of fixed size windows according to the number of instances. For each chunk, the instances are assigned to the existing clusters or a new cluster is generated with respect to the probability values calculated by using a Dirichlet model. CF vectors representing the clusters include the number of documents, frequency values for each unique term and the total number of words. Clusters can be obtained for any language in such a way that the language of the text is determined by using an external tool and it is translated into English. Then, the hypernyms for key terms are extracted with another external tool. The aim of using hypernyms is to expand the cluster features to improve clustering performance and these hypernyms are used to calculate the similarity between two clusters in different languages.

As another method in this group, Lifelong Learning Augmented Short Text Stream Clustering (LAST) [35] utilizes an episodic memory and an experience replay module. Here, experience replay indicates the update process of CF vectors. Episodic memory is the space where short texts are randomly saved and the experience replay is processed by sampling from this episodic memory. The use of such an episodic memory increases the influence of recent clusters. Replay interval and store interval values are the parameters for the model. They indicate periodic intervals and they are represented as the count of timestamps that should be passed for an operation. In the LAST method, when the replay interval is completed, samples from the episodic memory are retrieved to process the experience replay module. After that the probability values to assign the incoming instance to an existing cluster or to generate a new cluster are calculated. Finally if the store interval period is completed, the incoming short text is written to the episodic memory and old texts are removed. The detection of old text is not clearly described and this process remains ambiguous in [35].

MStream [36] is widely used as a baseline method for most of the algorithms. It is a Dirichlet Process based model. There is another version of the method, called MStreamF, which includes forgetting property to handle topic evolution. MStreamF

removes outdated clusters from the model while processing the data stream. In both MStream abd MStreamF, the clusters are represented with CF vectors, which include frequencies of the words in cluster, the number of documents in cluster and the total number of words in cluster.

MStream and MStreamF handle the input short text streams as batches. In the current batch, for each document, the algorithm calculates probability for the assignment to the existing K clusters and an extra probability for generating a new cluster by using a Dirichlet Process based approach. Hence, totally K+1 probabilities are obtained, and the maximum one is selected to make a decision. This phase is called *one pass clustering process*. After all documents are clustered, it is iterated several more times, such that the clusters are refined by redistributing the documents. MStreamF assumes that there is a fixed number of batches to be processed. When a new document is arrived in the stream, if the number of batches are greater than the maximum number of batches to be saved, the documents in the oldest batch are removed from the model and the CF vectors are updated. These removed batches and their clustering results are generally saved on a secondary storage environment to use in a further offline analysis.

Another Dirichlet Process based method with single word probability is proposed in [46]. This method uses Gibbs sampling and assumes that all of the words in a short document belongs to a single topic. The proposed method called Dynamic Clustering Topic Model (DCT) utilizes the topic distribution of previous documents at time t as a prior information and uses the new streaming documents to change the topic distribution. There are two versions of the method, Short Term Dependency for DCT and Long Term Dependency for DCT. Long term dependency model can relate the distributions up to L time steps before, while short term dependency model can relate the distributions only in the current and the previous timestamps. DCT initially distributes the incoming short documents at time t randomly to the topics. Then for the number of iterations given as parameter, it makes sampling with respect to the count of incoming documents at current timestamp and removes the sample from the assigned cluster. For each iteration assignments are repeated with new calculated probabilities and distributions.

One of the most recent methods about short text stream clustering is Dynamic Clustering For Short Text Stream Based On Dirichlet Process (DCSS) [53]. It has a similar approach to the other Dirichlet based methods such that it calculates the probability values for assigning an incoming short text to one of the existing clusters or generating a new cluster. As in most of the other methods in the same group, DCSS has an iteration phase where for a predefined number of iterations, the short texts are redistributed to clusters. The cluster feature values are updated with respect to these re-assignments. DCSS has a similar approach to MStream in many aspects but as Zhang et al. indicate, MStream does not consider the correlation of the topic distribution at neighbouring time points whereas DCSS does.

We can summarize the Dirichlet Process Based Methods as follows. MStream/MStreamF include the basic process of the methods in this group and they are generally used as a baseline for the evaluation of the Dirichlet Process based methods. As it can be seen in the Table 2.1, the methods generally process the data stream instance by instance except two of them, the method by Rakib et al. and NDML. Also DCT and DCSS do not use CF vectors whereas all of the remaining methods do. DP-BMM/DP-BMM-FP, OSDM and Rakib et al.'s method use co-occurrences of the words and therefore they are grouped *Dirichlet Process Based Methods with Co-Occurrence View* while the others are placed in *Dirichlet Process Based Methods with Single Word Probability*.

2.2.2.2 Similarity Based and Incremental Methods

In this section, similarity based methods having an incremental approach for short text stream clustering are reviewed in detail. One of the methods in this group is Incremental Clustering With Vector Expansion (ICVE) [49]. This method is considered as an enhancement for traditional incremental clustering methods. The similarity values are calculated with respect to co-occurences and this is used for vector expansion. Additionally, burst detection is performed after clustering and new events are detected. In ICVE, the data stream is processed in buckets. In each bucket, the most discriminating terms are determined for document representation. The essence of the method lies in applying vector expansion to document vectors in order to capture the

semantic similarity between them. Vector expansion is obtained by checking the second order (SO) co-occurence of terms which is obtained by calculating the cosine similarity between co-occurence vectors of two terms. These similarity values are used for the expansion of tweet vectors and the centroid of active clusters. Then, a traditional incremental clustering process is applied to this expanded vectors. As the final step, burst detection is processed and bursty terms are extracted from the buckets. On the basis of the given thresholds, new events are detected and the cluster with a bursty term is marked as an event cluster.

Another similarity based incremental method, ComStreamClust, is proposed in [50]. This method performs outlier detection and re-assgins these outliers to clusters. As the first step, previously specified amount of tweets are randomly assigned to previously specified number of clusters. Then the tweets are redistributed to clusters on the basis of their sentence embeddings. To fulfill this, the cosine similarity between the tweet's embedding and each cluster centroid is calculated, and the tweet is assigned to the cluster having the maximum similarity. If this similarity value is below a given threshold, then a new cluster is constructed. The tweets are processed by using a sliding window mechanism. At a specified period, outlier detection is performed and outliers are re-distributed to the clusters. The method also uses a fading rate and the clusters having faded-weights lower than a threshold are removed from the model.

In FastStream [51], Rakib et al. propose a method to reduce the cost of finding the clusters relevant to the incoming text. When a new text comes in the stream, a novel similarity between the incoming text and the cluster centroids is calculated. In this method, the short texts have features as unigram, bigram and biterm with respect to the words included in short text. The method uses an index structure called *Inverted Index* which is a hash map like data structure including mappings from document features of unigram, bigram or biterm to documents [54]. With the help of this index structure only the clusters containing the features of the incoming text are retrieved instead of all clusters. By this way, this approach reduces the cost of finding clusters similar to the incoming instance. If the cluster with the maximum similarity has a similarity above the threshold value, the text is assigned to this cluster. Otherwise, a new cluster is constructed. To remove the outdated clusters, the mean and standard deviation values are calculated for the sizes and identifiers of the clusters as in

the same way in [44]. Here, *Delete-interval* value, which is a parameter indicating when to remove the outdated clusters. This parameter is defined in terms of number of incoming short texts. Another incremental method in this group is proposed by Soleymanian et al. [55] and this method makes concept modeling by generating embeddings for documents with PSDVec [56]. In this method, micro concepts that are sets of semantically similar embedded words are generated and for clustering K-means and K-means++ are used. An offline clustering phase is also proposed which is executed upon a user request and it utilizes micro concepts generated in online phase of the method.

As it can be inferred from the Table 2.1, ComStreamClust and FastStream process the incoming data stream instance by instance while ICVE uses buckets. FastStream utilizes CF vectors to represent the clusters whereas the others do not use CF vectors. Finally, ICVE and FastStream uses co-occurences of the words to cluster the data stream and ComStream do not handle co-occurences of the words.

2.2.2.3 Word Relation Network Based Methods

In this section, word relation network based methods for short text stream clustering are reviewed. Two of these methods are EWNStream [37] and EWNStream+ [32], which is the improved version of EWNStream. These methods handle the data stream in batches and construct a word relation network by using the frequency values as node values and the co-occurence values as edge values. They represent the clusters with Cluster Abstract (CA) data structure similar to CF vectors. In CA, the cluster index, the number of short texts, sum of timestamps, squared sum of timestamps and core term sets are included for each cluster. To determine the core term sets, Fast Core Term Group Search (FCTGS) algorithm is applied. In FCTGS, the words in word relation network are ordered in descending order of node values. The closeness value of a node with its neighbours is calculated starting from the first node. If the closeness value is above a given threshold, this node is added to core term group. For similarity calculation, the intersection of the incoming text and the core term groups of a cluster are considered. The text is assigned to the cluster having the maximum

intersection. After the assignment, the CA value for the cluster is updated.

The EWNStream+ differs with the previous version EWNStream in two points:

- There is a new closeness metric for EWNStream+.
- The EWNStream+ applies a decay policy to handle the fading of topics in real world. When a new batch is started to be processed, the outdated clusters are determined according to their timestamp values. The word relation network is updated and CA of these clusters are deleted after the removal of these outdated clusters.

Edge Significance based Louvain Algorithm (ESBLA) can also be considered as a graph based method in this group [57]. ESBLA operates a community detection process which has four main steps. Firstly, it calculates Conditional Term Frequency-Average Inverse Window Frequency (CTF-AIWF) values and the calculation is similar to TF-IDF (Term Frequency-Inverse Document Frequency). ESBLA obtains Event Candidate Score (ECS) by using these CTF and AIWF values. Then candidate keywords are selected with respect to ECS values and a graph is constructed where nodes represent these keywords. The weight values for the graph are calculated with structural similarity ve semantic similarity values which are obtained according to the co-occurences of keywords in tweets. After that an algorithm similar to Louvain [58] is run where keywords are put into their own community initially and community detection is operated with modularity gain values. The process goes on while the modularity gain is positive. Finally, location detection is done and the events are visualized on Google Maps.

Our proposed method T-GSC is also a graph based approach, however there are significant differences. T-GSC has a two-phase clustering process and also it uses a hierarchical method. Therefore, it does not need any threshold values as parameters for the model. Also, T-GSC uses both sentence embeddings and node embeddings and combines the power of these two concepts in clustering.

The results of experiments conducted to evaluate the proposed methods are given in

Table 2.2 and Table 2.3. Here, GN refers to *Google News* and Twe is the short form of *Tweets*. Due to the number of presented experiment results, we present them in two tables.

Table 2.2: Results of Methods on Datasets

Met.	GN	GN-T	Twe. (TREC)	Twe.-T (TREC)	Twe. (Turk. Twe.)	Twe. (Spec.)	Twe. (Spec. NDCG)	Reut.
DP-BMM	0.881	0.882	0.864	0.865	-	-	-	-
DP-BMM-FP	0.865	0.837	0.845	0.850	-	-	-	-
OSDM	0.815	0.858	-	-	-	-	-	0.552
Rakib et al.	0.862	-	-	0.892	-	-	-	-
NDML	-	-	-	-	-	-	-	-
LAST	0.840	-	0.840	-	-	-	-	-
MStr.	0.834	0.850	0.844	0.882	-	-	-	-
MStr.F	0.797	0.873	0.823	0.923	-	-	-	-
DCT	-	-	-	-	-	-	0.642	-
DCSS	0.639	0.907	0.811	0.936	-	-	-	-
OSGM	0.815	0.822	0.854	0.854	-	-	-	-
OSGM-ES	0.822	0.836	0.853	0.852	-	-	-	-
ICVE	-	-	-	-	0.800<Best Precision <1.000	-	-	-
ComStreamClust	-	-	-	-	-	-	-	-
FastStream	-	0.844	-	0.848	-	-	-	-
EWNStream+	-	-	-	-	-	0.800 <BestNMI <0.900	-	-

Table 2.3: Results of Methods on Datasets (cont.)

Method	Reuters-T	SO-T	NT	NTS	CrisisLex (Recall)	FA-CUP (Topic Recall)	Covid-19 (Topic Recall)
DP-BMM	-	-	-	-	-	-	-
DP-BMM-FP	-	-	-	-	-	-	-
OSDM	0.554	-	-	-	-	-	
Rakib et al.	-	-	0.830	0.756	-	-	-
NDML	-	-	-	-	0.844	-	-
LAST	-	-	-	-	-	-	-
MStr.	-	-	-	-	-	-	-
MStr.F	-	-	-	-	-	-	-
DCT	-	-	-	-	-	-	-
DCSS	-	-	-	-	-	-	-
OSGM	-	-	-	-	-	-	-
OSGM-ES	-	-	-	-	-	-	-
ICVE	-	-	-	-	-	-	-
ComStreamClust	-	-	-	-	-	1.000	0.917
FastStream	-	0.777	-	0.774	-	-	-
EWNStream+	-	-	-	-	-	-	-

In addition to these results, a summary is given in Table 2.4. For some of the methods, such as ICVE and EWNStream+, the results are illustrated as graphs instead of listing in exact numbers. For such methods, the results are expressed as intervals in the table.

The basic observations on these reported results can be summarized as follows:

Table 2.4: Overall Best Results of Methods on Datasets

Method	Best Result	Datasource	Notes
DP-BMM	0.882	Google News-T	NMI value
DP-BMM-FP	0.865	Google News	NMI value
OSDM	0.858	Google News-T	NMI value
Rakib et al.	0.892	Tweets-T	NMI value, TREC microblog
NDML	0.844	CrisisLex	Recall value
LAST	0.840	Google News	NMI value
MStream	0.882	Tweets-T	NMI value
MStr.F	0.923	Tweets-T	NMI value
DCT	0.642	Tweets	Normalized Discounted Cumulative Gain (NDCG) value [59], a specific constructed set of tweets
DCSS	0.936	Tweets-T	NMI value
OSGM	0.854	Tweets-T	NMI value
OSGM-ES	0.853	Google News-T	NMI value
ICVE	0.800 <Best Precision <1.000	Turkish Tweets dataset, exact results are not given	Precision value
ComStreamClust	1.000	FA-Cup	Topic Recall value
FastStream	0.848	Tweets-T	NMI Value TREC microblog
EWNStream+	0.800 <Best NMI <0.900 (around 0.850)	Tweets	Specific tweet collection, exact results are not given

- NMI is used as a performance metric for the majority of these methods.
- Some of the datasets such as Google News or Tweets are commonlu used for the performance evaluation of the methods.
- According to the results, DP-BMM/DP-BMM-FP outperform the other methods on real world Google News and Tweets datasets, and DP-BMM has the best NMI value for these two datasets.
- Generally, the performance of the methods for the synthetic datasets obtained from Google News and Twitter are better than the original real world datasets. This may indicate that the adjustments and editions in original datasets affect and improve the performance of methods.
- DCSS performs best in Google News-T and Tweets-T datasets whereas DP-BMM and MStreamF follow it on the these datasets.
- DCSS provides better results than the other methods on synthetic datasets. For instance, its NMI value is 2.8% higher than its closest successor,DP-BMM. However, its performance is poor with respect to other methods in original datasets.
- Some of these methods such as DP-BMM/DP-BMM-FP provide better clustering results for Google News than Twitter whereas some of them such as MStream/MStreamF have lower performance for Google News than Twitter. This can indicate that some of these methods are more appropriate for formal

contents as in Google News while others are more appropriate for informal short texts as in Twitter.

- There are various synthetic datasets such as StackOverflow-T and NT, but they are used in only one of the reviewed methods. Another dataset, NTS, is used in two of the reviewed methods and hence they are not commonly used in performance evaluation.
- The datasets used for evaluation are generally in English but NDML is the only one which can work on different languages. ICVE is also performed on a dataset consisting only Turkish tweets.
- It is also observed that ComStreamClust has a perfect topic recall value on FA-Cup dataset.

CHAPTER 3

ENHANCEMENTS FOR DATA STREAM CLASSIFICATION

3.1 m-kNN (Mean Extended kNN) and CSWB (Combined Sliding Window Based) Classifier

The proposed methods in this section are published in [4] and [60]. The details about these methods and the content of this section is adapted from these studies.

3.1.1 m-kNN (Mean Extended kNN)

In m-kNN, we apply kNN within sliding windows with the difference from the traditional kNN that *k-1* instances are selected within the window, whereas the last instance is used as an average of the history. At the beginning of the method we fill the current window with the most recent past instances. After that, within the current window, by using Euclidean distance, k-1 nearest neighbors of the incoming instances are found. Additionally, we also calculate centroids of the classes by using the past instances. Hence, we obtain class representatives from the history. Among the class representatives, we determine the most similar one, and this instance is used as the k^{th} nearest neighbor. As in the conventional kNN, the class label is determined with majority voting among these *k* instances.

Assuming that we learn the actual class of the instance in the next time instance, the representative of the class is updated. In order to slide the window, this instance is pushed into the head of the window, and the oldest instance is removed.

The algorithm for m-kNN can be summarized as given in Algorithm 1.

Algorithm 1 m-kNN

procedure M-KNN(Data stream, parameter for kNN: k, the size of the sliding window: n)
    f: number of features
    c: number of class labels
    $claLab[c][f]$: Features list for each class label
    w:=$i_0, i_1, ..., i_{n-1}$ ▷ Put the first $n-1$ instances into sliding window
    for $i = 0;\ i < c;\ i++$ **do**
        for $j = 0;\ j < f;\ j++$ **do** claLab[i][j]:=mean value of feature ▷ Calculate the mean values of features for each class
        end for
    end for
    ins: incoming instance in data stream
    $eucDis[n-1]$: Euclidean Distances
    for $i = 0;\ i < n-1;\ i++$ **do** $eucDis[i]$:=EuclDist(ins,$w[i]$) ▷ Calculate Euclidean Distance between each element in the sliding window and incoming sample
    end for
    $neaNei[]$:=k-1 nearest neighbors to ins
    $eucDis[c]$: Euclidean Distance ▷ Find the $k-1$ nearest neighbors of incoming sample in sliding window
    for $i = 0;\ i < c;\ i++$ **do** $eucDis[i]$:=EuclDist(ins,$claLab[i]$) ▷ Calculate Euclidean Distance between ins and each class with the previously calculated average values of attributes
    end for
    $neaNei[k]$:=Nearest neighbor to ins in $claLab[]$ ▷ Add the nearest class to the $k-1$ neighbors
    $assCla$:=Class label having the majority in k-nearest neighbors ▷ Assume that we learn the actual label of the instance after classification

20: **for** $i = 0;\ i < c;\ i++$ **do**

21: **for** $j = 0;\ j < f;\ j++$ **do** Update the mean values of features for this actual class of the instance

22: **end for**

23: **end for**

24: Add ins to the sliding window

25: Remove $w[0]$ from sliding window ▷ Update the sliding window

26: **return** $assCla$

27: **end procedure**

In order to further improve the method for a dynamic and adaptive nature, a dynamic size for sliding window is elaborated on. Additionally, using a dynamic number of nearest neighbors obtained class representatives is included as well. Finally, misclassified instances in the current window are replaced with the these instances obtained from the class representatives.

- **Using a dynamic size for sliding window:** For this approach we have an error count that indicates the count of misclassified instances in the current window. If this count exceeds a pre-defined threshold value, we discard the portion of the window including the first misclassified instance. After cutting this portion of window, for the next iterations where classifications are correct, we continue to extend the sliding window.

- **Using a dynamic count for the nearest neighbors obtained from centroids of classes:** For this approach when the error count exceeds given threshold, we increment the number of nearest neighbors obtained from the centroids of the classes among the K-nearest-neighbors up to a pre-defined maximum value. As a result, we can increase the weight of the average values for the classes. When the error count decreases below the given threshold, we decrement this count for nearest neighbors.

- **Replacing misclassified instances with nearest centroid of classes:** For this approach, if the error count exceeds the given threshold, we replace the first occurrences of misclassified instances with a pre-defined count in the current

window.

3.1.2 CSWB (Combined Sliding Window Based) Classifier

As the second enhancement for sliding window based classifiers, CSWB is proposed. CSWB combines m-kNN, Naive Bayes and kNN classifiers. For voting, we investigate two alternative approaches:

- **Majority voting with equal weights:** In this approach, after each classifier completes its process, if at least two of them produce the same result, the instance is assigned to this class. Otherwise, since Naive Bayes has high accuracy results in general according to our experiments, it has a higher priority to determine the class label.
- **Voting with current accuracy:** In this version, we keep the accuracy values up to the new classification step and sum up the accuracy values for classifiers having the same result. Finally, we assign the instance to the class having the highest total accuracy value.

3.1.3 Experiments

3.1.3.1 Experiment-1: Analysis on m-kNN

In this experiment we have used four real-world datasets. The details of data sets used in this experiment can be given as follows:

- **KDD Cup 99':** The data set, which is about network intrusion [1], includes 42 attributes and contains about 10M instances. In our experiments, we used a 10% portion containing about 494K instances.
- **Electricity Market:** The data set contains instances collected from the Australian New South Wales Electricity Market [2]. In this electricity market, there

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99
[2] https://www.openml.org/d/151

are not fixed prices and they are affected by demand and supply. The prices are set every five minutes. The data set contains 45312 instances and the class label identifies the change of the price relative to a moving average of the last 24 hours.

- **Forest Cover Type:** The data set contains observations about forest cover types of 30 x 30 meter cells in US [3]. The instances in this data set have 54 attributes such as elevation, aspect and slope.

- **Air Quality:** It is about the amount of several chemicals in the air [4]. We have used the values for the amounts of chemicals such as tin oxide, titania, tungsten oxide and indium oxide. We have clustered the values of indium oxide and determine the labels for classification.

We implemented m-kNN and MC-NN and also we used Apache SAMOA to execute VHT method. For m-kNN our parameters are *k=10, window_size=100* and for MC-NN *error_threshold=5, performance_threshold=0.75*.

The results of our tests for our m-kNN method, VHT and MC-NN on KDD Cup 99, Electricity Market, Forest Cover Type, and Air Quality datasets are given in Figure 3.1, 3.2, 3.3, and 3.4, respectively. In these figures, the highest accuracy values for each method among the results obtained with different parameters are taken into account.

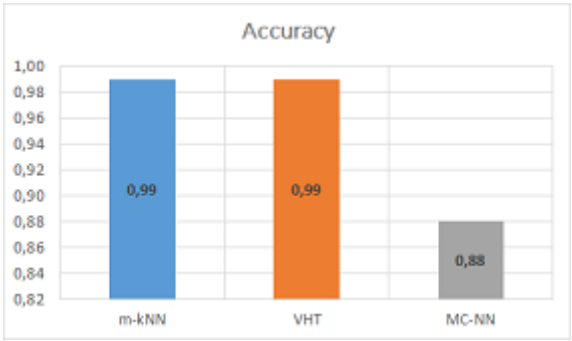

Figure 3.1: Accuracy for KDD Cup 99' in Experiment 1

[3] https://archive.ics.uci.edu/ml/datasets/Covertype
[4] https://archive.ics.uci.edu/ml/datasets/Air+quality

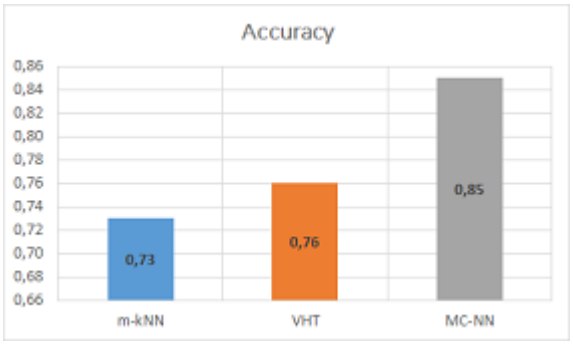

Figure 3.2: Accuracy for Electricity Market in Experiment 1

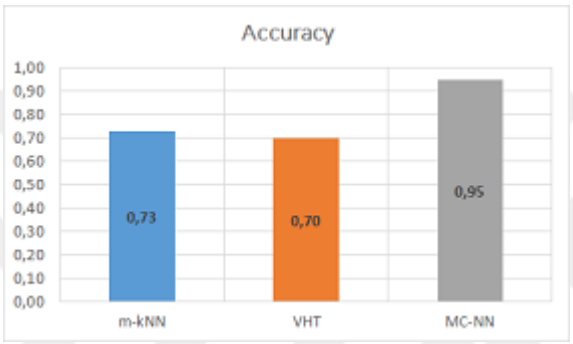

Figure 3.3: Accuracy for Forest Cover Type in Experiment 1

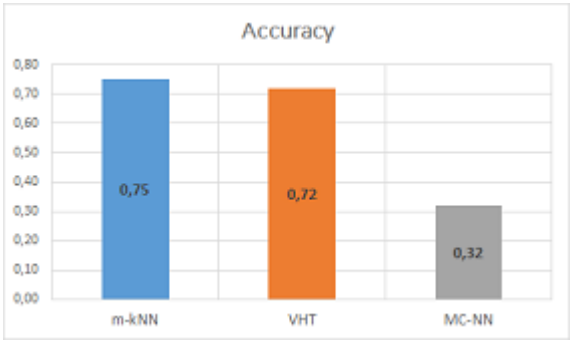

Figure 3.4: Accuracy for Air Quality in Experiment 1

According to the results of our experiments, we can see that our m-kNN method has a high accuracy for KDD Cup 99' data set as 99%. It also has the best accuracy values for Air Quality data set. On the other hand, it has lower accuracy values for Electricity Market and Forest Cover Type data sets with respect to MC-NN.

3.1.3.2 Experiment-2: Analysis on CSWB

In this experiment, we used several other data sets from real-world and also two synthetic data sets.

- **Appliances Energy Prediction:** The samples in this data set were collected periodically with an interval of 10 min for about 4.5 months [5]. We used the columns temperature in living room, outside temperature, outside pressure, wind speed and the energy consumption in the data set. We clustered the values of energy consumption and determine the labels for classification .

- **Human Activity Recognition (HAR):** This data set is obtained from the recordings of 30 subjects performing activities of daily living while carrying a waist-mounted smartphone with embedded inertial sensors[6]. There are 561 attributes and 10299 instances in this data set .

- **ATM Terminal Data sets:** This data set contains amount of money withdrawn from ATM machines. Each data instance contains an identity for the ATM machine, a date and an amount. We used the data of two ATM terminals for one year period and duplicated this data to a period of 20 years. We also extracted several features including *month, day of month, day of week* and *is work day* from the date information. To label the instances we discretized the *amount of money* feature by applying k-Means clustering. This is the first version of our ATM data set (ATM v1). As the second version we also added several other features on weather conditions as *temperature, humidity and wind speed* by using the location of ATM terminal (ATM v2).

- **SEA:** This is a synthetic data set that contains 60,000 examples, 3 attributes and 2 classes [7]. In this dataset, the attributes are numeric between 0 and 10 [61].

[5] https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

[6] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

[7] http://www.liaad.up.pt/kdus/downloads/sea-concepts-dataset

- **Hyperplane:** This is another synthetic data set, which contains 10,000 instances with 10 attributes and 2 classes [8].

For the first part of the analysis we conducted experiments by adding enhancements for making our m-kNN method more dynamic and adaptive. In the analysis, each of the following enhancements is applied separately:

- Using a dynamic size for sliding window
- Using a dynamic count for the nearest neighbors obtained from centroids of classes
- Replacing misclassified instances with nearest centroid of classes

Table 3.1: Results after enhancements for m-kNN are enabled (Dynamic count enabled refers to dynamic count of nearest mean enabled)

Dataset	Original m-kNN	Dynamic Window Enabled	Dynamic count enabled	Replace with nearest mean enabled
ATM v1	**0.29**	**0.29**	0.14	**0.29**
KDD CUP 99	**0.99**	**0.99**	**0.99**	**0.99**
Air Quality	**0.75**	**0.75**	0.70	**0.75**
Appliances Energy	**0.61**	**0.61**	0.57	**0.61**
Electricity	**0.73**	**0.73**	**0.73**	**0.73**
Hyperplane	**0.74**	**0.74**	0.60	**0.74**
SEA	**0.82**	0.74	0.77	0.74
HAR	**0.81**	**0.81**	0.78	**0.81**
Forest Cover Type	0.73	0.73	0.72	**0.80**

The results of this experiment are given in Table 3.1. According to the results we can see that the method with dynamic number of nearest neighbors lowers the accuracy values and the other enhancements do not change the results. The dynamic count enabled version where the number of nearest neighbors from the centroids can change may be elaborated as a future work to see if it can produce better results. The dynamic count may be the half of k or it may be equal to k.

For the second part of this experiment, we applied m-kNN (without dynamic and adaptive extensions), Naive Bayes, traditional kNN, and CSWB classifier in two versions of voting. The results of our tests on the data set for our m-kNN method, VHT

[8] https://www.win.tue.nl/~mpechen/data/DriftSets/hyperplane1.arff

Table 3.2: Summary of the results in Experiment-2

Dataset	k	window_size	mkNN	kNN	Naive Bayes	CSWB v1	CSWB v2
Air Quality	5	100	0.77	0.77	0.76	**0.78**	**0.78**
Appliances Energy	5	25	**0.64**	0.63	0.63	**0.64**	**0.64**
Appliances Energy	5	500	0.63	0.63	0.51	0.63	**0.64**
HAR	5	250	0.91	0.91	0.87	**0.92**	**0.92**
HAR	5	500	0.92	0.92	0.82	**0.93**	**0.93**
ATM v2	5	25	0.30	0.35	0.31	**0.36**	**0.36**
ATM v2	5	50	0.32	0.38	0.31	**0.39**	**0.39**
ATM v2	5	250	0.30	0.42	0.32	**0.43**	**0.43**

and MC-NN are given in Figure 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, and 3.14.

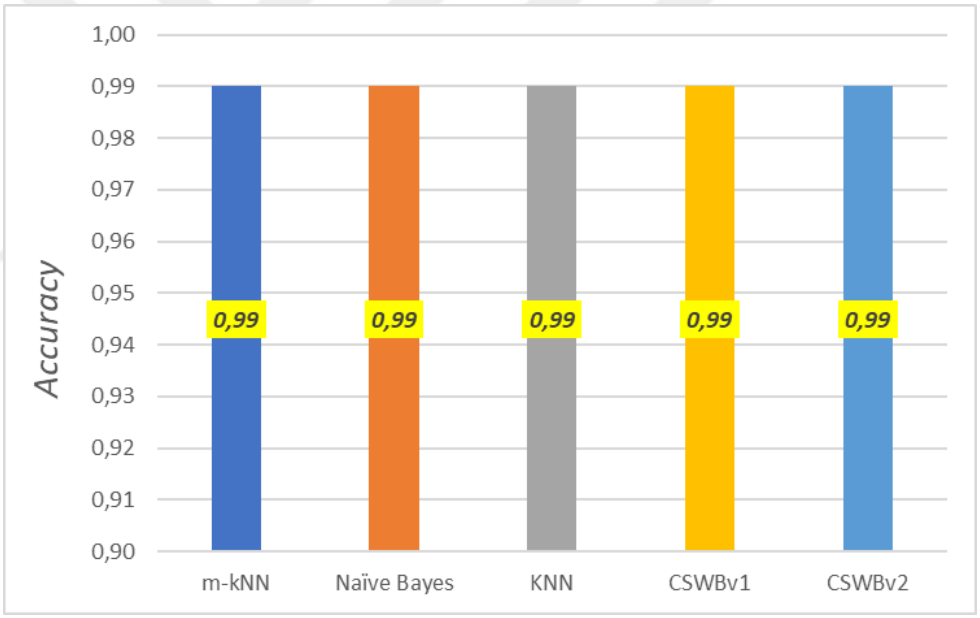

Figure 3.5: Accuracy for KDD CUP 99 in Experiment 2

When the results are compared under equal valued parameters, it can be seen that m-kNN and CSWB methods have better results that the other methods for several data sets. These results are given in the Table 3.2. In this experiment we have used k values: {5, 10} and window size values:{25, 50, 100, 250, 500}.

For each method the highest accuracy is taken into account for these varying param-

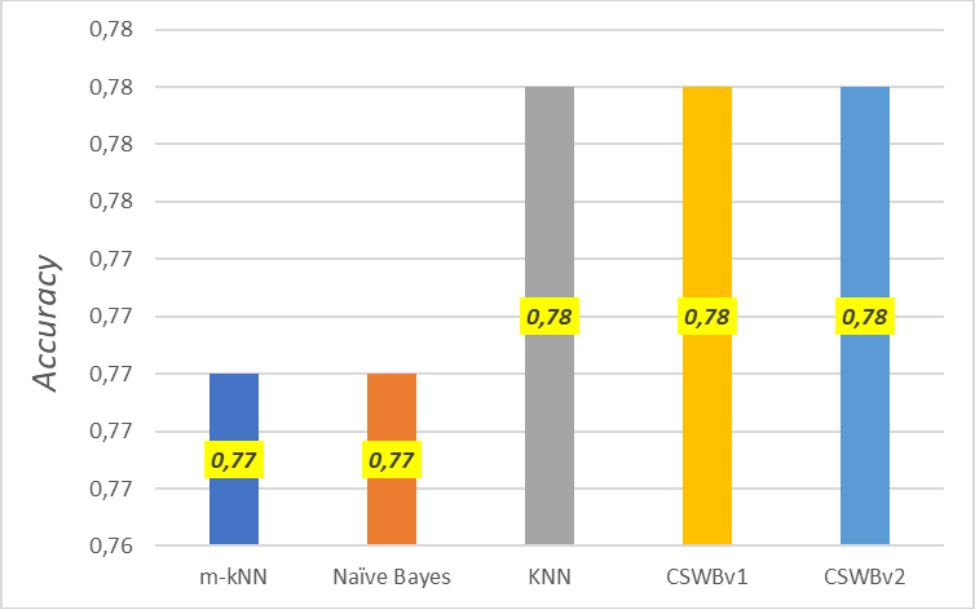

Figure 3.6: Accuracy for Air Quality in Experiment 2

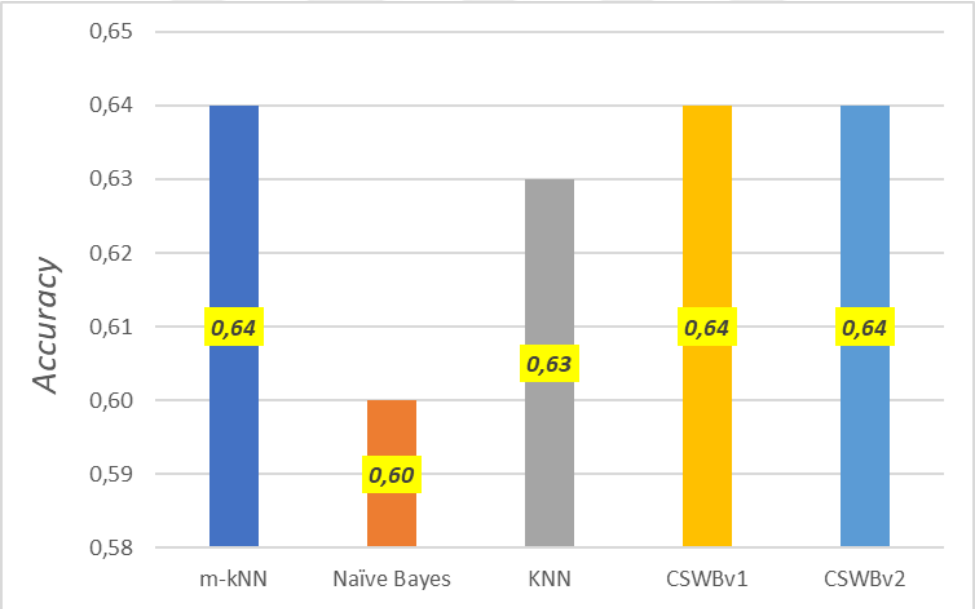

Figure 3.7: Accuracy for Appliances Energy Pred. in Ex.2

eters. According to the results among these 10 data sets we can see that m-kNN has the highest accuracy for 3 of the data sets and CSWB classifier is the best for 6 of the data sets. When we analyse the results in Table 3.2, we can see that our enhancements give better results when k=5 with respect to k=10. This may indicate that making k respectively smaller can reveal the effect of additional nearest instance

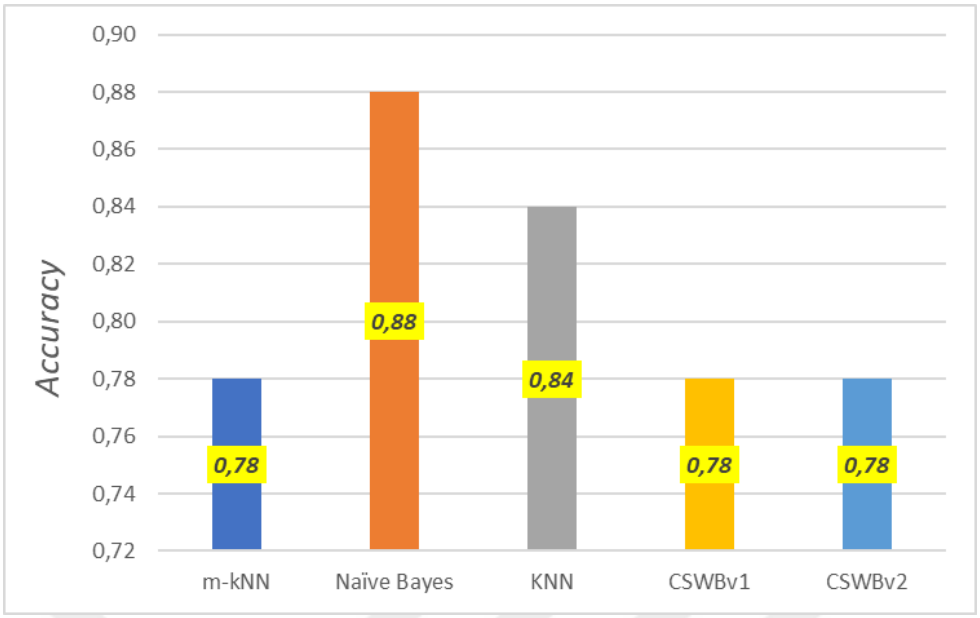

Figure 3.8: Accuracy for Electricity Market in Experiment 2

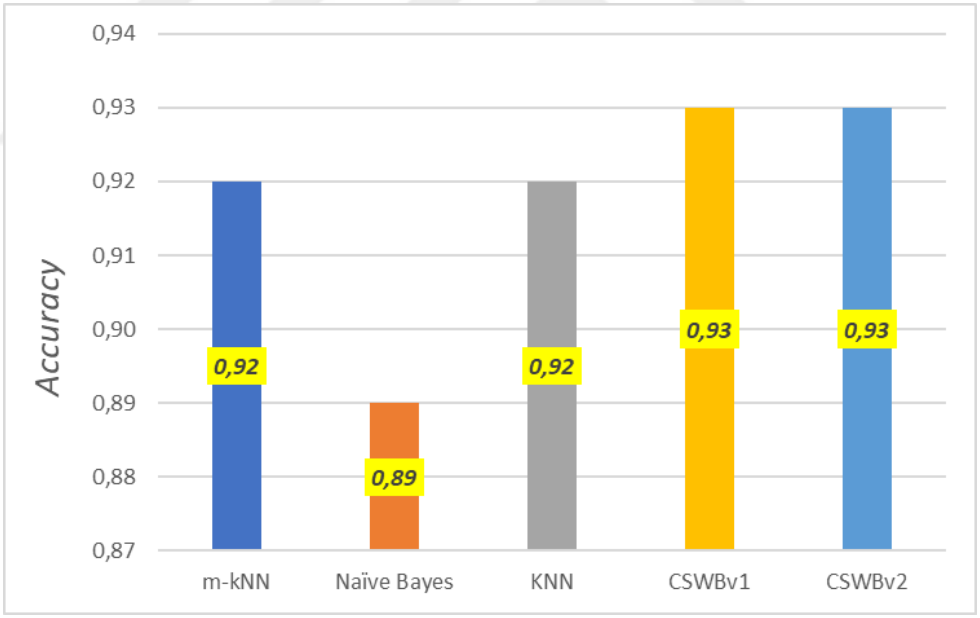

Figure 3.9: Accuracy for Human Activity Recognition in Ex.2

coming from the mean values of attributes and increasing k may degrade the success of our enhancements.

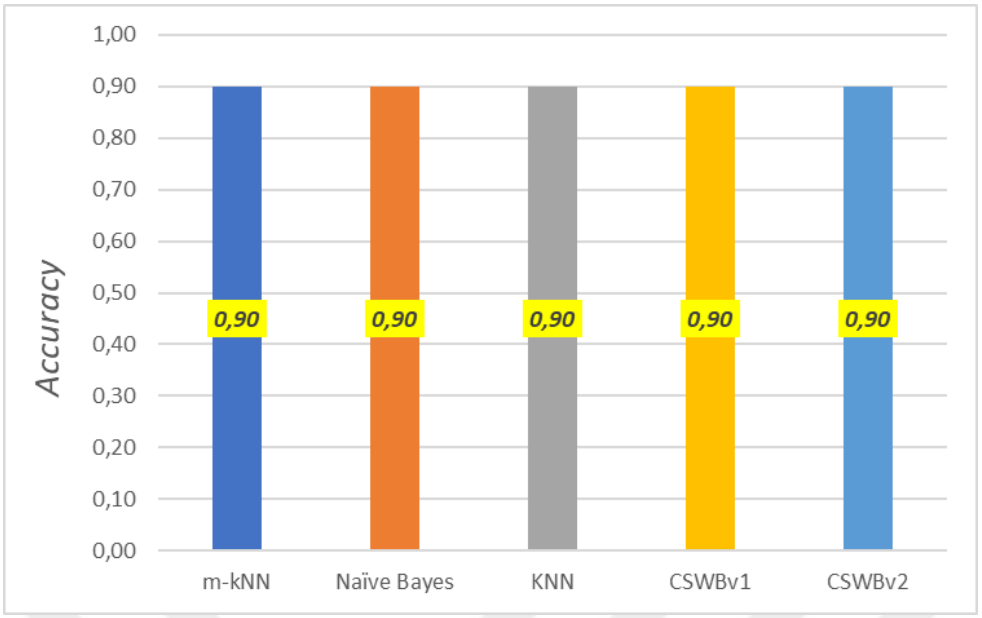

Figure 3.10: Accuracy for Forest Cover Type in Ex.2

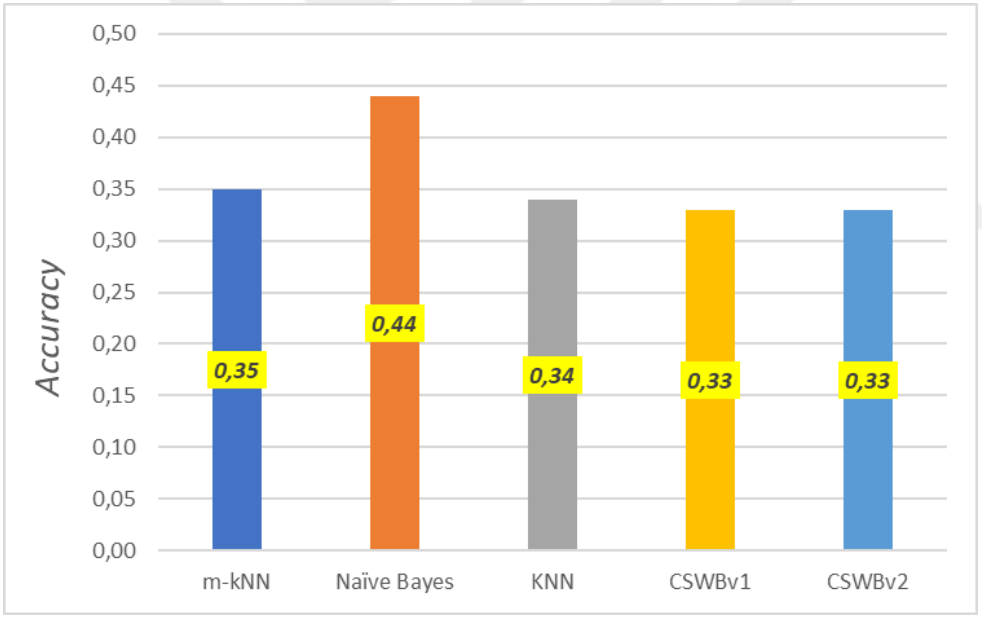

Figure 3.11: Accuracy for ATM v1 in Experiment 2

3.2 Extended Hybrid Stream Classifiers: CSWB-e and CSWB-e2

In this section, we propose two new versions of CSWB where we use new classifiers together with m-kNN. In CSWB-e, m-kNN is combined with K* (K-Star) and C4.5. In CSWB-e2, in addition to m-kNN, K* and Naive Bayes classifier are used. The mo-

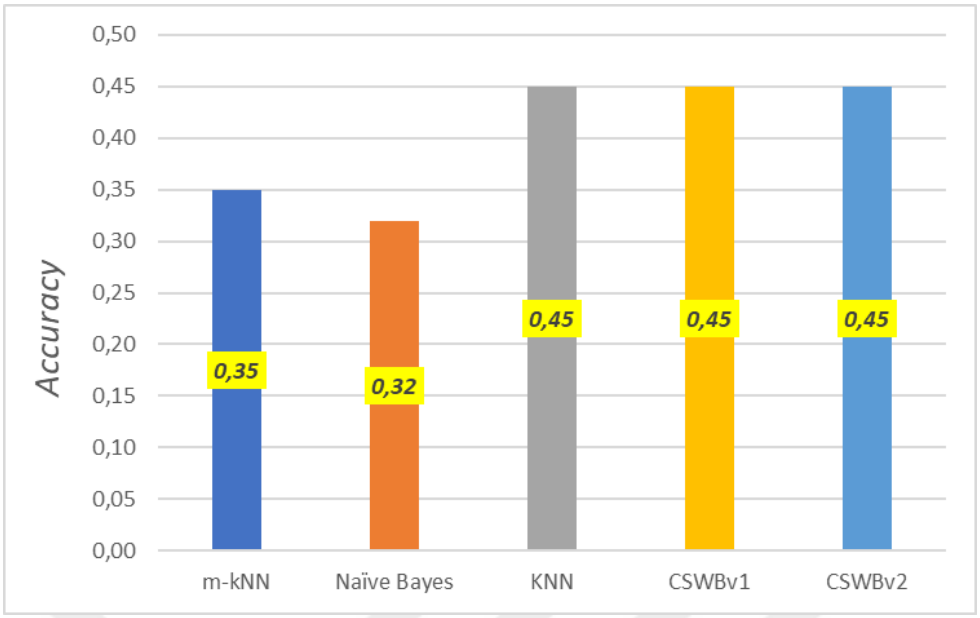

Figure 3.12: Accuracy for ATM v2 in Experiment 2

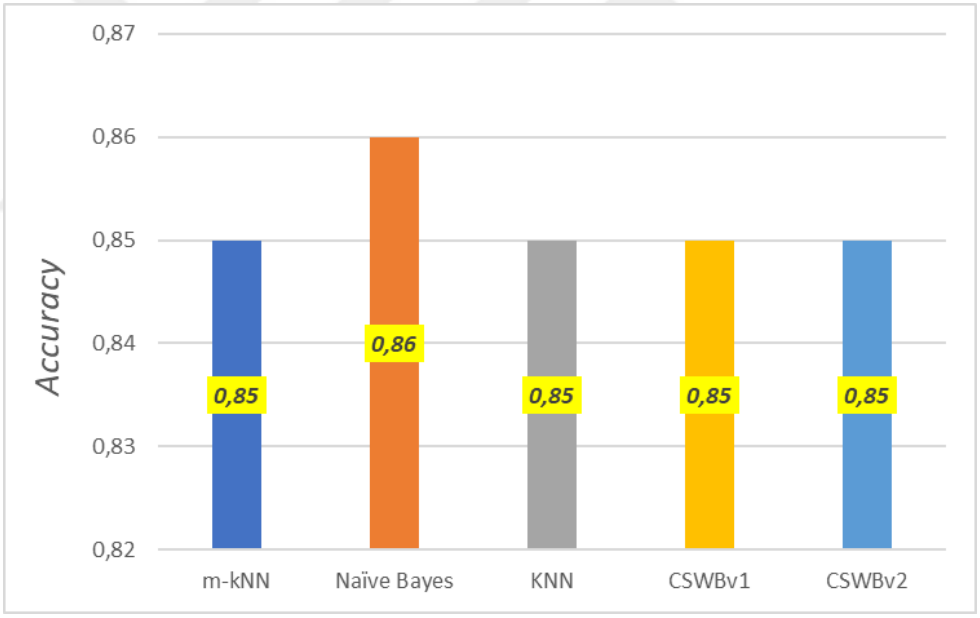

Figure 3.13: Accuracy for SEA in Experiment 2

tivation for including Naive Bayes in CSWB-e2 is based on our previous observation [4] that it contributed well to the performance of CSWB.

The first newly added classifier is K*, which is a nearest neighbor based classifier [5]. In K*, entropy is used as the distance measure. The approach to calculate the

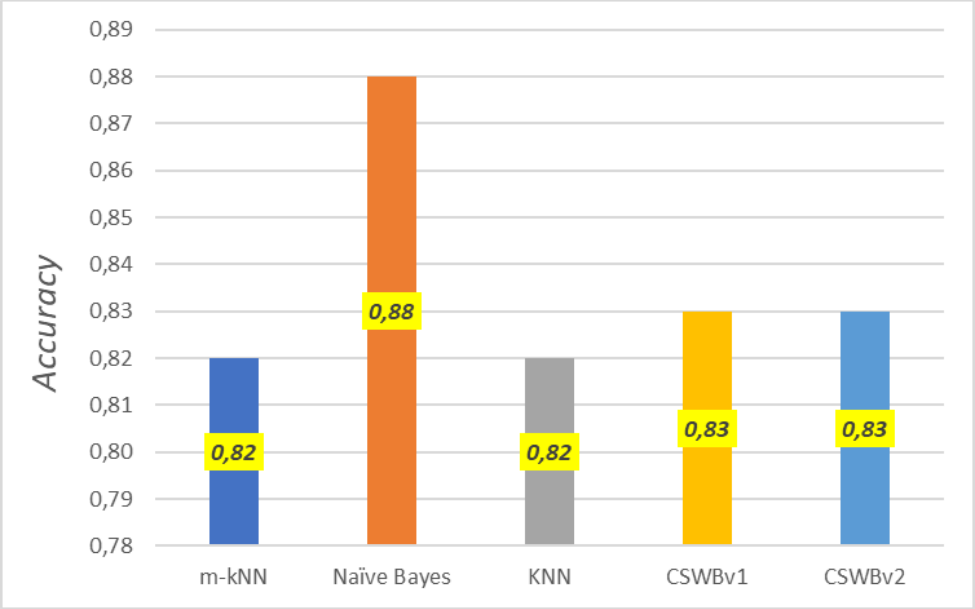

Figure 3.14: Accuracy for Hyperplane in Experiment 2

distance between two instances is inspired from information theory, and the main idea is that this distance can be viewed as the complexity of transforming one instance into another.

The second classifier that we included in our hybrid model is C4.5 [6], which is a decision tree based classifier, and is the successor of the ID3 algorithm. The algorithm uses the concept of information entropy for node selection. For each node of the decision tree, C4.5 selects the attribute that most effectively splits list of the training data samples. To this aim, C4.5 selects the attribute with the highest information gain. This process is applied recursively on the partitioned data. As in CSWB, in CSWB-e and in CSWB-e2, we may consider two different voting schemata in the assembler. In [4], it was already shown that the voting under *current accuracy* model performed well. Hence in our experiments, we use voting with current accuracy schema among the participating classifiers.

3.2.1 Experiments

In our previous work [4], we compared the classification accuracy performance of m-kNN against MC-NN [20] and VHT [21]. We analyzed CSWB classifier on several

real-world and synthetic data sets. We compared the accuracy performance of CSWB in comparison to kNN and Naive Bayes when applied as individual classifiers.

In this work, we conducted two new sets of experiments. In the first one, we analyzed the effect of window size parameter on three new data sets for m-kNN. Additionally, we detailed the analysis on one of the data sets in terms of precision, recall and f-Score metrics. In the second experiment, we analyzed the accuracy performance of CSWB-e and CSWB-e2 under different parameter settings against m-kNN, K*, C4.5, Naive Bayes classifier and CSWB.

3.2.2 Analysis on the Effect of Window Size for m-kNN

In our first experiment, we analyzed the accuracy of m-kNN on three new data sets:

- **Avila:** This is data set includes 800 images of the "Avila Bible" which is a giant Latin copy of the whole Bible and produced during the XII century between Italy and Spain. The pages are written by 12 copyist and each pattern has 10 features. The data have been normalized by using Z-normalization method. There are 10430 samples in training data set and 10437 samples in testing data set [9].

- **Poker Hand:** In this data set, there are records representing a hand of five playing cards drawn from a standard deck of 52. Each of the cards has two attributes as suit and rank. Therefore there are 10 predictive attributes in the data set. The last attribute for an instance is the class describing the *Poker Hand* [10].

- **Landsat Satellite:** This data set contains multi-spectral values of pixels by 3x3 neighborhood in a satellite image. There are 6435 instances in this data set and each line contains the pixel values in the four spectral bands of each of the 9 pixels in 3x3 neighborhood. The label of each instance shows the classification of the central pixel [11].

[9] https://archive.ics.uci.edu/ml/datasets/Avila
[10] https://archive.ics.uci.edu/ml/datasets/Poker+Hand
[11] https://archive.ics.uci.edu/ml/datasets/Statlog+%28Landsat+Satellite%29

In order to analyze the effect of window size on the classification accuracy, m-kNN is run under several window size values for *k=5* and *k=10* parameters. The window size values we have used in this experiment are {10, 20, 30, 40, 50, 75, 100, 200, 300, 500}.

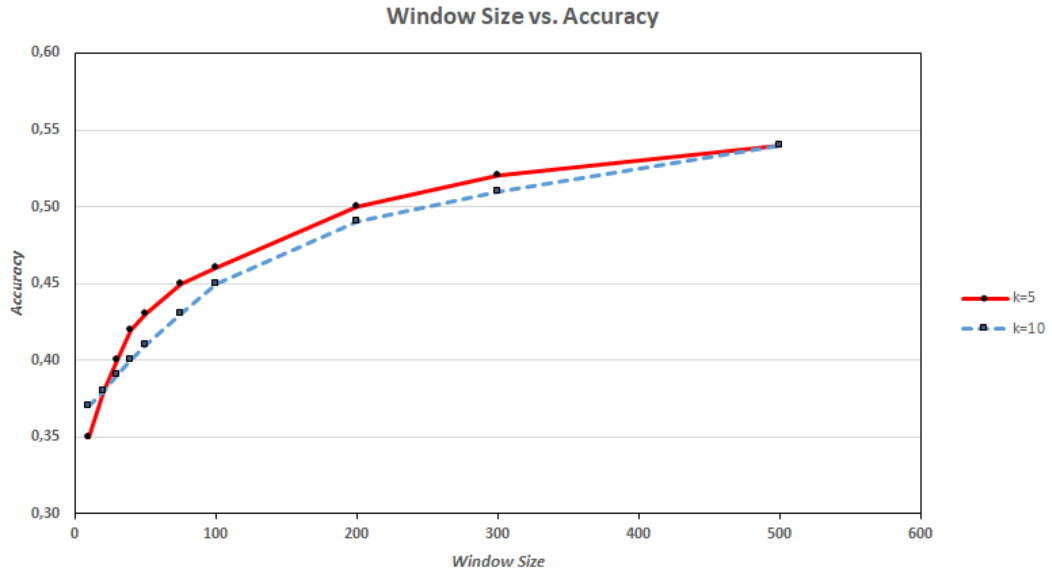

Figure 3.15: Accuracy performance for m-kNN on Avila data set under varying window size values with k=5 and k=10

The accuracy values of m-kNN for Avila data set under increasing window sizes can be seen in Figure 3.15. In the figure it is seen that the accuracy values are increasing smoothly with the increase in window size and reach the top value for *window size=500*. The behaviour of the results are similar for *k=5* and *k=10* values but the slopes are slightly different.

In Figure 3.16, the accuracy values of m-kNN for Poker Hand data set under varying window size values are presented. According to the results, the accuracy values for Poker Hand data set are increasing with window size values for both *k=5* and *k=10* values but in some steps the accuracy is not affected by the change in window size and remain in the same levels.

In Figure 3.17, we can see the accuracy values for Landsat Satellite data set. The results reveal that the accuracy values are increasing in proportional to the window size values up to 100 and this is the maximum value for Landsat Satellite data set.

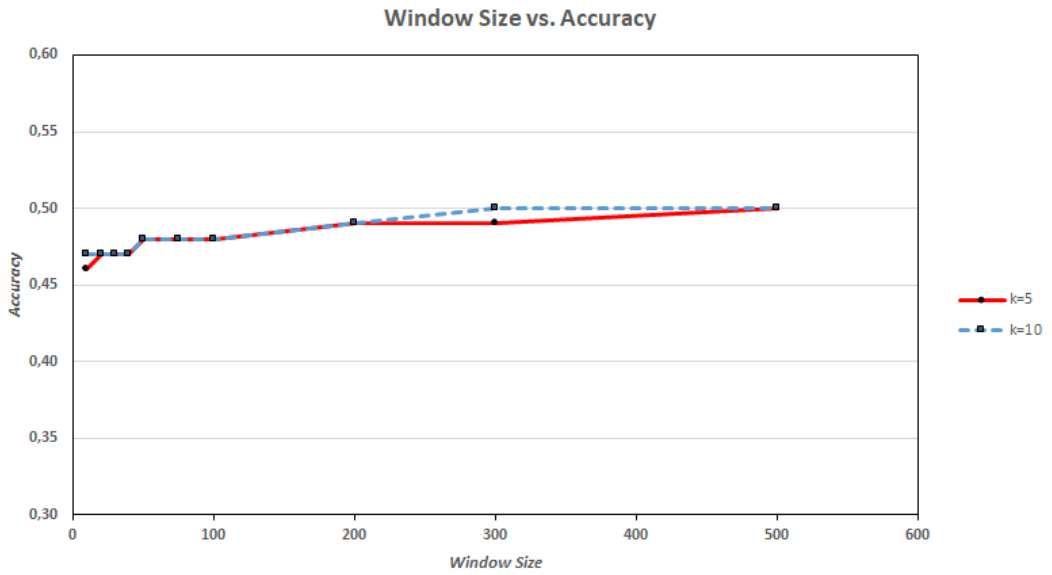

Figure 3.16: Accuracy performance for m-kNN on Poker Hand data set under varying window size values with k=5 and k=10

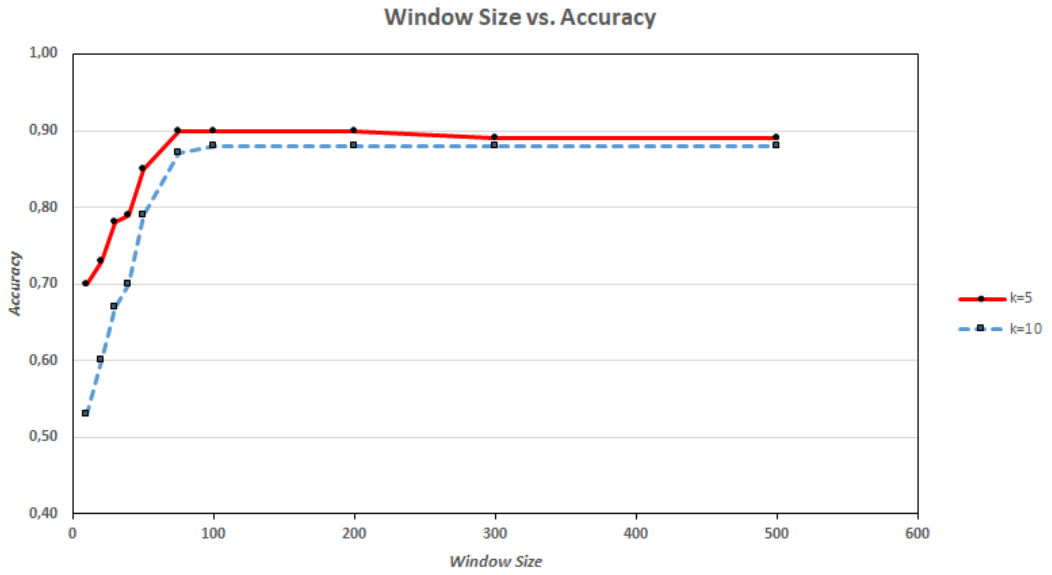

Figure 3.17: Accuracy performance for m-kNN on Landsat Satellite data set under varying window size values with k=5 and k=10

After that the accuracy values remains the same and the trend is similar for both of the *k=10* and *k=10* values. We can conclude that, for this data set, the classification accuracy is less sensitive to the change in the window size.

The accuracy performance of m-kNN for these data sets is compared against two other methods, MC-NN and VHT, which were used for comparison also in [4]. The comparison of the results for m-kNN, MC-NN and VHT are given Figure 3.18. In this analysis, we consider the best window size settings, which were determined in the above mentioned experiments. The results show that m-kNN gives the highest accuracy results for Avila and Landsat Satellite data sets, while the VHT has the highest accuracy value for Poker Hand data set, yet the accuracy value for m-kNN is close to VHT for this data set.

We further analyzed the classification performance for Landset Satellite data set in terms of precision, recall and f-Score metrics. We focused on *Cotton Crop*, *Grey Soil* and *Red Soil* classes, which have higher occurance in the data set.

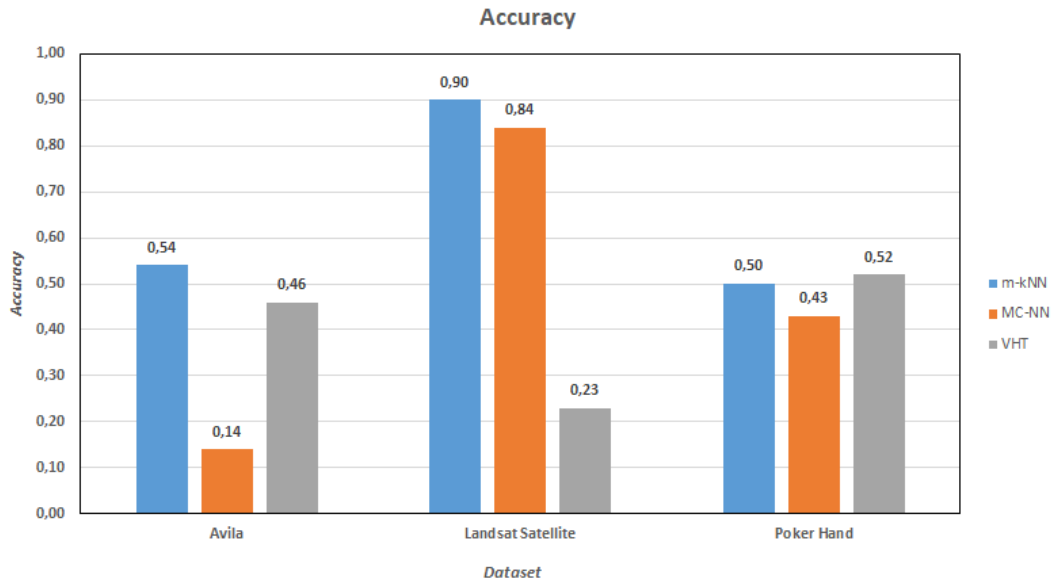

Figure 3.18: Comparison of accuracy performance of the methods for Avila, Landsat Satellite and Poker Hand data sets

The precision, recall and f-Score values for *Cotton Crop* class in Landsat Satellite data set under varying window size is given in Figure 3.19 and Figure 3.20. According to the results, with *k=5*, the recall and f-Score values are increasing up to *window size=100*. On the other hand, precision has a different behavior such that an increase is followed by a decrease and then stable value. F-score value remains around the same level where recall continues to increase up to *window size=300*. For *k=10*, we

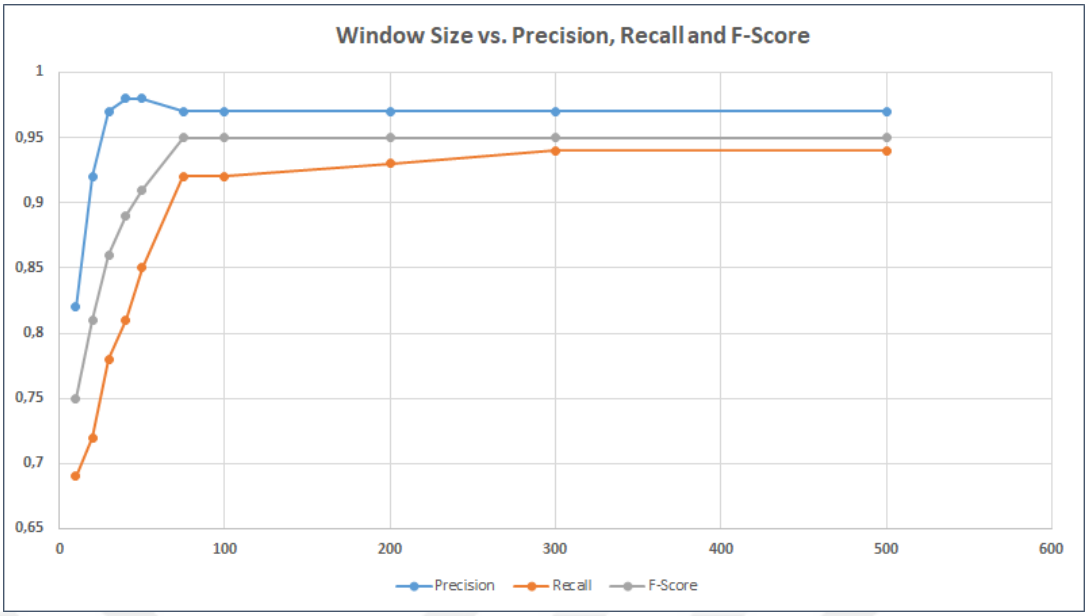

Figure 3.19: Precison, recall and f-score values for m-kNN for *Cotton Crop* class in Landsat Satellite under varying window size with k=5

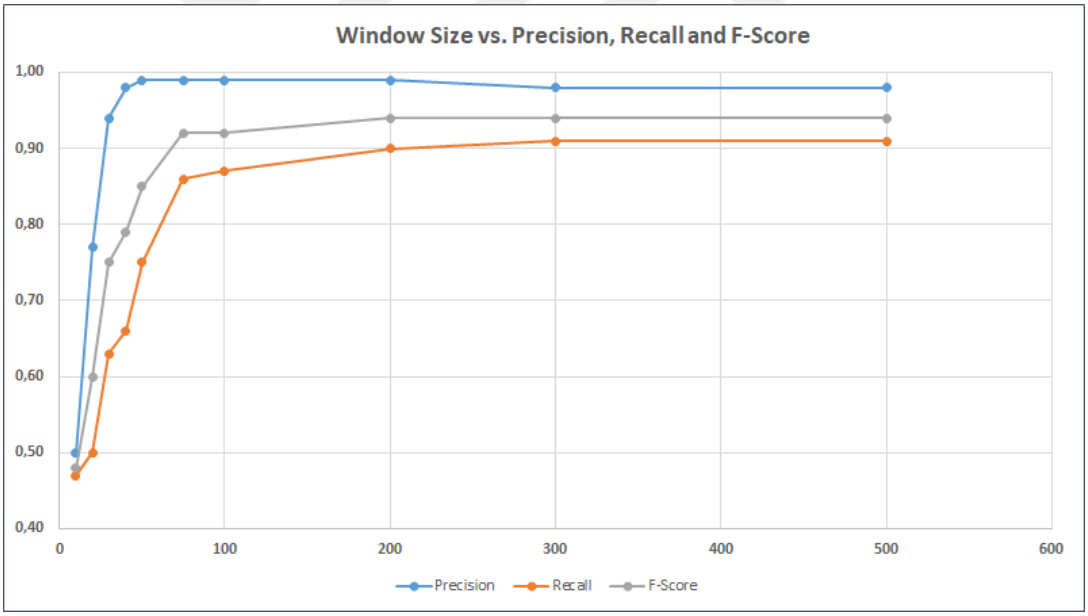

Figure 3.20: Precison, recall and f-score values for m-kNN for *Cotton Crop* class in Landsat Satellite under varying window size with k=10

see a similar yet smoother increase and stability in all three metrics.

The precision, recall and f-score results for *Grey Soil* class under *k=5* and *k=10* are

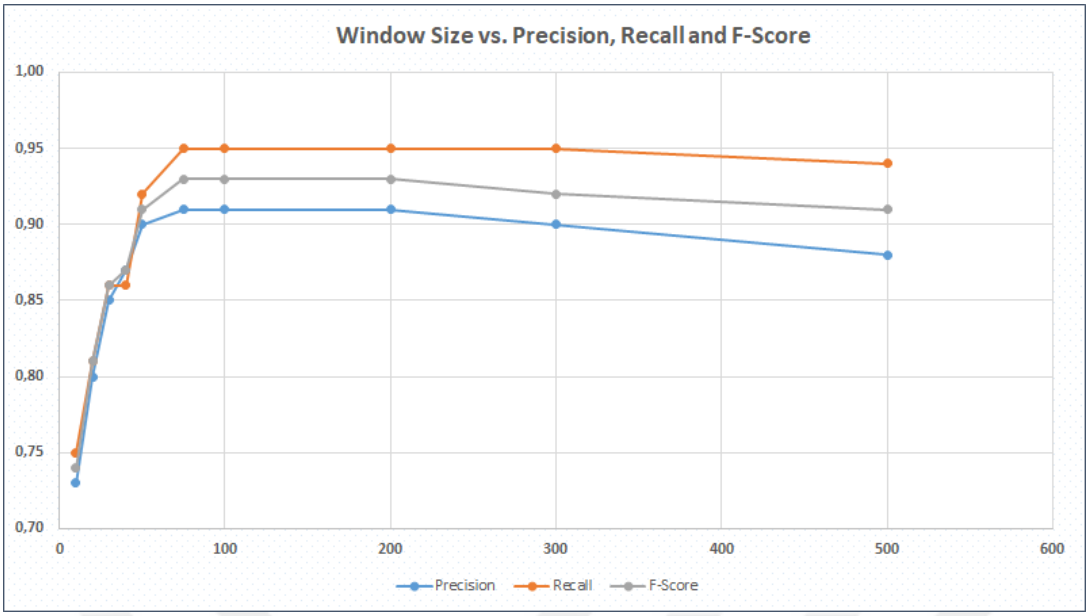

Figure 3.21: Precison, recall and f-score values for m-kNN for *Grey Soil* class in Landsat Satellite under varying window size with k=5

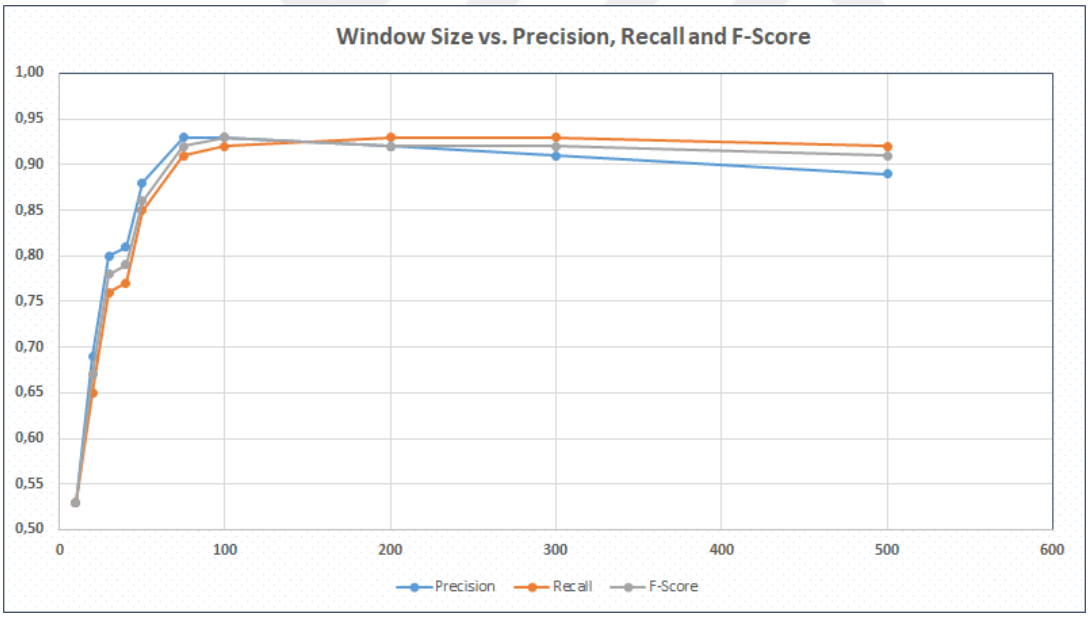

Figure 3.22: Precison, recall and f-score values for m-kNN for *Grey Soil* class in Landsat Satellite under varying window size with k=10

presented in Figure 3.21 and Figure 3.22. According to the results, with *k=5*, the values are increasing up to *window size=75*. Then for all the metrics, the values remain stable followed by a decrease as the window size gets larger. The amount

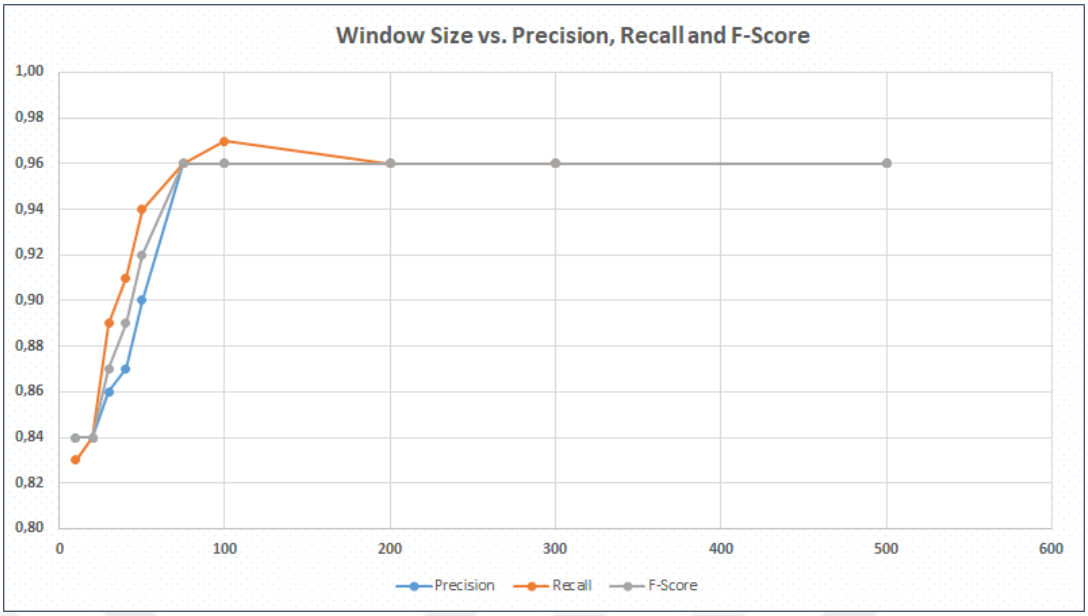

Figure 3.23: Precison, recall and f-score values for m-kNN for *Red Soil* class in Landsat Satellite under varying window size with k=5

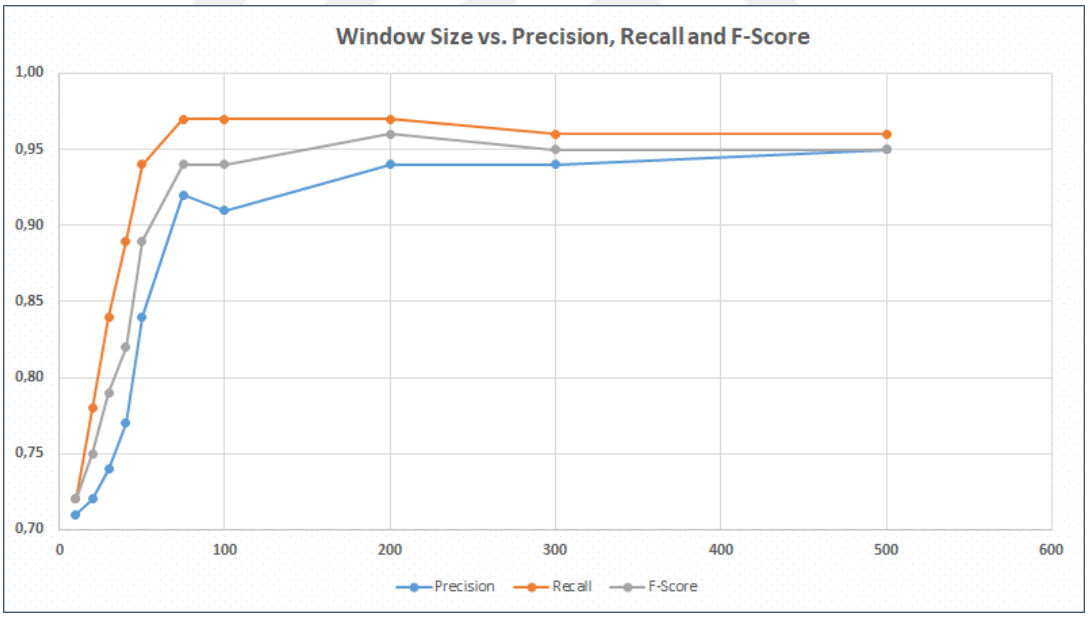

Figure 3.24: Precison, recall and f-score values for m-kNN for *Red Soil* class in Landsat Satellite under varying window size with k=10

of decrease is observed to be higher for precision. The results reveal that the gap between the precision, recall and f-Score values is smaller under *k=10*.

The precision, recall and F-Score values of m-kNN for "Red Soil" class in Landsat Satellite data set changing with the window size can be seen in Figure 3.23 and Figure 3.24. For the first graph for "Red Soil" class in Landsat Satellite data set, the values are increasing up to *window size=75* and then goes on a straight line where recall continues to increase up to *window size=100* and then meets the same line with precision and F-score. For the second graph recall and F-Score increases up to *window size=100* and precision has a decrease after *window size=100*. Then these three values get closer up to *window size=500*.

As a summary, it is observed that the accuracy and other three metric of precision, recall and f-score values increase in parallel to the increase in the window size up to certain values. The stability points change for each data set.

3.2.3 Analysis on Accuracy Performance of CSWB-e and CSWB-e2

In our next set of experiments, we investigate the accuracy performance of the proposed hybrid methods, *CSWB-e* and *CSWB-e2*, under parameters of k={5, 10} and varying window sizes in comparison to individual classifiers and *CSWB*. We have used the data sets from [4], which are Air Quality [12], Appliances Energy Prediction [13], Electricity Market [14], Human Activity Recognition [15], Forest Cover Type [16], SEA [17] and Hyperplane [18]. Additionally, we report the results for Avila, Poker Hand and Landat Satellite data sets, which are described in Section 3.2.2.

The accuracy results under varying *k* and *window size* parameters are presented in Table 3.3. The *window size* values are selected for each data set differently as those that provide the best results for the data set. When we compare the results under the same the parameter values we can see that the proposed hybrid methods, CSWB-e and CSWB-e2, perform better in 6 of the 10 data sets, and in 26 of the 40 cases.

[12] https://archive.ics.uci.edu/ml/datasets/Air+quality
[13] https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction
[14] https://www.openml.org/d/151
[15] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
[16] https://archive.ics.uci.edu/ml/datasets/Covertype
[17] http://www.liaad.up.pt/kdus/downloads/sea-concepts-dataset
[18] https://www.win.tue.nl/~mpechen/data/DriftSets/hyperplane1.arff

Table 3.3: Accuracy performance comparison for CSWB, CSWB-e and CSWB-e2

Dataset	k	win.size	m-kNN	K*	C4.5	CSWB	CSWB-e	CSWB-e2
AirQual.	5	250	0.77	**0.79**	0.75	0.74	**0.79**	**0.79**
AirQual.	10	250	0.77	**0.79**	0.75	0.76	**0.79**	**0.79**
AirQual.	5	500	0.77	**0.79**	0.76	0.77	**0.79**	**0.79**
AirQual.	10	500	0.77	**0.79**	0.76	0.78	**0.79**	**0.79**
App.En.Pre.	5	25	0.64	**0.69**	0.65	0.64	0.67	0.67
App.En.Pre.	10	25	0.62	**0.69**	0.65	0.63	0.68	0.67
App.En.Pre.	5	50	0.64	**0.69**	0.65	0.63	0.67	0.66
App.En.Pre.	10	50	0.62	**0.69**	0.65	0.63	0.68	0.66
Elec.Mar.	5	25	0.77	0.90	**0.92**	0.76	0.81	0.79
Elec.Mar.	10	25	0.70	0.90	**0.92**	0.77	0.81	0.79
Elec.Mar.	5	50	0.78	**0.91**	**0.91**	0.77	0.81	0.80
Elec.Mar.	10	50	0.73	**0.91**	**0.91**	0.78	0.81	0.79
Hum.Act.Rec.	5	250	0.91	0.38	0.93	0.76	**0.94**	0.91
Hum.Act.Rec.	10	250	0.88	0.38	0.93	0.75	**0.94**	0.89
Hum.Act.Rec.	5	500	0.92	0.37	0.93	0.89	**0.94**	0.92
Hum.Act.Rec.	10	500	0.91	0.37	0.93	0.91	**0.94**	0.91
For.Cov.Type	5	250	0.86	**0.93**	0.87	0.84	0.91	0.90
For.Cov.Type	10	250	0.81	**0.93**	0.87	0.83	0.91	0.87
For.Cov.Type	5	500	0.90	**0.94**	0.91	0.82	**0.94**	0.92
For.Cov.Type	10	500	0.87	**0.94**	0.91	0.87	0.93	0.91
SEA	5	250	0.81	0.82	0.82	0.79	**0.84**	**0.84**
SEA	10	250	0.83	0.82	0.82	**0.85**	0.84	**0.85**
SEA	5	500	0.82	0.84	0.84	0.84	0.85	**0.86**
SEA	10	500	0.85	0.84	0.84	0.82	**0.86**	**0.86**
Hyperplane	5	250	0.76	0.71	0.70	0.69	0.76	**0.81**
Hyperplane	10	250	0.80	0.71	0.70	0.78	0.81	**0.83**
Hyperplane	5	500	0.78	0.74	0.73	0.81	0.78	**0.83**
Hyperplane	10	500	0.82	0.74	0.73	0.83	0.78	**0.85**
Avila	5	300	0.52	0.56	0.57	0.52	**0.60**	0.56
Avila	10	300	0.51	0.56	0.57	0.51	**0.60**	0.56
Avila	5	500	0.54	0.61	0.62	0.55	**0.64**	0.60
Avila	10	500	0.54	0.61	0.62	0.54	**0.64**	0.60
PokerHand	5	300	**0.49**	0.46	0.45	**0.49**	0.48	**0.49**
PokerHand	10	300	**0.50**	0.46	0.45	0.49	0.49	0.49
PokerHand	5	500	**0.50**	0.46	0.45	0.49	0.49	0.49
PokerHand	10	500	**0.50**	0.46	0.45	0.49	0.49	**0.50**
Landsat Satellite	5	100	0.90	**0.92**	0.85	0.90	**0.92**	**0.92**
Landsat Satellite	10	100	0.88	**0.92**	0.85	0.88	0.91	0.91
Landsat Satellite	5	200	0.90	**0.91**	0.86	0.90	**0.91**	**0.91**
Landsat Satellite	10	200	0.88	**0.91**	0.86	0.88	**0.91**	**0.91**

In several of the data sets, individual classifiers perform better, but the accuracy performance of the proposed hybrid classifiers are very close. Hence, on the overall, we observe a preferable performance for the hybrid methods, providing good accuracy for any kind of data. Depending on the nature of the data set, the classification accuracy of *CSWB-e* and *CSWB-e2* may vary. In general, among *CSWB-e* and *CSWB-e2* there is almost a tie situation.

CHAPTER 4

T-GSC (TWO PHASE GRAPH BASED SHORT TEXT STREAM CLUSTERER)

4.1 Proposed Method: T-GSC (Two Phase Graph Based Short Text Stream Clusterer)

The proposed method, T-GSC, is based on a sliding window approach. It includes two phases of clustering in order to refine the clustering further. The method has two parameters, as batch count and batch size. The combination of batch count and batch size parameters constitute the window size. The overview of T-GSC workflow is illustrated as in Figure 4.1.

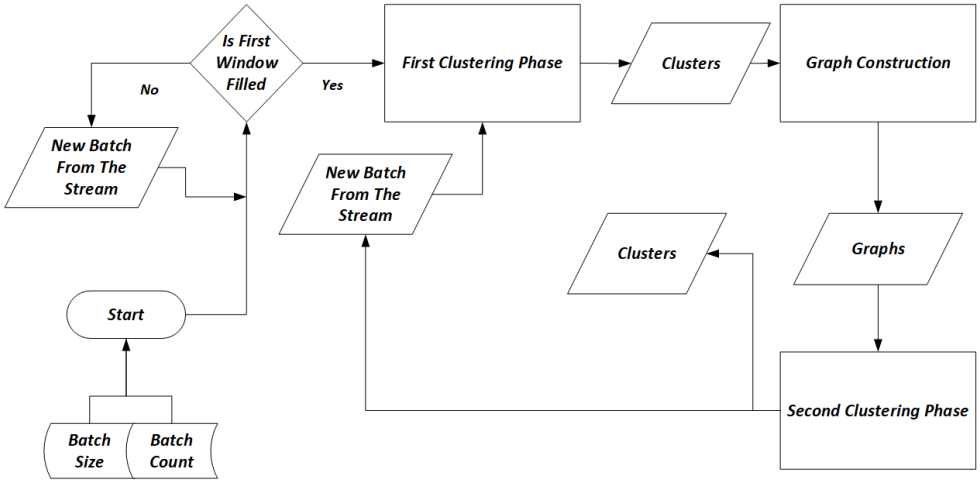

Figure 4.1: The workflow of T-GSC

According to this flow, the instances from data stream are put into the current sliding window as batches. When the first window is filled with the batch of instances, clus-

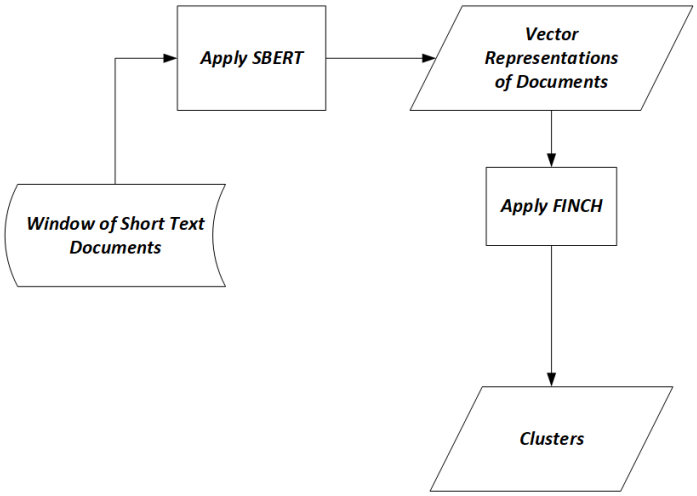

Figure 4.2: T-GSC First phase of clustering

tering process is started. T-GSC includes two phases of clustering. The first phases clusters the short terms through their vectoral representations (embeddings). By using the output of the first round, graph structures are constructed. The second phase takes these graphs as inputs. Assignments of short texts to clsuters is completed at the end of the second round. This process is repeated along the data stream after the sliding window is updated.

The details of the first phase of clustering is illustrated in Figure 4.2. In this phase, the sentence embeddings for the short text documents are generated by applying SBERT. The vector representations of short texts are clustered by using FINCH algorithm, as the hierarchicial clusterer of the first round. At the end of the first phase, initial version of clusters containing short text documents are obtained.

In the second phase of the clustering, node embeddings are generated by using FastRP from the graphs obtained in the previous phase. As the next step, a second round FINCH is applied on these vector representations These results are merged with the assignments of the previous iteration. The steps in the second phase are illustrated in Figure 4.3.

Processing a window of short text documents contains these two clustering phases

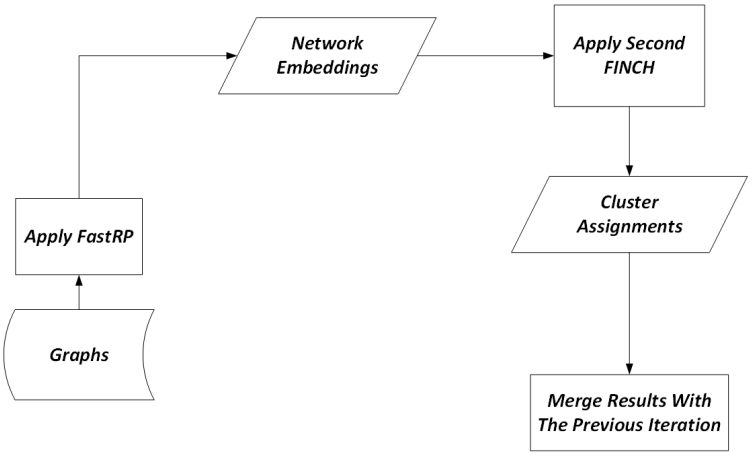

Figure 4.3: T-GSC Second phase of clustering

Algorithm 2 T-GSC

```
for Each batch from the short text stream do
    Add short text documents to the first window
    if The initial window is filled then Process this window with Algorithm 3
    end if
    Remove the outdated batch from the window
end for
for Each batch from the short text stream do
    Add new batch to the window
    Process the window with Algorithm 3
    Remove the outdated batch from the model
end for
```

by applying FINCH. The whole process of T-GSC is also given in Algorithm 2. The steps of the processing one window of short text streams is presented as a pseudocode in Algorithm 3.

The details of the graph construction step are given in Algorithm 4. For each cluster generated at the end of the first phase, a graph is constructed. In the graphs, the nodes represent short text documents in clusters and the nodes having words in common have an edge connecting them. The weight of these edges are determined by the number of common words.

Algorithm 3 Process One Window Of Short Text Documents

```
for Each document in window do
    Generate sentence embeddings for short text documents by using SBERT
end for
Apply first round of FINCH algorthm to cluster documents
Generate graphs from these clusters
for Each graph in current model do
    Apply FastRP and generate network embeddings
end for
Apply second FINCH to these vector representation of graphs
```

Algorithm 4 Graph construction

```
for Each cluster generated in the first phase do
    Add nodes to represent short text documents
    Compare the short text documents in cluster as pairs
    if There are common words between each document then
        Add an edge between nodes representing short text documents
        Edge weight:= Number of common words
    end if
end for
```

At the end of the second phase of clustering, the generated clusters are merged with the previous ones. This merge operation is held by using a similarity matrix, where the rows are the new cluster labels, the columns are the previous clusters and the entries are the number of common short text documents. A current cluster and a previous cluster are matched if they mutually have the maximum similarity with each other. A current cluster and the previous cluster may have the maximum similarity with other current and previous clusters. Therefore, to determine a matching, we check whether the maximum similarity is mutual. The content of the matched clusters are merged and they are removed from the matrix. The similarity checking and matching continues until there is no change in the similarity matrix. After this, the remaining clusters are matched to the ones with maximum similarity values. The count of current clusters may be different from the count of previous clusters. If the count of current clusters is more than the count of previous ones, some of them will

Algorithm 5 Merging the clusters

```
Construct a similarity matrix between current clusters and previous clusters
for Each current cluster do
    Find the previous cluster having maximum similarity
    if Current cluster has also the maximum similarity for the previous cluster then
        Match these clusters
        Merge and remove these clusters from the matrix
    end if
    Repeat this until there is no change
end for
if There remains clusters in similarity matrix then
    for Each remaining cluster do
        Find the previous cluster having maximum value
        Match these two clusters
    end for
end if
The remaining previous clusters are discarded
```

remain unmatched and these are the newly emerged clusters. On the other hand, if the count of previous clusters is more than the current ones, this causes some previous clusters unmatched which means no more instance will be assigned to these clusters. The process is given in Algorithm 5.

When the second phase is completed, sliding window is updated. According to our sliding window based approach, we repeat two-phase clustering process for each window. In each iteration, we remove a batch of short text documents from the rear of the window after clustering is completed and we add a batch of short text documents from data stream to the front. The structure of a window and how it is updated is illustrated in Figure 4.4.

In order to analyze the clustering capability of T-GSC, a sample trace of the method is visualized as given in Figure 4.5. For the visualization, 100 tweets are sampled from Tweets dataset and the algorithm is applied for four iterations on this sample set.

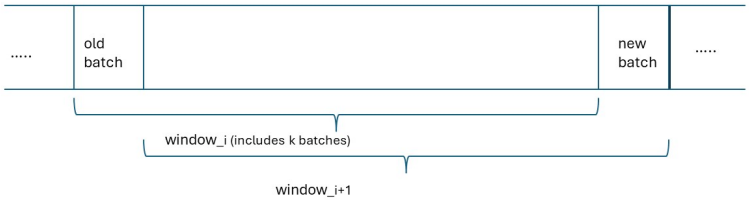

Figure 4.4: T-GSC Sliding Window Based Approach

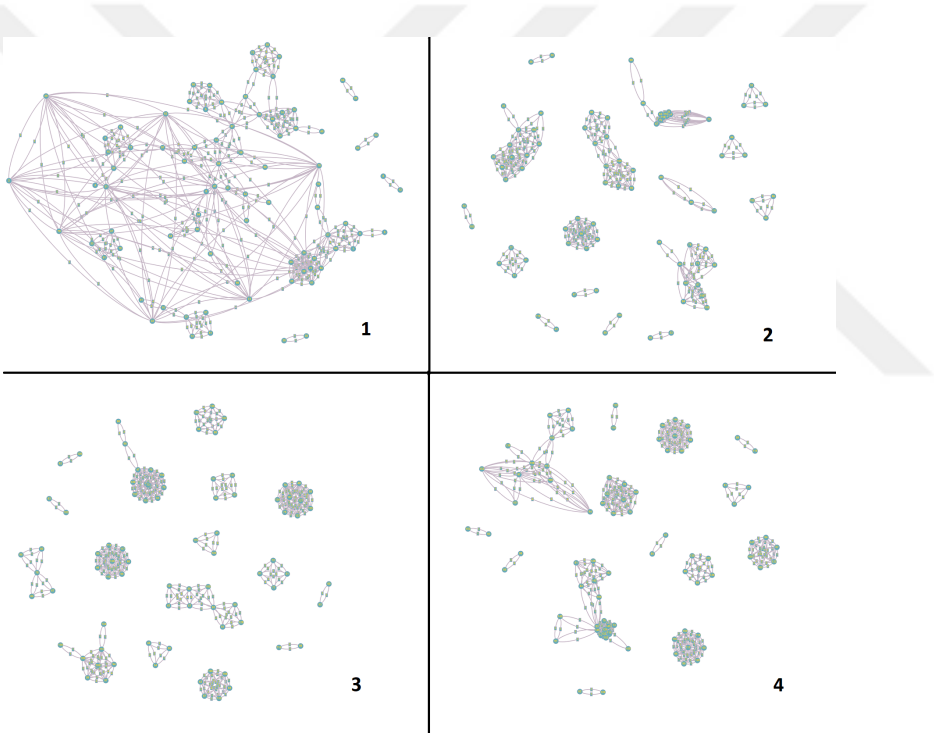

Figure 4.5: Sample Iterations of Clustering With T-GSC

hese graphs are constructed by using an online tool [1]. In this figure, it can be seen that clusters can be clearly identified at the end of the first iteration and the clusters remain clear for the next three iterations.

[1] https://graphonline.ru/en

4.2 Experiments and Results

4.2.1 Datasets

The proposed method is implemented in Python, and the experiments are conducted with four benchmark datasets. The details of the datasets are given in Table 4.1.

Table 4.1: The Details of Datasets Used in the Experiments

Dataset	No. of Instances	No. Of Topics	Avg. No. of Words
Tweets	30322	269	7,97
Google News	11108	152	6,23
NTS	61248	438	6,67
StackOverflow29K	29000	11307	8,59

There are several datasets in the literature consisting of short texts obtained from Twitter [2] and used in data stream clustering studies. In this study, we use the tweet collection from 2011-2015 obtained from Text REtrieval Conference (TREC) microblog track[3]. This dataset contains 30322 tweets and 269 topics. The tweets are labelled and assigned to these 269 topics. The average number of words in this dataset is 7.97.

The second dataset is Google News, which includes news articles having titles and snippets [4]. There are 11,108 articles and these articles are grouped into 152 topics. The average number of words in these short documents is 6.23. This dataset is generated by Yin and Wang, and used in [62].

The third dataset is obtained from StackOverflow, and presented by Rakib et al. [5]. We use a partition of data containing 29.000 documents from this dataset, named as StackOveflow29K .There are 11,307 clusters in this dataset and the average number of words in the instances is 8.59.

The final dataset is NTS [6], which is created by combining the texts of the abovemen-

[2] https://twitter.com
[3] https://trec.nist.gov/data/microblog.html
[4] https://google.com/news/
[5] https://github.com/rashadulrakib/short-text-stream-clustering/
[6] https://rb.gy/zetjaz

tioned two datasets (Google News and Tweets) and another dataset from StackOverflow [7], and hence named accordingly. The dataset consists of 61,428 texts distributed into 438 clusters. The average number of words for the short text documents in this dataset is 6.67.

4.2.2 Comparison with the State-of-The-Art Methods

For the comparison of the results of our proposed method with the state-of-the-art methods, NMI measure is used, as most of the results in the literature are reported in NMI. For T-GCS, the best NMI results obtained from parameter tuning analysis are considered. In the experiments of other methods that we run the provided implementations, we use the offered or default parameters. The methods whose implementations are used to obtain results of tests in our experiments are DP-BMM, DP-BMM-FP, MStream, MSTreamF, OSDM and OSGM. For DP-BMMM, α = 0,3 and β = 0,02 and For DP-BMM-FP, α = 0,5 and β = 0,002 are used as parameters of the models. In both methods, iteration count is taken as 10. In MStream/MStreamF, iterNum = 5, sampleNum = 1, wordsInTopicNum = 5 , α = 0,03 and β = 0,03 are taken. For OSDM, α = 0,002, β = 0,0004 and λ = 0,000006 are used as model parameters. For OSGM, α = 0,05 and β = 0,0002 are taken. For the Tweets dataset, NMI values for reported in the original articles of the methods are used for comparison. The results are presented in Table 4.2. In the table, it is seen that the proposed method can reach a higher level of accuracy in terms of NMI values for the Tweets dataset.

For the StackOverflow29K dataset, to obtain the results for the SoTA methods, the implementations provided by the authors of these methods in GitHub [8] [9] [10] [11] [12] are run. The best NMI scores obtained for each of the state of the art methods are given in Table 4.2. According to the results, T-GSC outperforms the other methods by more than 20%. Additionally, some of the NMI results are very low, which may be due to

[7] https://github.com/jacoxu/STC2/blob/master/dataset/StackOverflow.txt
[8] https://github.com/junyachen/BMM
[9] https://github.com/JayKumarr/OSDM
[10] https://github.com/rashadulrakib/short-text-stream-clustering/tree/master/BatchClustering
[11] https://github.com/jackyin12/MStream
[12] https://github.com/JayKumarr/OSGM

Table 4.2: Comparison with the State-of-The-Art methods (in NMI). The top-2 are written in bold fonts.

Method	Tweets	StackOverflow29K	Google News	NTS
T-GSC	**0,87**	**0.86**	0,83	**0,80**
DP-BMM [33]	**0,86**	0,55	**0,88**	0,74
DP-BMM-FP [33]	0,85	**0,70**	0,86	0,40
LAST [35]	0,84	–	0,84	–
MStream [36]	0,84	0,37	0,83	0,70
MStreamF [36]	0,82	0,37	0,79	0,71
DHP+ICF [63]	0,82	–	–	–
SDHP+ICF [63]	0,83	–	–	–
DCSS [47]	0,81	–	0,63	–
OSGM [48]	0,85	0,63	0,81	**0,81**
OSGM-ES [48]	0,85	–	0,82	–
OSDM [34]	–	0.23	0,81	**0,81**
Rakib et al. [44]	–	0,60	**0,86**	0,76

the structure of the data, where the number of topics in this dataset is relatively high with respect to the other datasets. Thus, the average count of short text documents in clusters gets lower and some methods may have difficulty in detecting the relation between instances. As a result, this may degrade the accuracy of these methods.

We also made experiments with Google News and compared our results with other methods. T-GSC reached high accuracy values while DP-BMM is the leader for this dataset. The count of short text documents in this dataset is fairly lower than the other datasets. T-GSC can produce better results with relatively larger window size values. In addition to the low count of instances, the average number of words in Google News is also lower than the other datasets. These properties of Google News dataset prevent T-GSC to discover the relations between short text documents and reach higher accuracy values.

For the NTS dataset, similarly, the NMI results of the SoTA methods are obtained by running the methods' codes in GitHub, and the best NMI scores obtained for each method are given in Table 4.2. In the table it is seen that, T-GSC outperforms the other methods except for OSGM and OSDM. The score of T-GSC follows these two methods as the second best result by a difference of 0,01. NTS dataset is obtained by combining the datasets of Google News, StackOverflow and Tweets, and resulting collection is shuffled randomly [44]. T-GSC is based on the graphs obtained by the common words between short text documents. Since the texts are obtained from different datasets and shuffled in a random way, the ratio of the common words may decrease in the first phase of clusters obtained with FINCH. Also, T-GSC has a sliding window mechanism, and the generation method of NTS dataset may prevent the related short text documents to be in the same window. Additionally, Google News dataset has a lower mean for the number of words, which may effect the mean of the overall dataset. This also may affect the clustering quality of T-GSC.

4.2.3 Parameter Tuning Experiments

In order to obtain the optimum values for the batch count and the batch size, a set of parameter tuning experiments are conducted. For the Tweet dataset, the results are given in Table 4.3. Similarly, for the StackOverflow29K, NTS and Google News datasets, the parameter tuning analysis results are given in Table 4.4, Table 4.5, and Table 4.6, respectively. Note that batch count and batch size values used are different for each of the datasets. This is basically due to the difference in the total size and the nature of the datasets.

For the Google News dataset, we present a more detailed analysis having homogeneity and completeness metrics as well. In the results, when the window size gets larger, the proposed method can reach to better clustering quality measures. Also, it can be said that for the same size of window, larger batch sizes can give a better performance. In the results, it is also seen that the completeness and homogeneity values are changing in parallel to the NMI values.

Using window size as a standalone parameter, which is independent of batch count, was another option to configure the proposed approach. Hence, another experiment is

Table 4.3: Parameter tuning experiments on the Tweets dataset

Batch Count	Batch Size	NMI
5	2750	0,79
5	3750	0,82
10	2250	0,85
10	2500	0,85
15	1700	0,87
15	1750	0,86
20	1100	0,83
20	1250	0,85
30	500	0,74
30	750	0,82
100	250	0,83
100	275	0,85

Table 4.4: Parameter tuning experiments on the StackOverflow29K dataset

Batch Count	Batch Size	NMI
10	500	0,67
10	1000	0,74
10	1500	0,79
10	1750	0,81
10	2500	0,86
10	2600	0,86
20	500	0,73
20	700	0,77

conducted to analyze the effect of setting window size independent of the batch count. The experiments are conducted on Google News and StackOverflow29K datasets under varying batch size and window size settings. The results of the experiment are given in Table 4.7. In the table, we observe that using window size as a parameter

Table 4.5: Parameter tuning experiments on the NTS dataset

Batch Count	Batch Size	NMI
10	3000	0,76
10	3500	0,77
10	4000	0,78
10	4500	0,79
10	4750	0,80
10	4800	0,79
10	5000	0,78

independent of batch size (and batch count) does not bring any improvement in clustering accuracy. In the original parameter structure, T-GSC can reach the NMI value of 0,70 for the Google News dataset with batch size 300 and batch count 25, whereas the accuracy values for the window based configuration having the same batch size are lower. Similarly, for the StackOverflow dataset, T-GSC can reach the NMI value of 0,74 when batch the size is 1000 and batch count is 10, whereas the NMI values are lower for the window size based configurations. These results may indicate that the window contains more related documents when the window size is a multiple of batch size. Changing this approach may degrade the similarity and integrity of short text documents in current window.

4.2.4 Ablation Analysis

In order to analyze the effect of second phase of clustering on the clustering performance, another experiment is conducted. Note that in first phase of T-GSC, FINCH clustering is applied on the vector representation of short text documents which are obtained with SBERT. In the second phase, graphs are constructed from the clusters obtained in the first level, and FINCH clustering is applied, this time on these graphs. To measure the effect of this second phase for clustering quality, we transformed our method into a single level clustering approach without generating graphs and applying second phase of clustering. The analysis is performed on Google News and Stack-

Table 4.6: Parameter tuning experiments on the Google News dataset

Batch Count	Batch Size	NMI	Homogeneity	Completeness
5	1600	0,81	0,80	0,82
5	1700	0,83	0,83	0,83
5	1800	0,82	0,82	0,82
5	1825	0,81	0,81	0,81
10	800	0,77	0,77	0,78
10	900	0,80	0,81	0,80
10	950	0,82	0,83	0,82
10	1000	0,82	0,83	0,81
25	300	0,70	0,70	0,70
25	350	0,77	0,77	0,77
25	375	0,80	0,80	0,79
25	400	0,81	0,82	0,81
30	200	0,62	0,62	0,63
30	250	0,70	0,70	0,70
30	300	0,77	0,78	0,75
30	325	0,79	0,79	0,78
50	100	0,52	0,52	0,52
50	150	0,67	0,67	0,66
50	175	0,74	0,75	0,72
50	200	0,79	0,80	0,79

Overflow29K datasets under varying batch count and batch size values. The results given in Table 4.8 show that there is a remarkable effect obtained by the second phase of clustering in T-GSC. When the batch count values are examined, it is seen that the second phase is more effective in small batch counts and batch size values. It can

Table 4.7: Experiment with window size independent of batch count

Dataset	Window Size	Batch Size	NMI
Google News	2000	300	0,33
Google News	3000	300	0,41
Google News	5000	300	0,54
Google News	7500	300	0,69
StackOverflow29K	3000	1000	0,58
StackOverflow29K	5000	1000	0,57
StackOverflow29K	7500	1000	0,60
StackOverflow29K	9000	1000	0,62

be observed that the improvement is decreasing when the windows are getting larger. Also it can be said that the effect of the second phase is more apparent in Google News with respect to StackOverflow and this may be related with the structure of datasets where the number of clusters in StackOverflow29K is relatively higher than in GoogleNews.

In order to analyze if there could be any improvement about the merge phase of T-GSC, another experiment is made. In this experiment, a threshold is added to the merge phase where the remaining documents in similarity matrix are merged. Experiment is made with different batch count and batch size values for Google News and Tweets dataset. The best results are taken into account and the results are given in Table 4.9. According to the results, this modification does not produce any improvements in terms of accuracy.

Another experiment for ablation analysis is also conducted to see if using another embedding method can increase the accuracy values for T-GSC. Instead of utilizing SBERT, TF-IDF is used to generate vectors for the short text documents. The results are given in Table 4.10. According to the results, while this version can reach higher NMI score for Google News dataset, its results are wors than the original version of T-GSC for other datasets. Since Google News is a relatively smaller dataset with respect to other ones, these results can indicate that using TF-IDF for larger dataset is not a proper method to capture the related documents.

Table 4.8: Comparison of Single and Two Phase Clustering for T-GSC (in terms of NMI values)

Dataset	B. Count	B. Size	1-Phase Clustering	2-Phase Clustering (T-GSC)
Google News	5	1600	0,66	0,81
Google News	5	1700	0,71	0,83
Google News	5	1800	0,77	0,82
Google News	10	800	0,70	0,77
Google News	10	900	0,75	0,80
Google News	10	1000	0,79	0,82
Google News	25	350	0,74	0,77
Google News	25	400	0,78	0,81
StackOverflow29K	10	1500	0,79	0,79
StackOverflow29K	10	1750	0,80	0,81
StackOverflow29K	10	2500	0,84	0,86
StackOverflow29K	10	2600	0,85	0,86
StackOverflow29K	20	700	0,77	0,77
Tweets	5	1000	0,78	0,82
NTS	5	1000	0,75	0,76
NTS	5	1200	0,74	0,76

4.2.5 Complexity Analysis

T-GSC has three main steps such as generation of vector representations by using embeddings, clustering phases by using FINCH and merging clusters obtained in consecutive iterations. To examine the space complexity of T-GSC, it can be considered that the complexity of the first phase is related with the complexity of SBERT embeddings and the complexity for the second step is mostly related with FINCH. Network embeddings for the second phase are generated by using FastRP and the time complexity for FastRP can be given as $O(O(n+m)\cdot k\cdot d)$ where d indicates dimensionality and k is the maximum power. When the merge phase is reviewed, the space complexity is proportional to the number of clusters and their instances obtained in the previous iteration and the current iteration. On the overall, T-GSC passes data once as in the limitations of data stream mining and the complexity can be defined as $O(n)$ where n is the number of short text documents in data stream.

Table 4.9: Accuracy of T-GSC with a threshold in merge phase

Dataset	T-GSC	T-GSC with Merge Threshold
Tweets	0,87	0,79
Google News	0,83	0,78

Table 4.10: Comparison of TF-IDF and SBERT versions for T-GSC (in terms of NMI values)

Dataset	T-GSC with TF-IDF	T-GSC with SBERT
Google News	0,89	0,83
StackOverflow29K	0,66	0,86
Tweets	0,85	0,87
NTS	0,79	0,80

To make an analysis about the time complexity and running time of T-GSC with other methods, an additional experiment is conducted on Google News dataset. The experiments are made on a system with Intel(R) Core(TM) i7-6700HQ CPU processor and 16 GB memory. The results of this experiment are give in Table 4.11.

According to the results, Rakib et al. is clearly faster than other methods and OSDM follows that method. T-GSC is not better than most of the methods according to running time scores but still it is competitive and this can be improved by making some modifications in T-GSC such making some operations such as generating embeddings or merging in a more parallel approach. Also, most of the time consumed during the execution is spent to generate embeddings for SBERT. This may degrade the time efficiency of T-GSC, whereas it increases the accuracy values since it is better for semantic similarity. On the other hand, this time complexity cost can be considered as a trade off for obtaining high accuracy.

Table 4.11: Comparison of our proposed method for running time in seconds with the-state-of-the-art methods

Method	**Running Time (s)**
T-GSC	185
DP-BMM	865
DP-BMM-FP	190
MStream	160
MStreamF	130
OSGM	105
OSDM	85
Rakib et al.	70

CHAPTER 5

CONCLUSION

5.1 CONCLUSIONS

Clustering and classification which are traditional tasks of data stream mining are studied in this thesis work. For data stream classification, new methods are proposed as enhancements for current techniques and approaches. Since our enhancements are based on sliding window approach and only the instances in the current window from the whole data stream is kept, they are scalable for data streams with huge sizes. The memory space required to execute these enhancements are proportional to the window size plus the number of class labels, as the instances having the current mean values for the features are maintained throughout the execution process of data stream.

Experiments are made to make an analysis of our enhancements and run them on several datasets. According to the results obtained from different data sets, our approaches have higher accuracy values with respect to other methods for several data sets. When the results of our experiments are reviewed, it can be seen that m-kNN has lower accuracy values for some datasets such as Electricty Market. This can be related with the poor association between the current state of the data and the behaviour pattern coming from the past. As a result it can be concluded that sharp changes in data streams can lower the accuracy performance of m-kNN and it can be preferred when there is a stronger linkage and a slight transition between the past and current state of the data.

Hybrid methods are also proposed in this study, *CSWB-e* and *CSWB-e2*, having improvements over *m-kNN* and *CSWB-e* presented in [4]. For the hybrid methods, our motivation comes from our previous observation for improvement potential via new

classifiers. To this aim, *K**, *C4.5* are included as the new individual classifiers into the assemblers. The results show that the proposed hybrid methods, especially *CSWB-e2*, outperforms the other models. As another contribution, the performance of *m-kNN* algorithm is anlayzed further on additional three data sets. These data sets have different nature from those studied in [4]. Two of them are image data sets, whereas one of them is a streaming set of tuples that represent hands in a poker game. The results show that *m-kNN* provides good performance in comparison to two previous solutions in the literature. Our experiments investigate further details such as the effect of window size on classification performance, and precision, recall and f-score results per class.

For the future work, further analysis could be made on failure cases to be able devise improvements and for CSWB classifier different classifiers from the literature can be combined with m-kNN to improve the accuracy performance further. In addition, developing parallel versions of classifiers may be studied to reach a higher performance. Mixed type data sets can be considered as an open research area for data streams, and working on enhancements for classification in mixed data environments can be another direction for research.

In the second part of our study, a specific area of data stream clustering is studied, short text stream clustering. A deep review for the current methods is made and a survey classifying these methods according to their clustering approaches and also analyzing them by making comarison with different aspects is prepared. In our survey, the methods are classified into three groups. The first group uses Dirichlet Process in order to utilize the probabilities about assigning the incoming text to current clusters or generating a new cluster. The next group is similarity based and incremental methods. These methods construct a vector representation for the short documents and calculate the similarity values with the text and the current cluster. The last group is word relation network based methods. They use a word relation network including the frequency and co-occurence values to operate clustering.

After completing the survey, developing a method for short text stream clustering is studied on. A new streaming short text clustering method called T-GSC is proposed. The proposed method uses graphs for the clustering process. Additionally, it is a two

phase method which applies the clustering two times on different forms of the data, in order to improve the clustering quality. To measure the performance of our method in terms of clustering quality measures, experiments are conducted on four benchmark datasets, revealing that the proposed method, can produce better results or compatible with the best results with respect to the-state-of-the-art methods.

As a future work, the graph construction process of our method can be modified by considering different aspects such as the number of words or co-occurences of them between short text documents. The order of occurence may also be taken into account to improve the accuracy of our method. While designing T-GSC, SBERT is used for vector representations and modifications which ares specific for short text stream clustering in SBERT may be proposed as a future work. Also, improvements may be offered to increase the computational performance of our model such as developing parallel versions for some of the steps. For instance, construction of graphs and generation of network embeddings for clusters are independent of each other and they may be calculated concurrently.

REFERENCES

[1] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[2] S. Badiozamany, *Real-time data stream clustering over sliding windows*. PhD thesis, Acta Universitatis Upsaliensis, 2016.

[3] E. Maden and P. Karagoz, "Recent methods on short text stream clustering: A survey study," *Wiley Interdisciplinary Reviews: Computational Statistics*, p. e1610, 2023.

[4] E. Maden. and P. Karagoz., "Enhancements for sliding window based stream classification," in *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR,*, pp. 181–189, INSTICC, SciTePress, 2019.

[5] J. G. Cleary and L. E. Trigg, "K*: An instance-based learner using an entropic distance measure," in *Machine Learning Proceedings 1995*, pp. 108–114, Elsevier, 1995.

[6] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1st ed., October 1992.

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

[8] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[9] S. Pan, J. Wu, X. Zhu, and C. Zhang, "Graph ensemble boosting for imbalanced noisy graph stream classification," *IEEE transactions on cybernetics*, vol. 45, no. 5, pp. 954–968, 2014.

[10] T. P. da Silva, G. A. Urban, P. de Abreu Lopes, and H. de Arruda Camargo, "A fuzzy variant for on-demand data stream classification," in *2017 Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 67–72, IEEE, 2017.

[11] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On demand classification of data streams," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 503–508, 2004.

[12] R. Yang, S. Xu, and L. Feng, "An ensemble extreme learning machine for data stream classification," *Algorithms*, vol. 11, no. 7, p. 107, 2018.

[13] M. Woźniak, P. Ksieniewicz, B. Cyganek, A. Kasprzak, and K. Walkowiak, "Active learning classification of drifted streaming data," *Procedia Computer Science*, vol. 80, pp. 1724–1733, 2016.

[14] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*, pp. 443–448, SIAM, 2007.

[15] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Proceedings of the 28th annual ACM symposium on applied computing*, pp. 801–806, ACM, 2013.

[16] A. Bifet, J. Zhang, W. Fan, C. He, J. Zhang, J. Qian, G. Holmes, and B. Pfahringer, "Extremely fast decision tree mining for evolving data streams," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1733–1742, 2017.

[17] M. R. Sousa, J. Gama, and E. Brandão, "A new dynamic modeling framework for credit risk assessment," *Expert Systems with Applications*, vol. 45, pp. 341–351, 2016.

[18] D. Shi, J. Zurada, and J. Guan, "Identification of human factors in aviation incidents using a data stream approach," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[19] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: massive online analysis," *Journal of Machine Learning Research*, vol. 11, 05 2010.

[20] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Generation Computer Systems*, vol. 75, pp. 187–199, 2017.

[21] N. Kourtellis, G. D. F. Morales, A. Bifet, and A. Murdopo, "Vht: Vertical hoeffding tree," in *2016 IEEE International Conference on Big Data (Big Data)*, pp. 915–922, IEEE, 2016.

[22] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. USA: Elsevier, 2011.

[23] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," tech. rep., Stanford, 2006.

[24] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive clustering for dynamic iot data streams," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 64–74, 2016.

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

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

[27] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, 2007.

[28] Y. Lu, Y. Sun, G. Xu, and G. Liu, "A grid-based clustering algorithm for high-dimensional data streams," in *Advanced Data Mining and Applications: First International Conference, ADMA 2005, Wuhan, China, July 22-24, 2005. Proceedings 1*, pp. 824–831, Springer, 2005.

[29] J. Gama, P. P. Rodrigues, and L. Lopes, "Clustering distributed sensor data streams using local processing and reduced communication," *Intelligent Data Analysis*, vol. 15, no. 1, pp. 3–28, 2011.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[31] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the american statistical association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[32] S. Yang, G. Huang, X. Zhou, V. Mak, J. Yearwood, *et al.*, "Ewnstream+: Effective and real-time clustering of short text streams using evolutionary word relation network.," *Int. J. Inf. Technol. Decis. Mak.*, vol. 20, no. 1, pp. 341–370, 2021.

[33] J. Chen, Z. Gong, and W. Liu, "A dirichlet process biterm-based mixture model for short text stream clustering," *Applied Intelligence*, vol. 50, no. 5, pp. 1609–1619, 2020.

[34] J. Kumar, J. Shao, S. Uddin, and W. Ali, "An online semantic-enhanced dirichlet model for short text stream clustering," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 766–776, 2020.

[35] J. Qiang, W. Xu, Y. Li, Y. Yuan, and Y. Zhu, "Lifelong learning augmented short text stream clustering method," *IEEE Access*, vol. 9, pp. 70493–70501, 2021.

[36] J. Yin, D. Chao, Z. Liu, W. Zhang, X. Yu, and J. Wang, "Model-based clustering of short text streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2634–2642, 2018.

[37] S. Yang, G. Huang, X. Zhou, and Y. Xiang, "Dynamic clustering of stream short documents using evolutionary word relation network," in *International Conference on Data Service*, pp. 418–428, Springer, 2019.

[38] R. Molina, W. Hasperué, and A. V. Monte, "D3cas: Distributed clustering algorithm applied to short-text stream processing," in *Argentine Congress of Computer Science*, pp. 211–220, Springer, 2018.

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

[40] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[41] A. Kalogeratos, P. Zagorisios, and A. Likas, "Improving text stream clustering using term burstiness and co-burstiness," in *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*, pp. 1–9, 2016.

[42] J. Kleinberg, "Bursty and hierarchical structure in streams, data mining and knowledge discovery," in *elected Papers from the 8th ACM SIGKDD Int. Conf. on Knowledge I Discovery and Data Mining? Part*, vol. 7, pp. 372–397, 2002.

[43] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[44] M. R. H. Rakib, N. Zeh, and E. Milios, "Short text stream clustering via frequent word pairs and reassignment of outliers to clusters," in *Proceedings of the ACM Symposium on Document Engineering 2020*, pp. 1–4, 2020.

[45] J. Kumar, R. Kumar, A. U. Haq, and S. Shafiq, "A non-parametric multi-lingual clustering model for temporal short text," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 58–61, IEEE, 2020.

[46] S. Liang, E. Yilmaz, and E. Kanoulas, "Dynamic clustering of streaming short documents," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 995–1004, 2016.

[47] W. Xu, Y. Li, and J. Qiang, "Dynamic clustering for short text stream based on dirichlet process," *Applied Intelligence*, pp. 1–12, 2021.

[48] J. Kumar, S. U. Din, Q. Yang, R. Kumar, and J. Shao, "An online semantic-enhanced graphical model for evolving short text stream clustering," *IEEE Transactions on Cybernetics*, 2021.

[49] O. Ozdikis, P. Karagoz, and H. Oğuztüzün, "Incremental clustering with vector expansion for online event detection in microblogs," *Social Network Analysis and Mining*, vol. 7, no. 1, pp. 1–17, 2017.

[50] A. Najafi, A. Gholipour-Shilabin, R. Dehkharghani, A. Mohammadpur-Fard, and M. Asgari-Chenaghlu, "Comstreamclust: a communicative multi-agent approach to text clustering in streaming data," 2021.

[51] M. Rashadul Hasan Rakib and M. Asaduzzaman, "Fast clustering of short text streams using efficient cluster indexing and dynamic similarity thresholds," *arXiv e-prints*, pp. arXiv–2101, 2021.

[52] K. Liu, J. He, and Y. Chen, "A topic-enhanced dirichlet model for short text stream clustering," *Neural Computing and Applications*, vol. 36, no. 14, pp. 8125–8140, 2024.

[53] W. Zhang, C. Dong, J. Yin, and J. Wang, "Attentive representation learning with adversarial training for short text clustering," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[54] M. Ilic, P. Spalevic, and M. Veinovic, "Inverted index search in data mining," in *2014 22nd Telecommunications Forum Telfor (TELFOR)*, pp. 943–946, IEEE, 2014.

[55] M. Soleymanian, H. Mashayekhi, and M. Rahimi, "An incremental clustering algorithm based on semantic concepts," *Knowledge and Information Systems*, pp. 1–33, 2024.

[56] S. Li, J. Zhu, and C. Miao, "Psdvec: A toolbox for incremental and scalable word embedding," *Neurocomputing*, vol. 237, pp. 405–409, 2017.

[57] J. Singh, D. Pandey, and A. K. Singh, "Event detection from real-time twitter streaming data using community detection algorithm," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 23437–23464, 2024.

[58] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[59] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[60] E. Maden and P. Karagoz, "A hybrid sliding window based method for stream classification," in *Knowledge Discovery, Knowledge Engineering and Knowledge Management: 11th International Joint Conference, IC3K 2019, Vienna, Austria, September 17-19, 2019, Revised Selected Papers 11*, pp. 94–107, Springer, 2020.

[61] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377–382, ACM, 2001.

[62] J. Yin and J. Wang, "A dirichlet multinomial mixture model-based approach for short text clustering," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 233–242, 2014.

[63] A. Saha and B. Ganesan, "Short text clustering in continuous time using stacked dirichlet-hawkes process with inverse cluster frequency prior," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Maden, Engin
Nationality: Turkish (TC)

EDUCATION

Degree	Institution	Year of Graduation
M.S.	METU Computer Engineering	2010
B.S.	Hacettepe University Computer Engineering	2006

PUBLICATIONS

1. Maden, E., & Karagoz, P. (2023). Recent methods on short text stream clustering: A survey study. Wiley Interdisciplinary Reviews: Computational Statistics, 15(6), e1610.

2. Maden, E., & Karagoz, P. (2020). A Hybrid Sliding Window Based Method for Stream Classification. In Knowledge Discovery, Knowledge Engineering and Knowledge Management: 11th International Joint Conference, IC3K 2019, Vienna, Austria, September 17-19, 2019, Revised Selected Papers 11 (pp. 94-107). Springer International Publishing.

3. Maden, E., & Karagoz, P. (2019, September). Enhancements for Sliding Window based Stream Classification. In KDIR (pp. 181-189).

4. Senkul, P., Onder, N., Onder, S., Maden, E., & Nyew, H. M. (2012). Discovering patterns for architecture simulation by using sequence mining. In Pattern

Discovery Using Sequence Data Mining: Applications and Studies (pp. 212-236). IGI Global.