12-2017

A framework for clustering and adaptive topic tracking on evolving text and social media data streams.

Gopi Chand Nutakki
University of Louisville

A FRAMEWORK FOR CLUSTERING AND ADAPTIVE TOPIC TRACKING ON EVOLVING TEXT AND SOCIAL MEDIA DATA STREAMS

By

Gopi Chand Nutakki
M.Sc., Computer Science,
Western Kentucky University, Bowling Green, KY

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Computer Science and Engineering

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

December 2017

A FRAMEWORK FOR CLUSTERING AND ADAPTIVE TOPIC TRACKING ON EVOLVING TEXT AND SOCIAL MEDIA DATA STREAMS

By

Gopi Chand Nutakki
M.Sc., Computer Science,
Western Kentucky University, Bowling Green, KY

A Dissertation Approved On

December 1, 2017

Date

by the following Dissertation Committee:

Olfa Nasraoui, Ph.D., Dissertation Director

Nihat Altiparmak, Ph.D.

Hichem Frigui, Ph.D.

Mary P. Sheridan, Ph.D.

Roman Yampolskiy, Ph.D.

ACKNOWLEDGEMENTS

Dr. Olfa Nasraoui has been more than just a mentor to me. Her never ending support immensely inspires me and helped me complete this dissertation.

I would like to thank my dissertation committee for their support and guidance.

I would like to convey my gratitude to my lab-mates Behnoush Abdollahi and Wenlong Sun for their help and support. I would also like to convey my gratitude to my work colleagues and mentors Huda Melky, Joshua Hayes, Cindy Smith, and Chantel Wilson for their support. I would also like to say thank you to my family and friends.

This work was conducted in part using the resources of the University of Louisville's research computing group and the Cardinal Research Cluster. I would like to thank James Harrison Simrall and the CRC staff for their help with the computing resources.

Last but not least, this research was partially supported by US National Science Foundation Data Intensive Computation Grant IIS-0916489.

ABSTRACT

A FRAMEWORK FOR CLUSTERING AND ADAPTIVE TOPIC TRACKING ON EVOLVING TEXT AND SOCIAL MEDIA DATA STREAMS

Gopi Chand Nutakki

December 1, 2017

Recent advances and widespread usage of online web services and social media platforms, coupled with ubiquitous low cost devices, mobile technologies, and increasing capacity of lower cost storage, has led to a proliferation of Big data, ranging from, news, e-commerce clickstreams, and online business transactions to continuous event logs and social media expressions. These large amounts of online data, often referred to as data streams, because they get generated at extremely high throughputs or velocity, can make conventional and classical data analytics methodologies obsolete. For these reasons, the issues of management and analysis of data streams have been researched extensively in recent years. The special case of social media Big Data brings additional challenges, particularly because of the unstructured nature of the data, specifically free text. One classical approach to mine text data has been Topic Modeling. Topic Models are statistical models that can be used for discovering the abstract "topics" that may occur in a corpus of documents. Topic models have emerged as a powerful technique in machine learning and data science, providing a great balance between simplicity and complexity. They also provide sophisticated insight without the need for real natural language understanding. However they have been designed to cope not with the type of text data abundant on social media platforms, but rather traditional medium size corpora consisting of longer documents, adhering to a

specific language and typically spanning a stable set of topics. Unlike traditional document corpora, social media messages tend to be very short, sparse, noisy, and do not adhere to a standard vocabulary, linguistic patterns, or stable topic distributions. They are also generated at high velocity that impose high demands on topic modeling. Finally, their evolving or dynamic nature makes any set of results from topic modeling quickly become stale in the face of changes in the textual content and topics discussed within social media streams.

In this dissertation, we propose an integrated topic modeling framework built on top of an existing stream-clustering framework called Stream-Dashboard, which can extract, isolate, and track topics over any given time period. The proposed approach to topic modeling is different from a generic Topic Modeling approach because it works in a compartmentalized fashion, where the input document stream is split into distinct compartments, and topic modeling is applied on each compartment separately. In this new framework, Stream Dashboard first clusters the data stream points into homogeneous groups. Then data from each group is ushered to a topic modeling algorithm which extracts finer topics from the group. The proposed framework tracks the evolution of the clusters over time to detect milestones corresponding to changes in topic evolution, and to trigger an adaptation of the learned groups and topics at each milestone.

Furthermore, we propose extensions to existing topic modeling and stream clustering methods, including: an adaptive query reformulation approach to help focus or adapt topic discovery with time and an adaptive stream clustering algorithm incorporating the automated estimation of dynamic, cluster-specific temporal scales for adaptive forgetting to help facilitate clustering in a fast evolving data stream.

Our experimental results show that the proposed adaptive forgetting clustering algorithm can mine better quality clusters; that our proposed compartmentalized framework is able to mine topics of better quality compared to competitive baselines; and that the proposed framework can automatically adapt to focus on changing topics using the proposed query reformulation strategy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND MOTIVATION

Recent advances and widespread usage of online services and social media platforms, coupled with ubiquitous low cost devices, mobile technologies, and increasing capacity of lower cost storage, has led to a proliferation of big data, ranging from e-commerce and online business transactions to news and entertainment website dynamic content and click-streams, continuous event logs, and social media expressions. This data can be a potent source for mining knowledge about human activity online and how it relates to their social environment. Applications of this knowledge discovery range from real-time disaster management to social sensing for monitoring news and opinions for diverse applications such as humanitarian aid, marketing, and political campaign management. These large amounts of online data, often referred to as data streams, because they get generated at extremely high throughputs or velocity, can make conventional and classical data analytic methodologies obsolete. For these reasons, the issues of management and analysis of data streams have been researched extensively in recent years. A very important component in managing massive amounts of information is the integration of tools which can process data rapidly and efficiently. Processing large volumes of information has always been a challenging task and in many instances, the processing must be done online and the results presented in real-time [6–8]. For the online scenario, data arrive as streams which are fast, continuous, mutable, ordered, and potentially unbounded [9].

The special case of social media Big Data brings additional challenges, particularly because of the unstructured nature of the data, specifically free text. Natural language text has a rich structure where individual words are composed of morphemes, words are pieced together to reflect syntactic structure, and all pieces collaborate to express meaning

[10]. Inferring these three types of structure from text morphology, syntax, and semantics has occupied much of computational linguistics and natural language processing research through the years [11]. One classical approach to mine text data has been topic todeling [3, 12]. Topic Models are statistical models that can be used for discovering the abstract *topics* that may occur in a corpus of documents. Topic modeling is a form of machine learning that is based on the assumption that a document is composed of multiple topics and that a collection of documents represents a collection of topics [12]. A topic captures the information of what the document is speaking about and influences many aspects of the document, including word selection, sentence structure, and tone of writing etc [13].

Latent semantic indexing (LSI) is a classical technique to store and retrieve documents that addresses challenges caused by ambiguity in natural language. LSI has roots in a probabilistic framework through the development of probabilistic latent semantic indexing (pLSI) [11, 14]. This formulation has led to several additional probabilistic topic models for documents, most notably latent Dirichlet allocation (LDA) [3]. Since 2003, probabilistic topic models have been applied to many applications in natural language processing and machine learning, and several extensions have been proposed to the LDA model [15].

Topic models and related techniques such as LSI and pLSI have emerged as powerful techniques in machine learning and data science, providing a great balance between simplicity and complexity. They also provide sophisticated insight without the need for real natural language understanding. However they have not been designed to cope with the type of text data that is abundant on social media platforms, but rather for traditional medium size corpora consisting of longer documents, adhering to a specific language and typically spanning a stable set of topics. Unlike traditional document corpora, social media messages tend to be very short, sparse, noisy, and do not adhere to a standard vocabulary, linguistic patterns, or stable topic distributions. They are also generated at high velocity that impose high demands on topic modeling; and their evolving or dynamic nature, makes any set of results from topic modeling quickly become stale in the face of changes in the textual content and topics discussed within social media streams.

Another family of machine learning techniques that has been used for unsupervised knowledge discovery in big data is clustering. Clustering is the process of grouping items in such a way that there is high intra-cluster similarity and low inter-cluster similarity. Organizing the data into clusters shows the internal structure of the data. Thus, Clustering is very useful to discover knowledge in the data [16, 17]. A data stream can roughly be thought of as a transient, continuously increasing sequence of time-stamped data [18]. Stream clustering is a technique that performs cluster analysis of data streams that is able to produce results in real time. The ability to process data in a single pass and summarize it, while using limited memory, is crucial to stream clustering. Partitioning a data stream into groups which are similar in a certain sense can help extract finer topics from each group. In order to maintain an up-to-date topic structure, it is necessary to analyze the incoming data in an online manner, tolerating not more than a constant time delay [19].

In this research, we explore and use online topic modeling techniques, coupled with an online stream clustering framework to extract and track topics, discussed on an online social media platform over a period of time.

1.1 Research Motivation and Challenges

Social media offers a platform to track and discover stories evolving over time all over the world. Extracting topics/stories along with the trends of sentiments of these topics provides a fast way to discover and follow new and interesting events. The extraction of new filtering terms may provide more leads to focus the discovery of stories evolving in different directions that the user may not be aware of, or are completely unknown to the user.

One application of our research is to use the ability to discover evolving clusters of topics within a fast moving text data stream, in order to be able to support specific tasks in a real life setting. For instance journalists and law enforcement professionals are faced with a challenging task when trying to extract information about certain events. One application of the proposed work is to develop an automated system to try to retrieve information related to a certain topic or event from social media streams. Unlike traditional Information

Retrieval, the problem is not limited to searching an existing index of documents against an input query. The differences are as follows:

- The query itself is not easy to formulate, thus it will need to be gradually constructed starting from an initially formulated query.
- There is no pre-indexed collection, rather a vast pool of social media data that has been collected with new data being added continuously.
- The collected data has a temporal and in some cases, location characteristic, and may need to be searched at different time instants and using a different location focus.
- The collected data is mainly generated by users on twitter. It reflects often spontaneous, uncensored Human conversations and free expression without editing or any kind of expectations of properness of language. Twitter streams tend to be rich in vernacular, and in many cases improper or offensive language, in addition to being extremely short (less than 140 characters).
- The desired retrieval results, unlike traditional IR, should be in the form of an evolving story. Story telling can be supported by an automated extraction of topics. Detecting milestones of change can be used to segment a story into pieces. When needed, stories can become complex searchable objects, and multiple stories can be combined into a bigger story or chain of events and vice versa.

1.2 Overarching Goals

The overarching goal of this dissertation is to develop an integrated framework for mining topics from massive text or social media data streams in real time. This goal requires solving the following problems: **(1)** Learning *evolving* topics from social media data streams; **(2)** Handling the diversity and open ended nature of discussion topics on social media data streams; **(3)** Adapting topic mining to automatically focus on emerging topics of discussion.

1.3 Outline of the Proposed Framework

We propose an integrated framework that is built based on the following components that together, address the problems that were stated above: **(1)** Handling content heterogeneity using a stream clustering algorithm with adaptive forgetting / dynamic temporal scales; **(2)** Topic Modeling for characterizing the discovered clusters in the case of text or social media data streams; **(3)** Handling temporal heterogeneity using automated milestone detection; **(4)** Adapting the initial text stream filtering strategy using query reformulation. Below, we briefly review some background and challenges for the building blocks of these components.

1.3.1 Topic Models

Topic models [3, 20] can discover the latent semantics embedded in documents, and the semantic information can be much more useful to identify document groups than raw term features. Generally, in document clustering approaches, documents are represented with a bag-of-words (BOW) model [21] which is purely based on raw terms and is insufficient to capture all semantics. Topic models are able to put words with similar semantics into the same group called *topic* where synonymous words are treated as related. Under topic models, a document corpus is projected into a topic space which reduces the noise that can affect similarity measures and the topic structure of the corpus can be identified more effectively.

1.3.2 Conquering Content Heterogeneity using Clustering

In a heterogeneous dynamic social media stream, documents usually belong to several big groups. Each group can have its own set of finer topics. Let $\zeta = \{C_{1,n}, C_{2,n}, C_{3,n}, \cdots, C_{k<k_{max},n}\}$ be the set of clusters after n data points were encountered. Let $S_n = \{S_{1,n}, S_{2,n}, \cdots, S_{k<k_{max},n}\}$ be a set of data points that were assigned to the respective clusters. Clustering can help identify the latent groups in a document collection and subsequently local topics specific to each group can be extracted using topic modeling. These fine-grained topics can facilitate

storytelling. Group-specific local topics can be used to summarize stories. Global topics can be used to remove background words and describe the general contents of the whole collection. When considering social media data like tweets, extracting topics from such an unstructured, small length, continuous new vocabulary, is a very challenging problem.

Unfortunately, Topic modeling is an intensive and costly process, both in terms of iterations and memory cost. When the data is a fast moving stream, topic modeling can find it hard to cope, despite some work on online topic modeling and distributed topic modeling. On the other hand, stream clustering methods, such as RINO-Streams (**R**obust clustering of data streams using **IN**cremental **O**ptimization) [22] and the STREAM-DASHBOARD [23] framework offer a fast alternative for online unsupervised learning from massive data streams. Hence, the idea of coupling topic modeling with the scalable stream clustering of RINO-Streams and the online cluster tracking, validation and evolution summarization of STREAM-DASHBOARD, seems to be a promising direction to reach the overarching goal of massive stream topic modeling and tracking. Stream clustering can provide a divide and conquer strategy to accelerate and improve the convergence properties of topic modeling even within the extremely challenging fast moving stream environment.

Document stream clustering and online topic modeling can mutually benefit each other. Document clustering can be combined with topic models to extract local topics specific to each cluster and global topics shared by all clusters. Thus, a unified framework can provide a platform to extract local and global stories. The unified framework tightly couples two components: the Stream-Dashboard framework [23] is used for discovering latent clusters in document streams and a Topic Modeling component [4] is used for mining finer topics specific to each cluster.

1.3.3 Conquering Temporal Heterogeneity

The quality of a learned topic model is sensitive to the choice of the window size, filter words etc; of a data stream. Since Stream Dashboard [23] can detect milestones (changes in topic trends) from the data streams automatically, it promises to be useful to divide the data

into homogeneous subsets that are later fed to the topic modeling component, which can in-turn extract finer topics. Stream-Dashboard can detect several trending clusters as well as their behavioral changes over time, including milestones which are time stamps of significant change. Topic Modeling with Stream-Dashboard can therefore extract trending clusters that are presented as a story in terms of a set of events, corresponding to cluster milestones. Whenever Stream-Dashboard encounters a milestone, topic modeling is triggered to extract topics from a particular cluster.

1.3.4 Query Reformulation

From these topics, seed words can be extracted to help guide topic modeling in the upcoming time intervals. Seed words can be generated using Query Reformulation and then used to filter the stream, thus providing more relevant and focused data to extract stories.

1.4 Research Contributions

Starting with an initial cross-section of a data stream, filtered using a small set of keywords, the proposed framework consists of multiple stages to (1) cluster the stream data, (2) extract the topics from each cluster, (3) extracts seed words from each cluster, and furthermore (4) track the topic evolution over time, while performing all the mining stages. The initial stage involves using the Stream-Dashboard framework which clusters the data stream points (i.e. documents). Stream-Dashboard divides the data points into a set of homogeneous clusters. This acts as a data stream filtering to help extract finer topics from a similar set of documents. Stream-Dashboard also provides a mechanism to identify temporal milestones which indicate a new trend of topics in the data stream.

In order to cope with the challenges inherent in clustering evolving data streams, **we propose a new stream clustering algorithm called AFTER-Streams (*A*daptive *F*orgetting *T*ime-Decay *E*volving *R*obust Stream Clustering Algorithm), endowed with adaptive forgetting/temporal decay to be able to cope with clusters with different temporal dynamics (lifetimes or horizons)** which in turn necessitate

different forgetting rates. This adaptive forgetting via dynamic cluster-specific temporal scales in turn allows us to discover better quality clusters. Our new clustering algorithm is evaluated on a variety of data streams, including: synthetic data (to control for stream properties), text documents from the TREC collections, social media posts, and network intrusion event log data.

The second stage involves invoking the topic modeling (TM) component. When Stream-Dashboard detects milestones, the topic modeling is performed on the clusters where the milestone occurred. Stream-Dashboard dynamically creates, merges, dissolves the clusters based on the detected milestones. The topic modeling component also performs similar operations on the topic models, i.e. to extract topics from the documents of a given cluster, to merge topic models from merged clusters, and to dissolve the topic models after their clusters have decayed enough.

The combination of the proposed stream clustering with adaptive forgetting and topic modeling within each cluster results in **a new proposed compartmentalized approach that can mine topics from a diverse data stream by automatically partitioning the stream in both content and time**. **Content Partitioning** partitions the data into different content clusters by automatically estimating *content* boundaries between the clusters based on text. In a similar fashion, the proposed **Temporal Partitioning** compartmentalizes the data in *time* by automatically estimating optimal time boundaries of change within each cluster. Temporal Partitioning also includes an automated adaptation to the temporal relevance horizon for each cluster and hence adapting to the forgetting of the topic clusters as they become no longer the subject of discussions on the social media stream. In order to automate the topic mining, we also present an approach to optimize the choice of hyper-parameters used for topic modeling.

Finally, **whenever a milestone is detected, several potential future seed words are extracted from the discovered topics and are relayed back as a reformulated query to the initial data stream filters to help guide or focus the future discovery toward emerging or more specific topics**. Our compartmentalized

topic mining with query reformulation is evaluated on several social media data streams from Twitter.

To summarize our contributions, we list our main goals, proposed methods, and evaluation plan, in Table 1.1. Our research questions, which guide the evaluation of our proposed methods, can be stated as follows:

Research Question 1 (RQ1): Does the proposed clustering with adaptive forgetting/temporal decay result in better quality clusters compared to baseline methods? This in turn can be refined into the following specific research questions:

- **RQ1.1:** Which parameters of the data stream show a significant effect on AFTER-Streams performance?
- **RQ1.2:** Which parameters of the AFTER-Streams algorithm show a significant effect on its performance?
- **RQ1.3:** Does the AFTER-Streams algorithm perform better than the baseline algorithms.

Research Question 2 (RQ2): Does the proposed compartmentalized topic mining approach result in better quality extracted topics compared to baseline methods? This in turn can be refined into the following specific research questions:

- **RQ2.1:** Does auto-tuning the Online LDA hyperparameters have a significant effect on the quality of the topics?
- **RQ2.2:** Does the Compartmentalized Framework perform better than the baseline algorithm?
- **RQ2.3:** Does the Compartmentalized Framework with Query Reformulation perform better than the baseline algorithm?
- **RQ2.4:** Does the Compartmentalized Framework with Query Reformulation produce more diverse set of topics?

TABLE 1.1

Summary of Research Contributions

Framework Component	Goal	Methodology	Research Questions	Validation Metrics	Evaluation Section
Clustering evolving data streams	Adapt to different evolving cluster data arrival speeds	Dynamic Forgetting (Chapter 3)	RQ1.1, RQ1.2, RQ1.3	Internal and External cluster validation metrics, e.g. F1-Score, DB-Index etc.	Sec 3.3
Topic Mining in evolving data streams	Obtain pure topics	Compatmentalized Framework (Sec 4.2)	RQ2.1, RQ2.2	Perplexity and Coherence	Sec 4.3
	Reduce vocabulary size				
	Detect milestones				
	Adapt stream data filters to evolving topics.	Query Reformulation (Sec 4.2.7)	RQ2.3, RQ2.4	New filter terms and unseen topics, Topic Agglomeration (qualitative evaluation).	Sec 4.3.8
	Discover new topics				

1.5 Organization of this Document

Chapter 2 presents the background and related work, mainly an overview of topic modeling and data clustering techniques, in particular for data streams. Chapter 3 presents our proposed extension to the RINO-Streams clustering algorithm, where an addition temporal scale is used in addition to content based scales, thus increasing the cluster quality. Chapter 4 presents our proposed Compartmentalized Online Topic Modeling Framework. All the evaluation experiments for the proposed research are presented at the end of their respective chapters. Finally, Chapter 5 concludes the dissertation.

CHAPTER 2

RELATED WORK

In this chapter, we review clustering algorithms in Section 2.1. Then we review similarity measures used in data clustering in Section 2.1.2. The Stream-Dashboard framework is reviewed in Section 2.1.10, presenting the basic ideas of Stream-Dashboard and its components, such as RINOStreams and TRACER. We then present an introduction to the topic modeling techniques, PLSI, LDA, Online LDA etc, in Section 2.3. We then present the concept of relevance feedback in Section 2.5 which lays the foundation for enriching the proposed compartmentalized online topic modeling framework with automated query reformulation.

2.1 Clustering Algorithms

The goal of data clustering is to discover the natural groupings of a set of patterns, points, or objects [24]. An operational definition of clustering can be stated as: "Given a representation of n objects, find K groups based on a measure of similarity such that the similarities between objects in the same group are high while the similarities between objects in different groups are low" [24]. An ideal cluster can be defined as a set of points that is compact and isolated.

Clustering algorithms can be broadly divided into two groups: hierarchical and partitional [25]. Hierarchical clustering algorithms recursively find nested clusters either in agglomerative mode or in divisive (top-down) mode. Partitional clustering algorithms find the clusters simultaneously as a partition of the data and do not impose a hierarchical structure. The most well-known hierarchical algorithms are single-link and complete-link; the most popular and the simplest partitional algorithm is K-means [26]. The goal of

K-means is to minimize the sum of squared errors over all K clusters.

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \|x_i - \mu_k\|^2 \tag{2.1}$$

The K-means algorithm depends on *three* user-specified settings: number of clusters K, cluster initialization, and distance metric. The most critical choice is K. While no perfect mathematical criterion exists, a number of heuristics are available for choosing K. The basic K-means algorithm has been extended in many different ways [24]. Some of these extensions deal with additional heuristics involving the minimum cluster size and merging and splitting clusters. In K-means, each data point is assigned to a single cluster. Fuzzy C-means [27] is an extension of K-means where each data point can be a member of multiple clusters with a membership value. Data reduction by replacing group examples with their centroids before clustering them was used to speed up K-means and fuzzy C-means. Bisecting K-means [17] recursively partitions the data into two clusters at each step. X-means [28] automatically finds K by optimizing a criterion such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC). These extensions introduce some additional algorithmic parameters that must be specified by the user. The Stream-Dashboard framework is discussed in Section 2.1.10. Later sections will also discuss various clustering algorithms, limitations of clustering, Relevance Feedback, and Topic Modeling with Sentiment Analysis.

2.1.1 Expectation-Maximization (EM)

Algorithm 2.1 Expectation Maximization Algorithm.

1. Select an initial set of model parameters (Θ)
2. **Expectation Step**: Find the probability that each data point belongs to each distribution
3. **Maximization Step**: Use the probabilities found in the E step to find new estimate of the model parameters (Θ) that maximize the likelihood (2.3)
4. Repeat steps 2 and 3 until the parameters' change is below a specified threshold value

The Expectation-Maximization algorithm (EM) [29], models the dataset as a mixture

of data points generated by K distributions with known form, such as Gaussian. EM tries to determine the model parameters, θ_j, using posterior probabilities to maximize the likelihood of the data under these estimated model parameters. If the j^{th} distribution has parameters θ_j, then $prob(x_i|\theta_j)$ is the probability of the i^{th} data point coming from the j^{th} distribution. Each distribution has a weight w_j which reflects the probability of being chosen to generate a data point, and the weights for all distributions sum to 1. If Θ is the set of all parameters, then the probability of the i^{th} object is given by:

$$prob(x_i|\Theta) = \sum_{j=1}^{K} w_j \, prob(x_i|\theta_j) \tag{2.2}$$

If the objects are assumed to be identically generated, then the probability of the dataset X (or the *likelihood* function) is the product of the probabilities of each data point:

$$prob(X|\Theta) = \prod_{i=1}^{N} \sum_{j=1}^{K} w_j \, prob(x_i|\theta_j) \tag{2.3}$$

where N is the number of data points. EM is listed in Algorithm 2.1. The EM algorithm provides a more general representation of data using mixture models.

2.1.2 Similarity Measures for Clustering Text Data

Web stream content is typically represented as a vector consisting of the suitably normalized frequency counts of words or terms. Each document contains only a small percentage of all the words that are ever used on the web. Document clustering is high dimensional, characterized by a highly sparse word-document matrix with positive ordinal attribute values and a significant amount of outliers. When documents are represented by a bag of words, the resulting document-word matrix typically represents data in over a thousand dimensions. Methods like spherical K-means algorithm for document clustering, graph based clustering approaches etc. attempt to avoid the pitfalls of dimensionality by transforming the problem formulation and uses a variety of similarity measures [30].

The Minkowski distance [31] is given as $L_p\left(x_a, x_b\right) = \left(\sum_{i=1}^{d} \left|x_{i,a} - x_{i,b}\right|^p\right)^{1/p}$ and is the standard metrics for geometrical problems. The Euclidean normalized similarity is

given as $s^{(E)}(x_a, x_b) = e^{-\|x_a - x_b\|_2^2}$. The Cosine similarity measure is given as $s^{(C)}(x_a, x_b) = \frac{x_a^{\uparrow} x_b}{\|x_a\|_2 \cdot \|x_b\|_2}$. Cosine similarity does not depend on the length, this allows documents with the same composition but different totals to be treated identically. In collaboration filtering, correlation is used to predict a feature from a highly similar mentor group of objects whose features are known. The normalized Pearson correlation is defined as $s^{(P)}(x_a, x_b) = \frac{1}{2}\left(\frac{(x_a - \bar{x}_a)^{\dagger}(x_b - \bar{x}_b)}{\|x_a - \bar{x}_a\|_2 \cdot \|x_b - \bar{x}_b\|_2} + 1\right)$. Jaccard similarity can be extended to continuous or discrete non-negative features using $s^{(P)}(x_a, x_b) = \frac{x_a^{\dagger} x_b}{\|x_a\|_2^2 + \|x_b\|_2^2 - x_a^{\dagger} x_b}$.

Similarity measures should be invariant to transformations natural to the problem domain. Normalization may strongly affect clustering in a positive or negative way. The features have to be chosen carefully to be on comparable scales and similarity has to reflect the underlying semantics for the given task. Euclidean similarity is translation invariant but scale variant while cosine is translation variant but scale invariant.

2.1.3 Challenges in Clustering

Data representation is one of the most important factors that influences the performance of the clustering algorithm. Clustering being a difficult problem, the definition of clusters can be very vague, and it is difficult to define an appropriate similarity measure and objective function. Automatically determining the number of clusters has always been one of the most difficult problems in data clustering [32]. Most methods for automatically determining the number of clusters cast it into the problem of model selection [33]. Usually, clustering algorithms are run with different values of K; the best value of K is then chosen based on a predefined criterion. A minimum message length (MML) criteria in conjunction with the Gaussian mixture model (GMM) can be used to estimate K [24]. Another approach using the principle of Minimum Description Length (MDL) was used for selecting the number of clusters. The other criteria for selecting the number of clusters are the Bayes Information Criterion (BIC) and Akaike Information Criterion (AIC) [34]. Gap statistics [35] is another commonly used approach for deciding the number of clusters.

2.1.4 The COBWEB Algorithm

COBWEB [36] is a conceptual clustering algorithm and can discover understandable patterns in data. It computes a hierarchical clustering model, in the form of a classification tree. Given a new point, COBWEB descends the tree along an appropriate path, updating the counts in the interior nodes along the way and looks for the best node to place the point on, using a category utility function. COBWEB is ill suited because it is not height balanced. The time complexity to add a new point to the clusters might degrade dramatically. COBWEB analyzes the resulting placements and computes whether it is a better choice than placing the point in one of the current clusters, and this way, outliers can be identified.

2.1.5 Spherical K-means

The objective of the standard *k-means* clustering is to minimize the mean-square error [37]:

$$E = \frac{1}{N} \sum_{x} \left\| x - \mu_{k(x)} \right\|^2 \tag{2.4}$$

where $k(x) = \arg\min_{k \in \{1,\cdots,K\}} \|x - \mu_k\|$is the index of the closest cluster centroid to x, N is the total number of data vectors. The underlying probability distribution for the standard k-means algorithm is Gaussian.

For high-dimensional data such as text documents and market baskets, *cosine* similarity is a superior measure to Euclidean distance. The spherical K-means algorithm aims to maximize the average cosine similarity objective given by:

$$L = \sum_{x} x^T \mu_{k(x)} \tag{2.5}$$

where $k(x) = \arg\max_k x^T \mu_k$. The main difference from standard k-means is that the re-estimated mean vectors need to be normalized to unit-length and the underlying probabilistic models are not Gaussian.

Document clustering has become an increasingly important technique for unsupervised document organization, automatic topic extraction, and fast information retrieval or filtering. If a vector space model, like Bag Of Words (BOW) is used, a text document gets mapped to a high dimensional vector with one dimension per *term*. Such vectors tend to be very sparse, and they have only non-negative entries. Also, vectors have vector properties such as length of the vector is much less important than their direction. This has led to the widespread practice of normalizing the vectors to unit length before further analysis, as well as to the use of the cosine between two vectors as a popular measure of similarity between them. Using unit-length term frequency-inverse document frequency (TF-IDF) vectors can lead to better clustering results than simply representing a document by multivariate Bernoulli or multinomial models [37].

2.1.6 Density-Based Stream Clustering

Discovering patterns hidden in a data stream imposes a great challenge . Requirements for stream clustering an evolving data streams may be listed as [38]:

- No assumption on the number of clusters.
- Discovery of clusters with arbitrary shape.
- Ability to handle outliers.

Data stream applications impose a limited memory constraint, and it becomes more difficult to provide arbitrary-shaped clustering results using conventional algorithms. Clusters with arbitrary shape are often represented by all the points in the clusters, which is generally unrealistic in stream applications. Due to the dynamic nature of evolving data streams, the role of outliers and clusters are often exchanged, and consequently new clusters often emerge, while old clusters fade out, it is more complex with noisy data. The *DenStream* algorithm [38] discovers arbitrarily shaped clusters in an evolving data stream with salient features listed as:

- The core-micro-cluster synopsis is designed to summarize the clusters with arbitrary shape in data streams. The memory is limited with the guarantee of the precision of micro-clusters.

- An outlier-buffer is introduced to separate the processing of the potential core-micro-clusters and outlier-micro-clusters.

- *DenStream* achieves consistently high clustering quality.

Cluster partitions on evolving data streams are often computed based on certain time intervals. There are three popular models: landmark window, sliding window, and damped window. In the damped window model, the weight of each data point decreases exponentially with time t via a fading function $f(t) = 2^{-\lambda \cdot t}$, where $\lambda > 0$. The higher the value of λ, the lower the importance of the historical data compared to more recent data. The overall weight of the data stream is a constant $W = v\left(\sum_{t=0}^{t=t_c} 2^{-\lambda t}\right) = \frac{v}{1-2^{-\lambda}}$, where t_c is the current time and v denotes the speed of the stream (i.e. the number of points arrived in one unit time) [38].

2.1.7 Stream Clustering

Numerous challenges exist in analyzing and updating the models that reflect the new data as it is observed in continuously arriving massive data streams [39]. One challenge is to design algorithms that can track changes in an incremental way without making growing demands on computation and storage resources. A Data stream is a continuous stream of new data points that makes computations on the past portions of the data repeatedly an impractical approach. Finding changes in clusters as new data is collected can be fruitful in scenarios like tracking the evolution of various events or topics on the internet. The basic requirements for stream clustering can be specified as [39]:

- Compactness of representation. Using a lengthy description of the clusters is not an option and would grow unbounded as new points arrive. Secondary memory representations of the current clusters in also neither possible nor desirable. A data stream

clustering algorithm must provide a representation of the clusters that is compact, grows appreciably with the number of points processed. Even a linear growth is intolerable.

- Fast, incremental processing of new data points. The tasks are generally on-line natured and there is a need for speed and incremental processing.
- The placement of new points cannot be decided by a function that requires comparison with all the points that have been processed in the past as this is too expensive with respect to time and resources.
- The function that decides the placement of new points has to exhibit good performance.
- The data stream can exhibit different trends during its lifetime, and consequently the points received at any given time may not fit well under the clustering model.
- The function that evaluates the point placement must have within its range a value for *outlier*. Dealing what to do with the outliers in an application dependent issue.

2.1.7.1 Tracking Clustering Models

As a new point i arrives, we need to check whether a new clustering model is needed. Equation 2.6 shows a bound that has been used to find the probability of a "hit", where X is the sum of independent variable X_i, p is the probability of a hit, n is the number of points, ε is the desired deviation of the estimate with respect to the real value and δ is an arbitrarily small probability. Equation 2.6 establishes that the estimate and the real value can be made to be arbitrarily close by the choice of δ [39].

$$Pr\left[|\frac{X}{n} - p| \leq \varepsilon\right] > 1 - \delta \tag{2.6}$$

Using the Chernoff inequality [40], the estimate of the success probability can be bounded, by bounding the probability that the estimate $\frac{X}{n}$ surpasses $(1+\varepsilon)p$, as shown in equation 2.7.

$$Pr\left[\frac{X}{n} > (1+\varepsilon)p\right] \leq e^{(-pne^2/3)} \tag{2.7}$$

Equation 2.6 will hold if the number of successful attempts, s to cluster points, is bounded by Equation 2.8 while n is bounded by Equation 2.9.

$$s > \frac{3(1+\varepsilon)}{\varepsilon^2} ln(\frac{2}{\delta}) \tag{2.8}$$

$$n \leq \frac{3(1+\varepsilon)}{(1-\varepsilon)\varepsilon^2 p} ln(\frac{2}{\delta}) \tag{2.9}$$

Equations 2.8 and 2.9 are very important to decide whether the current clustering model is valid under the new data that is being received. If after processing n points the model is able to successfully cluster atleast s of them, the clustering model is still valid; otherwise a new model needs to be developed. Developing a new model is application dependent and two actions are possible:

- Re-cluster the entire set of points seen so far, including the last n points that prompted the decision to re-cluster.
- Discard the old clusters and produce new clusters by considering only the previous n processed points. These bounds were used to effectively track clusters in data streams. Re-evaluation of the number of clusters is also needed.

2.1.7.2 Limitations of Stream Clustering Algorithms

BIRCH can only deal with metric attributes, it treats all the data points in the same way without considering the time when the data points arrived. COBWEB is targeted for handling discrete attributes and the category utility measurement used is very expensive to compute. All instances ever encountered are retained as terminal nodes in the hierarchy. As the dimensionality grows, the memory demands of FC can grow beyond the available memory. Speedup heuristics such as Pruning strategies can be used for better results, also sampling schemes. The computational bottleneck is the normalization. The technique uses a

basic distance metric to calculate the nearest micro-cluster for merging. Heavily dependent on the input parameters. Developments should be made in the discovery of clusters with arbitrary shape at multiple levels of granularity, dynamic adaption of the parameters in data streams, and investigation of the framework for outlier detection and density-based clustering in other stream models, in particular, in a sliding window model.

2.1.8 Robust Statistics

Nasraoui [41] presented a brief overview of Robst Statistics. Classical statistics guarantee optimality in their estimates given that the model assumptions are correct [42], however these assumptions are rarely met in practice. Rather, they are used only for mathematical convenience [43]. For example, if we assume that a sample of data follows a normal distribution, then the optimal estimate of the expected value is the mean of the data points. However, the presence of outliers, which is common in real data, can have an extreme influence on the mean value. Robust statistics [1, 42] seek to obtain a robust estimation of the parameters of a parametric model while not being too affected by outliers or small deviations from the assumed model.

Most existing clustering algorithms optimize a variant of Least Square (LS) cost, which estimates the parameters of a distribution by minimizing the squared residuals, i.e. its objective function is given by

$$min \sum_{j=1}^{N} r_j^2 \tag{2.10}$$

where $r_j = x_j - \theta$ is the residual between the j^{th} data point x_j and its assumed model θ, and N is the number of data points. LS is not robust since extreme outliers with arbitrarily large residuals can have a large influence on the resulting estimate [43].

2.1.8.1 M-Estimators and W-Estimators

An M-estimator attempts to limit the influence of outliers by replacing the square of residuals with a less rapidly increasing loss function.

The M-estimator $\Theta(x_1, x_2, ..., x_N)$ estimates the parameter θ by minimizing the loss function ρ as follows

$$\min_{\theta} \left\{ J = \sum_{j=1}^{N} \rho(x_j; \theta) \right\}. \tag{2.11}$$

The optimal estimate of the parameter is found by setting the derivative of the loss function to zero as follows

$$\frac{\partial J}{\partial \theta} = \sum_{j=1}^{N} \frac{\partial \rho(x_j; \theta)}{\partial \theta} = \sum_{j=1}^{N} \psi(x_j; \theta)\,. \tag{2.12}$$

When the M-estimator is shift equivariant, i.e. $\Theta(x_1+c, ..., x_N+c) = \Theta(x_1, ..., x_N)+c$ for any constant c, the loss function ρ and its derivative ψ can be written in terms of the residuals $r = x-\theta$. Moreover, a scale estimate S is used to obtain a scaled residual $r = \frac{x-\theta}{S}$. Hence, the objective function can be written as

$$\min_{\theta} \left\{ J = \sum_{j=1}^{N} \rho(\frac{x_j - \theta}{S}) \right\}. \tag{2.13}$$

W-estimators are an alternative to M-estimators, obtained by introducing a robust weight function $w(x)$ that represents the importance of each data sample x in estimating the parameter θ. Its relation to the M-estimator is given by

$$\psi(r) = w(r)r \tag{2.14}$$

Using (2.13) and (2.14), the optimal estimate of the parameter θ is found by solving

$$\sum_{j=1}^{N} w(\frac{x_j - \theta}{S})\frac{x_j - \theta}{S} = 0 \tag{2.15}$$

The ρ, ψ and w functions for some familiar M-estimators and W-estimators [1,42,44] are listed in Table 2.1. M-estimators and W-estimators rely on an estimate of scale and a constant tuning c. Most estimators use a multiple of the Median of Absolute Deviations

TABLE 2.1: Common M-estimators and W-estimators [1]

Type	$\rho(r)$	$\psi(r)$	$w(r)$	Range of r	Scale
L_2(mean)	$\frac{1}{2}r^2$	r	1	$\mathbb{R}$	none
L_1(median)	$\lvert r\rvert$	$sign(r)$	$\frac{sign(r)}{r}$	$\mathbb{R}$	none
Huber	$\frac{1}{2}r^2$ $k\lvert r\rvert - \frac{1}{2}k^2$	r $k\,sign(r)$	1 $\frac{k\,sign(r)}{r}$	$\lvert r\rvert \leq k$ $\lvert r\rvert > k$	MAD
Cauchy	$\frac{c^2}{2}log\left[1+(\frac{r}{c})^2\right]$	$\frac{r}{1+(\frac{r}{c})^2}$	$\frac{1}{1+(\frac{r}{c})^2}$	$\mathbb{R}$	MAD
Tukey	$\frac{1}{6}\left[1-(1-r^2)^3\right]$ $\frac{1}{6}$	$r(1-r^2)^2$ 0	$(1-r^2)^2$ 0	$\lvert r\rvert \leq 1$ $\lvert r\rvert > 1$	$c \times MAD$
Andrews	$\frac{1}{\pi^2}\left[1-cos(\pi r)\right]$ $\frac{2}{\pi^2}$	$\frac{1}{\pi}sin(\pi r)$ 0	$\frac{1}{r\pi}sin(\pi r)$ 0	$\lvert r\rvert \leq 1$ $\lvert r\rvert > 1$	$c \times MAD$
Welsch	$\frac{c^2}{2}\left[1-exp(-(\frac{r}{c})^2)\right]$	$r\,exp(-(\frac{r}{c})^2)$	$exp(-(\frac{r}{c})^2)$	$\mathbb{R}$	MAD

(MAD) as a scale estimate, which assumes that the noise contamination rate is 50%. MAD is defined as follows:

$$MAD(x_i) = med_i\left\{|x_i - med_j(x_j)\right\} \tag{2.16}$$

2.1.9 Tracking Noisy Evolving Data Streams

Data streams are massive datasets that arrive with a throughput that is very high, for that reason, the data can only be analyzed sequentially with a single pass. Existing approaches use processing the data points in an incremental manner [43], or by processing the data points in small batches. For stream mining, the optimal solution is based on minimizing the Sum of Squared Distances (SSQ) [45, 46], which is the same as the one used in *k-means* [43]. TECNO-STREAMS [47] incorporates temporal weights that allow gradual forgetting of older portions of the data stream, and focuses on the newer data. CluStream [45] performs Micro-Clustering [48] with the new concept of pyramidal time-frames. CluStream divides the clustering process into an online process that periodically stores summary statistics, and an offline process that uses only summary statistics [43].

Micro-clusters are an extension of BIRCH's Cluster Feature (CF) with temporal statistics, and the incremental updating of the CF is similar to BIRCH [43]. BIRCH, in turn, solves a LS criterion because the first order statistics are nothing more than the mean centroid values [43, 49]. TRAC-STREAMS [43] tracks evolving and noisy data streams by estimating clusters based on density, while taking into account the possibility of the presence of an unknown amount of outliers, the emergence of new patterns, and the forgetting of old patterns. TRAC-STREAMS is used for mining noisy and evolving data streams that is based on a fast iterative optimization approach amounting to robust statistical estimation, and is free of assumptions about the noise contamination rate or scale value.

2.1.10 Stream-Dashboard

Stream-Dashboard [23] is a framework that can be used to *mine*, *track,* and *validate* evolving data stream clusters simultaneously. Stream-Dashboard consists of three main components: an *online clustering* component, a *tracking and validation* component and a *configuration adaptation* component. The online clustering component incrementally maintains a clustering model of the data stream. The clustering model is represented as a set of properties or metrics for each of the clusters, such as the centroids and scales. The clustering model can be used as an input to a higher level application like Topic Modeling. The tracking and validation component monitors the characteristics of the clustering model by building and maintaining *regression models*.

2.1.10.1 The RINO-Streams Algorithm

RINO-Streams (**R**obust clustering of data streams using **IN**cremental **O**ptimization) [22], an incremental clustering algorithm inspired by TRAC-Streams [43]. Both algorithms extract evolving clusters from a massive data stream in a single pass, with detection of and resistance to the presence of outliers. The algorithms incrementally updates the clustering model using an estimation of centroids and scales, rooted in robust statistics [50]. Moreover, they detect outliers and merge clusters using a robust distribution-independent statistical

Chebyshev test [51], which ensures robustness to outliers and cluster compactness.

A data stream X consists of a set of data points that are indexed based on the order of their arrival, and presented as: $x_1, x_2...., x_N$ where N is the size of the data stream. Each cluster i at time n (i.e. after receiving n points) is defined using its centroid $c_{i,n}$, its scale $\sigma^2_{i,n}$, its soft cardinality (sum of weights of data points) $W_{i,n}$ and its age a_i. The centroid represents the location of the cluster center with respect to other clusters at any time, while the scale represents the influence zone around the centroid [43]. This scale value is determined using a weighted function with temporal aspect, such that it decreases with distance from the cluster centroid as well as with the time at which the data was presented to the cluster. Hence, newer data points would have more effect on the model than older ones, which allows capturing the evolution of clusters over time. The soft cardinality is the sum of the robust weights of each data point belonging to the cluster, and is an indicator of the quality of the cluster; a high cardinality means that the cluster represents a large portion of the data stream. The age is the difference between the current time/iteration or step (n) and the time when the cluster was first created, and is used to provide the cluster with a grace period a_{mature} before testing its quality based on the minimum density threshold δ_{min}, which prevents deleting clusters while in its infancy.

RINO-Streams would update the cluster parameters with the arrival of a new point incrementally, and it would keep, as a summary of the data stream, only the centroid ($c_{i,n}$), scale ($\sigma^2_{i,n}$), the sum of weights ($W_{i,n} = \sum_{j=1}^{n} w_{ij,n}$) and the age a_i for each cluster. Moreover, a test is added to eliminate bad clusters whose density is less than a threshold (δ_{min}) and are mature enough (i.e. $a_i > a_{mature}$).

The input parameters to RINO-Streams include the maximum number of clusters K_{max} which is a higher bound on the allowed number of clusters and is needed to control the size of memory used to store the clusters, the initial scale σ_0 which is assigned to the newly created cluster, the density threshold δ_{min} which is used to ensure that only good clusters with high density are kept, the maturity age a_{mature} which provides the newly created clusters with a grace period before testing its density quality, the forgetting factor

$e^{\frac{-1}{\tau}}$ which controls the decay of the data point weights over time and the Chebyshev bound constant t.

Cluster Parameter Updates in RINO-Streams In a dynamic environment, the data model is updated with one data point at a time. The cluster centroid and scale, density, and other measures are updated with each new point. A cluster is characterized by its location or center, its scale, or its age etc. The set of clusters and their characteristic parameters define a synopsis representation of the data stream, since the currency of the stream is taken into account to define the influence zone around each cluster, the synopsis will also reflect a more current summary of the data stream. The model will evolve with the arrival of new topics in the data stream. Each cluster defines an influence zone over the data space. Data that is more current will have higher influence compared to data that is less current. The influence zone is defined in terms of a weight function that decreases not only with distance from the data to the cluster prototype, but also with the time since the data has been presented to the cluster model. It is convenient to think of time as an additional dimension to allow the presence of evolving clusters [43].

For the i^{th}cluster, C_i and j^{th}data point, at the moment when the total size of the data stream accumulated to J inputs: $x_1, x_2, \cdots, x_j, \cdots, x_J$, as:

$$w_{ij,J} = w_{i,J}\left(d_{ij}^2\right) = e^{-\left(\frac{d_{ij}^2}{2\sigma_{i,J}^2} + \frac{(J-j)}{\tau}\right)} \tag{2.17}$$

where τ is an application-dependent parameter that controls the dime decay rate of the contribution from old data points. The size of an influence zone around a cluster prototype is defined and data samples falling far from this zone are considered outliers [43]. The old weights would experience a decay given as [43]:

$$w_{ij,J} = e^{\frac{-1}{\tau}} w_{ij,(J-1)} \tag{2.18}$$

The optimal dense cluster locations and scale values are searched by optimizing the criterion [43]:

$$\min_{c_{i,J},\sigma_{i,J}} \left\{ T_{iJ} = \sum_{j=1} w_{ij,J} \frac{d_{ij}^2}{\sigma_{i,J}^2} - \alpha \sum_{j-1}^{J} \right\}, = i, \cdots, C \tag{2.19}$$

where $\alpha = 1$, weight $w_{ij,J}$ can also be considered as the degree of membership of data point $x_j$in the *inlier* set or the set of good points.

Optimal incremental center update Given the previous centers resulting from the past *(J-1)* data points, $c_{i,J-1}$, the new centroids that optimize the equation 2.19 after the J^{th}data point is given as [43]:

$$c_{i,J} = \frac{e^{-\frac{1}{\tau}} c_{i,J-1} W_{i,J-1} + w_{iJ,J} X_J}{\left(e^{-\frac{1}{\tau}} W_{i,J-1} + w_{iJ,J} \right)} \tag{2.20}$$

where $W_{i,J-1} = \sum_{j=1}^{J-1} w_{ij,(J-1)} = W_{i,J-2} + w_{i(J-1),(J-1)}$is the sum of the contributions from previous data points, $x_1, x_2, \cdots, x_j, \cdots, x_{J-1}$

Optimal incremental scale update Given the previous scales resulting from the past *(J-1)* data points, $\sigma_{i,J-1}^2$the new scales that optimize the equation 2.19 after the J^{th} data point is given by [43]:

$$\begin{aligned} \sigma_{i,J}^2 &= \frac{(2+\alpha) e^{-\frac{1}{\tau}} \sigma_{i,J-1}^2 W D_{i,J-1} + w_{ij,J} d_{iJ}^4}{(2+\alpha) \left(e^{-\frac{1}{\tau}} W D_{i,J-1} + w_{iJ,J} d_{iJ}^2 \right)} \cdot W D_{i,J-1} \\ &= \sum_{j=1}^{J-1} w_{ij,(J-1)} d_{ij}^2 = W D_{i,J-2} + w_{i(J-1),(J-1)} d_{i(J-1)}^2 \end{aligned} \tag{2.21}$$

is the sum of the contributions from previous data points $x_1, x_2, \cdots, x_j, \cdots, x_{J-1}$.

Learning New data points and Relation to Outlier Detection A potential outlier is a data point that fails the *outlyingness* test for the entire cluster model, it may either be an outlier or a new emerging pattern. An outlier will form no mature clusters in the cluster model. Chebyshev bounds [52] can be used to test whether a data point is an outlier [43]. The Chebyshev bound for a random variable X with standard deviation $\sigma$is given as:

$$Pr\left\{ |X - \mu| \geq t\sigma \right\} \leq \frac{1}{t^2} \tag{2.22}$$

For testing a data point or a new cluster for *outlyingness* with respect to cluster C_i using Chebyshev Bound with significance probability $1/t^2$, the equation 2.22 can be rewritten as,

$$Pr\left\{|X-\mu|^2 \geq t^2\sigma^2\right\} \leq \frac{1}{t^2}$$

or equivalently.

$$Pr\left\{e^{\frac{-|X-\mu|^2}{2\sigma^2}} \leq e^{-t^2/2}\right\} \leq \frac{1}{t^2}$$

So, $IF\left(w_{ij,J} < e^{\left(-t^2/2\right)}\right)\ THEN\ x_j$ is an outlier with respect to cluster C_i. For testing a compatibility of cluster $C_i$and $C_k$with scales $\sigma^2_{i,J}$and $\sigma^2_{k,J}$ using Mutual Chebyshev Bounds with significance probability $1/t^2$, given the distance between these two clusters, d^2_{ik}, if the clusters are compatible, then the equation 2.22 can be rearranged in the form:

$$Pr\left\{e^{\frac{-d^2_{ik}}{2\sigma^2_i}} \leq e^{-t^2/2}\right\} \leq \frac{1}{t^2}$$

and

$$Pr\left\{e^{\frac{-d^2_{ik}}{2\sigma^2_k}} \leq e^{-t^2/2}\right\} \leq \frac{1}{t^2}$$

and results in the test:

$$IF\left(dist(C_i,C_k) < t^2\sigma_i^2\ AND\ dist(C_i,C_k) < t^2\sigma_k^2\right)\ THEN\ Merge\ C_i\ and\ C_K$$

The new cluster center is given as:

$$c_{new,J} = \frac{c_{i,J}W_{i,J} + c_{k,J}W_{k,J}}{W_{i,J} + W_{k,J}} \tag{2.23}$$

The density of the i^{th}cluster after presenting J data points from the stream is defined as:

$$\delta_i = \frac{\sum_{j=1}^{J} w_{ij,J}}{\sigma^2_{i,J}} \tag{2.24}$$

Clusters with low density and zero cardinality are eliminated. The computational complexity of TRAC-STREAMS [43] is given as $O(N)$. This framework is useful in scenarios like network activity data, newsfeeds and Web clickstreams etc.

RINO-Streams meets all the requirements of data stream clustering algorithms, noted as:

- **Compactness of representation:** Each cluster is represented using four components: centroid c_i which is of size equal to the dimensionality of data, the scale σ_i, the sum of weights W_i and the age a_i. Hence, the total memory requirement of the clustering model is a very compact representation.
- **Fast processing of new data points & Fast handling of outliers:** Each new data point is compared against all the clusters in a linear time $O(K)$, and then is discarded. And if the data point was determined to be an outlier, then it is used to create a new cluster which is also done in linear time.
- **Integration of offline and online data:** The clustering model is very compact and can be stored in main memory. However the clustering model at different time steps can also be stored offline in secondary memory, and can be easily accessed.
- **Presenting the discovered clusters instantly:** The cluster representatives can be used directly to plot the clustering model as a set of hyper-spheres centered at the cluster centroid and with influence area equal to the scale.
- **Making no assumptions on the number of clusters:** RINO-Streams does not assume the number of clusters in advance and only requires a maximum number of clusters allowed to control the memory space used.
- **Handling evolution:** RINO-Streams handles the evolution of data by the very definition of its "dynamic" robust weight, which uses a forgetting factor to give more emphasis to newer data points.

2.1.10.2 TRAcking and Validating Cluster Evolution using Regression Analysis

The TRACER (TRAcking and validating Clusters Evolution using Regression analysis) [53] framework aims at tracking the detected cluster's evolution over time by building

and maintaining a summarized regression model for each cluster's property (e.g. centroid coordinates, scale, or density). For each detected cluster, a set of regression models is built and maintained throughout the lifetime of the cluster (i.e. until it disappears or merges with other clusters). Four properties are used to describe a cluster: *centroid* (c), *scale* (σ), the *sum of robust weights* (W), or *density* (δ). The regression models are found by estimating the regression coefficients ($\hat{\beta}_0$ and $\hat{\beta}_1$) at intervals of time called *Regression Windows* each having width Δ_{Reg}, and reflects the desired details in the description of the evolution of clusters. The regression window should be relatively small to justify the use of a linear model instead of a more complex model for regression. Using a nonlinear model is possible, but would result in a more complicated estimation of regression coefficients.

The output of regression analysis are the linear regression coefficients β_0 and β_1 which are stored for every Δ_{Reg} data samples, hence for each cluster i at time n (i.e. after encountering n points) the β_0 coefficients are stored as $\beta_{0,c_{i,n}}, \beta_{0,\sigma_{i,n}}, \beta_{0,\delta_{i,n}}$ *and* $\beta_{0,W_{i,n}}$ for the properties $c_{i,n}$, $\sigma_{i,n}$, $\delta_{i,n}$ *and* $W_{i,n}$ respectively, and the same for the slope (β_1) coefficients. These regression coefficients can be used to examine the behavior of the cluster during the lifetime of the data stream.

TRACER: Milestone Detection The stored regression coefficients are used to model or summarize the behavior of each cluster over time, and possibly to predict future cluster behavior or detect any *deviation* from that behavior. These deviations are detected automatically using the R^2 regression diagnostics test, and the times at which these deviations occur are called *milestones*. Milestones represent important phases in the lifetime of a cluster, because they represent when major changes took place either in the structure of the cluster (i.e. internal changes) or in its relationship with other clusters (i.e. external changes). Each time that a *milestone* is detected, a new regression model is built (since it reflects a new behavior that needs to be captured), otherwise the old regression model needs only be updated to reflect the changes that took place. If no milestone was detected between two consecutive time periods $T1[z\Delta_{Reg}+1, (z+1)\Delta_{Reg}]$ and $T2[(z+1)\Delta_{Reg}+1, (z+2)\Delta_{Reg}]$ where z is a constant $z \in \{0, 1, 2, 3...\}$, then the updated model is found as follows:

$$\beta_{1,T1+T2} = \frac{\left[\Delta^2_{Reg} - 1\right] \times [\beta_{1,T1} + \beta_{1,T2}] + 6\Delta_{Reg} \times [\overline{y}_{T2} - \overline{y}_{T1}]}{8\Delta^2_{Reg} - 2} \quad (2.25)$$

$$\beta_{0,T1+T2} = \frac{\overline{y}_{T1} + \overline{y}_{T2}}{2} - \beta_{1,T1+T2} \times \left[2\Delta^2_{Reg}(z+1) + \Delta_{Reg}\right] \quad (2.26)$$

where $\beta_{0,T1}$ and $\beta_{1,T1}$ are the regression coefficients at time period $T1$, $\beta_{0,T2}$ and $\beta_{1,T2}$ are the regression coefficients at time period $T2$, $\overline{y}_{T1}$ and $\overline{y}_{T2}$ are the means of the dependent variable at time periods $T1$ and $T2$ respectively.

TRACER: Profiling and Monitoring Evolving Cluster Behavior Monitoring the regression models can eventually help build *behavioral* profiles for each cluster, which reflects how each of the cluster properties has behaved over time. A stable cluster has properties that remain stable for some time (i.e. the regression line is a plateau), which statistically means that the slope (β_1) is equal or close to *0*. If the cluster is not stable then a test on the sign of the slope can reflect how the cluster quality is changing (e.g. a positive slope in the density metric means that the cluster is improving). Using the confidence intervals CI provides a more reliable test for stability, because we are only interested in a plateau-like regression line and not necessarily a strict plateau.

TRACER is invoked every time Δ_{Reg} data points are encountered, and as an input, requires the Δ_{Reg} values of the cluster properties (σ, δ *and* W). These values are temporarily stored and then are discarded once TRACER is completed. TRACER quantifies the behavior and labels it as *stabilizing*, *increasing* or *decreasing*. When two clusters are to be merged, the new cluster should carry on both the structural (e.g. scale) as well as the behavioral (i.e. stability) information of the two clusters, while giving more importance to the one with higher quality (i.e. higher density). For clusters C_i and C_k to be merged, the stability measures of the higher quality cluster are inherited by the new cluster. The previous regression models of clusters C_i and C_k's metrics are combined only when the regression coefficients are similar, because they may represent different histories of the cluster evolutions, which is not always the case.

Advantages of TRACER Advantages of using TRACER over other comparable methods include:

- Validates the detected cluster on the fly. This is done via the behavioral profiles built for each cluster's metric.
- Low complexity, which is linear with the number of clusters (C) and number of snapshots (S) (which is equal to $\frac{N}{\Delta_{Reg}}$).
- Tracking and detecting the internal and external cluster changes that take place over time, while the other algorithms can only work with the internal changes or the data/domain changes.

2.2 Modeling Text Data and Dimensionality Reduction

The Bag of words (BOW) is a very common and widely used representation of documents. BOW vectors have a very high dimensionality where each dimension may correspond to one concept or one topic. Dimension reduction can be used to find the semantic space and its relationship to the BOW representation [54]. Clustering uses similarity (or dissimilarity) between documents to place them in their respective natural groupings or clusters. Soft clustering techniques can associate each document with multiple clusters, by viewing each cluster as a dimension [55]. Clustering hence induces a low-dimensional representation of documents. Finding a lower dimensional representation that reflects the original representation, and that maintains more of the original information than clustering is very beneficial, and can be used to extract more narrowed-down topics. Interpretation of the compressed dimensions may still be very difficult, and a document can only be fully understood by considering all of the dimensions together. Documents are associated with a number of latent topics, which correspond to both document clusters and compact representations identified from a corpus. Each document is assigned to the topics with different weights, which specify both the degree of membership in the clusters as well as the coordinates of the document in the reduced dimension space. The original feature representation plays a

key role in defining the topics and in identifying which topics are present in each document. The result is an understandable representation of documents that is useful for analyzing the themes in documents. The multinomial distribution [56] is a commonly used probabilistic model for text:

$$\mathcal{M}(X \mid \beta) \propto \prod_{v=1}^{|W|} \beta_v^{x_v}$$

Let D be a corpus of documents, indexed by d. W is the distinct set of terms in the vocabulary, indexed by v. X is a $|W| \times |D|$ matrix encoding the occurrences of each term in each document. K is set of a given number of topics, index bsy i. N_i is the number of tokens assigned to topic i. It captures the relative frequency of terms in a document. It is essentially equivalent to the BOW-vector with $\ell_1 - norm$ standardization as $\sum_{v=1}^{W} \beta_v = 1$. The Dirichlet distribution [13] is a conjugate distribution to the multinomial distribution, therefore serving as a commonly used prior for multinomial models:

$$\mathcal{D}(\theta \mid \alpha) = \frac{\Gamma\left(\sum_{i=1}^{|K|} \alpha_i\right)}{\sum_{i=1}^{|K|} \Gamma(\alpha_i)} \prod_{i=1}^{|K|} \theta_i^{\alpha_i - 1}$$

These distributions favor imbalanced multinomial distributions, where most of the probability mass may be concentrated on a small set of values. Thus, it is well suited for models that reflect commonly observed power law distributions in human language. Latent Semantic Indexing (LSI) [57] is based on the singular value decomposition (SVD) [58] of the term-document matrix. LSI projects both the documents and words into a lower dimensional space representing the semantic aspects of a document. Using these projections, LSI enables the analysis of documents at a conceptual level, overcoming the drawbacks of purely term-based analysis. LSI overcomes the issues of *synonymy* and *polysemy [21].*

Let X be the term-document matrix of a corpus. The d^{th}column, X_d represents a document d in the corpus and the $v - th$ row of the matrix X, denoted by T_v, represents a term v. SVD of X is given as:

$$X = U \sum V^T$$

The matrices U and V are orthonormal and $\sum$ is diagonal.

$$\sum = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{min\{W,M\}} \end{bmatrix}$$

The values $\sigma_1, \sigma_2, \ldots, \sigma_{min\{W,M\}}$are singular values of the matrix X and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{min\{W,M\}}$.

$$\hat{X} = \hat{U}\hat{\sum}\hat{V^T} = \begin{bmatrix} U_1 & \cdots & U_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_K \end{bmatrix} \begin{bmatrix} V_1^T \\ \vdots \\ V_K^T \end{bmatrix}$$

SVD produces the $(rank-K)$ matrix $\hat{X}$ that minimizes the distance from X in terms of the spectral norm and the Frobenius norm. Certain terms are very likely to be present based on the topic of a document. *Latent topic models* capture this idea by modeling the conditional probability that an author will use a term given the topic the author is writing about, by providing a mechanism to explicitly reason about latent topics.

2.3 Topic Modeling

Topic modeling [59] is a set of algorithms generally used for discovering a latent set of topics in any given set of documents. Topic modeling is an unsupervised method that analyzes the words in their original form and discovers the latent topics. In this section, we will provide a brief overview of two of the main approaches for topic modeling: Probabilistic Latent Semantic Indexing (PLSI) in Section 2.3.1 and Latent Dirichlet Allocation (LDA) in Section 2.3.2.

2.3.1 Probabilistic Latent Semantic Indexing

Probabilistic Latent Semantic Indexing (PLSI) was proposed by Hofmann [14], and uses the same conceptual assumptions as in Latent Semantic Indexing (LSI) [60], but follows a probabilistic approach. The initial step in finding the latent models is to convert a given

dataset into a bag-of-words (BOW) representation. BOW is represented as a matrix that captures the frequency (or the existence) of every word in each document. For a set of D documents and W unique words, the BOW matrix X would be of size $|D| \times |W|$ where each row represents a document and each column represents a unique word. PLSI assumes that each word $w \in W$ is generated into a document $d \in D$ that belongs to topic $i \in K$, where K is a set of topics. following the generative probabilistic approach specified in Algorithm 2.2.

Algorithm 2.2 PLSI Generative Probabilistic Approach.

1: First, a document d is sampled following a multinomial distribution of documents $p(d)$
2: Second, a topic i is sampled based on the topic distribution with respect to the selected document $\theta_{di} = p(z = i|d)$.
3: Finally, a term v is sampled based on the multinomial distribution of the selected topic $\Phi_{iv} = p(v = w|z = i)$

PLSI aims at associating a topic z with each term v in each document d. This joint probability can be expressed as:

$$\begin{gathered} p(v, d) = p(d) \times p(v|d) \\ where\ p(v|d) = \sum_{i}^{|K|} p(v|z = i) \times p(z = i|d) \end{gathered} \tag{2.27}$$

The Expectation Maximization algorithm [29] can be used to find these probabilities by maximizing the log-likelihood given by:

$$\mathcal{L} = \sum_{d=1}^{|D|} \sum_{v=1}^{|W|} x_{vd} \times log p(w = v, d) \tag{2.28}$$

$$\mathcal{L} = \sum_{d=1}^{|D|} \sum_{v=1}^{|W|} x_{vd} \times log \sum_{i=1}^{|K|} p(w = v|z = i) \times p(z = i|d) \times p(d) \tag{2.29}$$

where x_{vd} is the frequency of the term v in document d.

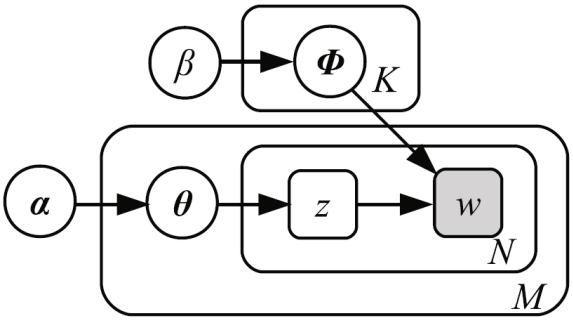


Figure 2.1: LDA Generative Process [3]

2.3.2 Latent Dirichlet Allocation (LDA)

	Description
D	A set of a given number of documents in a collection
W	A set of a given number of distinct words in a vocabulary
N	Total number of words in collection
K	A set of a given number of topics
w_{di}	i^{th} observed word in document d
z_{di}	Topic assigned to w_{di}
N_{wk}	Count of word assigned to topic
N_{dw}	Count of topic assigned in document
ϕ_k	Probability of word given topic k
θ_d	Probability of topic given document d
Γ	Gamma Function

TABLE 2.2

Description of variables used for LDA based algorithms.

PLSI might suffer from over-fitting, as it generates a large number of parameters (i.e. a parameter for each document and for each word into each topic). Also, it cannot find the probability of completely new documents nor incorporate new words. Latent Dirichlet

Allocation (LDA) was proposed by [3] as an alternative that solves the problems of PLSI. LDA follows a similar probabilistic approach to PLSI to generate the data. The list of variables is provided in Table 2.2. The generative process can be summarized as follows:

- The multinomial term distribution $\boldsymbol{\Phi_k} = \{\phi_{k1}, \cdots, \phi_{k|w|}\}$ for topic k is a multinomial distribution that follows a symmetric Dirichlet distribution with parameter in the vector $\boldsymbol{\beta} = \{\beta_1, \cdots, \beta_{|W|}\}$, where Γ denotes the Gamma function.

$$p(\boldsymbol{\Phi_k}|\boldsymbol{\beta}) = \frac{\Gamma(|W|\beta)}{\Gamma(\beta)^{|W|}} \prod_{v=1}^{|W|} \phi_{kv}^{\beta-1} \tag{2.30}$$

- The multinomial topic distribution $\boldsymbol{\theta_d} = \{\theta_{d1}, \cdots, \theta_{d|K|}\}$ for document d is a multinomial distribution that follows a Dirichlet distribution with parameters in the vector $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_{|k|}\}$

$$p(\boldsymbol{\theta_d}|\boldsymbol{\alpha}) = \frac{\Gamma \sum_i^{|K|} \alpha_i}{\prod_{i=1}^{|K|} \Gamma(\alpha_i)} \prod_{i=1}^{|K|} \theta_{di}^{\alpha_i - 1} \tag{2.31}$$

- The topic z_{dn} for each token indexed by n in document d is sampled from document topic distribution $\boldsymbol{\theta_d}$

$$p(z_{dn} = i|\theta_d) = \theta_{di} \tag{2.32}$$

- Each token w is sampled from the distribution associated with the selected topic

$$p(w_{dn} = v|z_{dn} = i, \phi_i) = \phi_{iv} \tag{2.33}$$

The generative process is shown in Figure 2.1. LDA provides a mechanism to find patterns of co-occurrence between the terms, and then using these patterns to find coherent topics. Hence, LDA finds topics in which the most probable terms frequently co-occur together in the documents, thus helping with polysemy. This is quite different from more naive independent term assumptions. Another advantage of LDA is that in the topic-specific term distributions $p(\Phi_i|\beta)$, the Dirichlet prior provides smoothing that assigns non-zero probabilities even to unseen terms in a document [61]. Finding the parameters of LDA (i.e.

β and α) can be accomplished by maximizing the likelihood function:

$$\mathcal{L} = \prod_{d=1}^{|D|}\prod_{n=1}^{|W|} p(w_{dn}|z_{dn},\mathbf{\Phi}) \times p(z_{dn}|\boldsymbol{\theta}_d) \times p(\boldsymbol{\theta}_d|\alpha) \times p(\mathbf{\Phi}|\beta) \tag{2.34}$$

$$= \phi_{zw}\theta_{dz}\frac{\Gamma\left(\sum_{i=1}^{|K|}\alpha_i\right)}{\prod_{i=1}^{|K|}\Gamma\left(\alpha_i\right)}\prod_{i=1}^{|K|}\theta_{di}^{\alpha_i-1}\frac{\Gamma\left(|W|\beta\right)}{\left[\Gamma\left(\beta\right)^{|W|}\right]}\prod_{v=1}^{|W|}\phi_v^{\beta-1} \tag{2.35}$$

Optimizing the likelihood directly is hard since the topic assignments z_{dn} are not given. As a result, approximation methods are used such as Collapsed Gibbs Sampling [62]. An online version of LDA was proposed in [4], which is based on online variational Bayes optimization. Online LDA aims at handling massive collections of documents, where each document is only examined once and then discarded.

2.3.2.1 Efficient Methods of Topic Model Inference

Yao, McCallum et.al [63] evaluated three different sampling-based inference methods *Gibbs1, Gibbs2,* and *Gibbs3* for LDA and proposed a new method SparseLDA. Gibbs sampling is an MCMC [64] method that involves iterating over a set of variables while sampling each one of them. Given enough iterations, Gibbs sampling for LDA produces samples from a posterior $P(z|w)$. The first method *Gibbs1*, samples new topics from the entire corpus; the second method *Gibbs2* samples only new documents, holding the parameters for a training corpus fixed; and the third method *Gibbs3* samples each new document independently. Gibbs3 is therefore an online version. When a new document arrives, the topics are sampled for a number of iterations using only topic word counts from the *training* documents and the new document, this differs from Gibbs1 and Gibbs2 in that it produces estimates of θ_d given only the words from a given *training set* of documents and in document d. Gibbs1 and Gibbs2 produce estimates given the entire data set.

SparseLDA involves a sampling algorithm and a data structure that substantially improves sampling performance. The probability of topic z in document d given an observed word w is given as:

$$
\begin{aligned}
P\left(z=t|w\right) &\propto \left(\alpha_t + n_{t|d}\right)\frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}} \\
P\left(z=t|w\right) &\propto \frac{\alpha_t \beta}{\beta V + n_{\cdot|t}} + \frac{n_{t|d}\beta}{\beta V + n_{\cdot|t}} + \frac{\left(\alpha_t + n_{t|d}\right) n_{w|t}}{\beta V + n_{\cdot|t}}
\end{aligned} \tag{2.36}
$$

where V is the vocabulary, $n_{w|t}$ is the number of tokens for w assigned to topic t and $n_{\cdot|t} = \sum_w n_{w|t}$, $n_{t|d}$ is the total number of tokens in the document assigned to topic t. The SparseLDA divides the process into *three* terms as show in Equation 2.36. The first term only changes when there is a change in α, the second term depends only on the document-topic counts, and the third term changes with the value of w. The method iterates over only a selected set of topics to improve the performance. The efficiency of the method depends on the ability to rapidly identify topics such that $n_{w|t} \neq 0$. The method also uses a data-structure tuple $\left(t,\, n_{w|t}\right)$ in a single 32 bit integer by dividing the bits into *count* segment and a *topic* segments The tuple helps the method improve the storage and retrieval efficiency.

2.3.3 Online Topic Modeling using Online Variational Bayes Inference for LDA

Algorithm 2.3 Online LDA with Variational Bayes [4].

Function: OnlineLDAVB()
Input: *A list of documents, hyper-parameters α and η.*
Output: *Topic and Word distributions.*

Define $\rho_t \triangleq (\tau_0 + t)^{-k}$
Initialize λ randomly
for t=0 to ∞ **do**
  Initialize $\gamma_{tk} = 1$
  repeat
    Set $\phi_{twk} \propto \exp\{\mathbb{E}_q\left[\log \theta_{tk}\right] + \mathbb{E}_q\left[\log \beta_{kw}\right]\}$
    Set $\gamma_{tk} = \alpha + \sum_w \phi_{twk} n_{tw}$
  until $\frac{1}{K}\sum_k |change\, in\, \gamma_{tk}| < 0.00001$
  Compute $\tilde{\lambda}_{kw} = \eta + D n_{tw} \phi_{twk}$
  Set $\lambda = (1-\rho_t)\lambda + \rho_t \tilde{\lambda}$
end for

The Batch Variational Bayes algorithm has constant memory requirements and empirically converges faster than batch collapsed Gibbs sampling [3, 4]. Yet the batch Variational Bayes still requires a full pass through the entire corpus each iteration and therefore

applying it on large datasets takes a lot of time. Hoffman and Blei et.al [4] have proposed an online variational inference algorithm for fitting λ, the parameters to the variational posterior over the topic distributions β. The algorithm, being simple, converges faster than the batch version. The list of variables is provided in Table 2.2.

The posterior over the per-word topic assignments z is parameterized by ϕ, the posterior over the per-document topic weights θ is parameterized by γ, and the posterior over the topic β is parameterized by λ. A good setting of the topics λ is one for which the Evidence Lower Bound (ELBO) $\mathcal{L}$ [20] is as high as possible after fitting the per-document variational parameters γ and ϕ with the Expectation step. Let $\gamma(n_d, \lambda)$ and $\phi(n_d, \lambda)$ be the values of γ_d and ϕ_d produced by the E-step. The goal is to set λ to maximize Equation 2.37, below.

$$\mathcal{L}(n, \lambda) \triangleq \sum_d \ell\left(n_d, \gamma\left(n_d, \lambda\right), \phi\left(n_d, \lambda\right), \lambda\right) \tag{2.37}$$

where $\ell\left(n_d, \gamma_d, \phi_d, \lambda\right)$ is the d^{th} document's contribution to the variational bound. The online VB for LDA is provided in Algorithm 2.3. As the t^{th} vector of word counts n_t is observed, an E step is performed to find locally optimal values of γ_t and ϕ_t, while holding λ fixed. In the true online case $D \to \infty$, corresponding to empirical Bayes estimation of β. λ is updated using a weighted average of its previous value and $\bar{\lambda}$. The weighted value given to $\bar{\lambda}$ is given by $\rho_t \triangleq (\tau_0 + t)^{-k}$, where $k \in (0.5, 1]$ controls the rate at which old values of $\bar{\lambda}$ are forgotten and $\tau \geq 0$ slows down the early iterations of the algorithm. Algorithm 2.3 presents pseudocode for the online LDA with variational bayes.

2.3.4 Hyper Parameter Estimation

Blei et.al. [3] and [65], discussed a method to optimize the hyper-parameters using the Newton-Raphson method [66], which can be used for estimating maximum likelihood of a Dirichlet distribution.

The Newton's method optimized a hyper parameter as:

$$\alpha_{\text{new}} = \alpha_{\text{old}} - H(\alpha_{\text{old}})^{-1} g(\alpha_{\text{old}})$$

where $H(\alpha)$ and $g(\alpha)$ are the Hessian matrix and gradient respectively at the point α. If the Hessian matrix is of the form:

$$H = \text{diag}(h) + \mathbf{1}z\mathbf{1}^{\text{T}}, \tag{2.38}$$

where $\text{diag}(h)$ is a diagonal matrix with the elements of the vector h along the diagonal, and applying matrix inversion we get:

$$H^{-1} = \text{diag}(h)^{-1} - \frac{\text{diag}(h)^{-1}\mathbf{1}\mathbf{1}^{\text{T}}\text{diag}(h)^{-1}}{z^{-1} + \sum_{j=1}^{k} h_j^{-1}}$$

Multiplying by the gradient, the ith component can be obtained as:

$$(H^{-1}g)_i = \frac{g_i - c}{h_i}$$

where

$$c = \frac{\sum_{j=1}^{k} g_j/h_j}{z^{-1} + \sum_{j=1}^{k} h_j^{-1}}.$$

2.3.5 Online LDA with Infinite Vocabulary

Topic models based on LDA [12] assume a predefined vocabulary. Zhai et.al. [5], presented a extension to Online LDA [12, 20], by drawing topics from a Dirichlet process which is a distribution over all strings instead of a finite Dirichlet distribution. The generative process is identical to that of the LDA. Algorithm 2.4 presents the generative process and the rest being identical of Online LDA algorithm presented it Section 2.3.3.

We present a set of experiments in Section 4.3.4, where we extended the Online LDA with infinite vocabulary by introducing the *automatic* hyperparameter estimation as discussed in Section 2.3.4.

Algorithm 2.4 Generative process for LDA with Infinite Vocabulary [5].

1: **for all** topics k **do**

2: Draw words $\rho_{kt}, (t = 1, 2, ...)$ from G_0

3: Draw b_{kt} $Beta(1, \alpha^{\beta}), (t = 1, 2, ...)$ from G_0

4: Set stick weights $\beta_{kt} = b_{kt}\Pi_{s<t}(1 - b_{ks})$

5: **end for**

2.3.6 Topic Modeling Evaluation

2.3.6.1 Perplexity

The most common way to evaluate a probabilistic model is to measure the log-likelihood of a held-out test set [67]. This is usually done by splitting the dataset into two parts: one for training, the other for testing. For LDA, a test set is a collection of unseen documents w_d, and the model is described by the topic matrix Φ and the hyperparameter α for the topic-distribution of documents. The LDA parameters θ is not taken into consideration as it represents the topic-distributions for the documents of the training set, and can therefore be ignored to compute the likelihood of unseen documents. Therefore, we need to evaluate the log-likelihood $\mathcal{L}(w)$ of a set of unseen documents w_d given the topics Φ and the hyperparameter α for the topic-distribution θ_d of documents as shown in Equation 2.39.

$$\mathcal{L}(w) = \log p(w|\Phi, \alpha) = \sum_d \log p(w_d|\Phi, \alpha) \tag{2.39}$$

The likelihood of unseen documents can be used to compare models; with higher likelihood (or lower perplexity) implying a better model. The measure, traditionally used for topic models, is the *perplexity* of held-out documents w_d defined as:

$$\textit{perplexity}(\text{test set } w) = \exp\left\{-\frac{\mathcal{L}(w)}{\text{count of tokens}}\right\}$$

which is a decreasing function of the log-likelihood $\mathcal{L}(w)$ of the unseen documents w_d; the lower the perplexity, the better the model [67].

2.4 Sentiment Analysis

The rise of social media has fueled a great interest in sentiment analysis. With the proliferation of online reviews, ratings, various user recommendations, and other forms of online user expression, sentiment analysis provides a way to look into what the users are discussing in regards to a given topic. Sentiment analysis helps in automating the process of filtering out the noise, better understand conversations, identifying relevant content, and *actioning* it appropriately [68]. A common problem is that most sentiment analysis

algorithms use simple terms to express sentiment. Cultural factors, linguistic nuances and differing contexts make it extremely difficult to identify the correct sentiment of the written text [68]. Also, the fact that humans often disagree on the sentiment of text illustrates how big a task it is. The shorter the string of text, the harder it becomes. Sentiment analysis has shown that Twitter can be seen as a valid online indicator of political sentiment [69]. Twitter political sentiment demonstrates close correspondence to various political positions, indicating that the tweets plausibly reflects the offline political landscape [69].

Sentiment analysis approaches may be grouped into four main categories: *keyword spotting*, *lexical affinity*, *statistical methods*, and *concept-level* techniques [70]. Keyword spotting classifies text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored [71]. Lexical affinity not only detects obvious affect words, it also assigns arbitrary words a probable *"affinity"* to particular emotions [72]. Statistical methods leverage on elements from machine learning such as latent semantic analysis, support vector machines etc. Unlike purely syntactical techniques, concept-level approaches leverage on elements from knowledge representation such as ontologies and semantic networks and, hence, are also able to detect semantics that are expressed in a subtle manner [73]. Hatto and Eric et.al. [74] has proposed *Valence Aware Dictionary for sEntiment Reasoning* (VADER), which uses a combination of qualitative and quantitative methods. VADER constructs and empirically validates a gold-standard list of lexical features (along with their associated sentiment intensity measures) which are specifically attuned to sentiment in microblog-like contexts. These lexical features are combined with consideration for a set of general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity.

2.4.1 Joint Sentiment Topic Modeling: The Batch Approach

Latent Dirichlet Allocation is effectively a generative model from which a new document can be generated in a predefined probabilistic procedure. The existing framework of LDA has three hierarchical layers, where topics are associated with documents, and words

are associated with topics. In order to model document sentiments, a joint sentiment layer, between the document and the topic layers, was proposed by Lin, He et.al. [75]. Joint Sentiment Topic modeling (JST) uses a lexicon based training data set containing positive, negative and neutral scores for each lexicon. JST is effectively a four layer model, where sentiment labels are associated with documents, under which topics are associated with sentiment labels and words are associated with both sentiment labels and topics [75].

Let there be D set of documents, denoted by collection $C = \{d_1, d_2, \cdots, d_D\}$; where each document in the corpus is a sequence of N_d words denoted by $d = (w_1, w_2, \cdots, w_{N_d})$, and each word in the document is an item from a vocabulary index with V distinct terms denoted by $\{1, 2, \cdots, V\}$. Let S be the number of distinct sentiment labels and T be the total number of topics. The generative process for the hierarchical Bayesian model is shown in Algorithm 2.5.

Algorithm 2.5 Generative process for the Hierarchical Bayesian model [75].

for all document d **do**
  choose a distribution $\pi_d \sim Dir(\gamma)$
  for all sentiment label l under document d **do**
    choose a distribution $\theta_{d,l} \sim Dir(\alpha)$
    for all word w_i in document d **do**
      choose a sentiment label $l_i \sim \pi_d$
      choose a topic $z_i \sim \theta_{d,l_i}$
      choose a word w_i from the distribution over words defined by the topic z_i and sentiment label l_i, φ_{zi}^{li}
    end for
  end for
end for

The hyper-parameters $\alpha$and β in JST can be treated as the prior observation counts for the number of times topic j is associated with sentiment label l, sampled from a document and the number of times words sampled from topic j associated with sentiment label l, respectively, before having observed any actual words [75]. The hyper-parameter γ can be interpreted as the prior observation count for the number of times sentiment label l sampled from documents before any words from the corpus are observed. In JST, there are three sets of latent variables that we need to infer, including the joint sentiment/topic-document

Algorithm 2.6 Gibbs sampling procedure for JST [75].

Input: *A list of documents, hyper-parameters, lexicons with sentiments.*
Output: *A list of topics per each sentiment.*

Initialize $V \times T \times S$ matrix Φ, $T \times S \times D$ matrix Θ, $S \times D$ matrix Π.
for $m = 1$ to M Gibbs sampling iterations **do**
  repeat
    Read a word i from a document
    Calculated the probability of assigning word i to topic and sentiment label.
    Sample a topic j based on the estimated probability.
    Sample a sentiment label k.
    Update the matrix Φ, Θ and Π with new sampling results.
  until all words have been processed
end for

distribution and the joint sentiment-document distributions. Algorithm 2.6 lists the steps involved in JST.

2.5 Relevance Feedback

In the field of Information Retrieval [76, 77], "Relevance Feedback" is used as a mechanism to get the user involved, to improve the final results. The process is based on the idea that it is difficult to formulate a good query when the user does not have any idea on the data collection, but it is easy to judge particular documents, and so it an interactive query refinement provides a very good platform. In such a scenario, relevance feedback can also be effective in tracking a user's evolving information need [76]: seeing some documents may lead users to refine their understanding of the information they are seeking [76]. The user provides feedback on the initial set of results, specifying the documents that are relevant and irrelevant [76]. The procedure involves the user issuing a simple query and the system returning an initial set of results [76]. The user then marks some of these returned results as relevant or irrelevant. The system then computes a better representation of the information based on this user feedback [76]. This feedback and re-computation can go for more than one iteration. The Rocchio Algorithm [78] is a classic algorithm for relevance feedback. It models a way of incorporating relevance feedback information into the vector space model, where the representation of a set of documents is in terms of vectors in a common vector

space [76].

Let a query vector be represented as $\overrightarrow{q}$, that maximizes similarity with a set of relevant documents while minimizing the similarity with irrelevant documents. If C_r is the set of relevant documents and C_{nr} is the set of irrelevant documents, then the optimal query is given as [79]:

$$\vec{q}_{opt} = \arg\max_{\overrightarrow{q}} \left[sim\left(\overrightarrow{q}, C_r\right) - sim\left(\overrightarrow{q}, C_{nr}\right) \right]$$

where *sim* can be the *cosine* similarity. The optimal query ($\vec{q}_{opt}$) is the vector difference between the centroids of the relevant and irrelevant documents for separating the relevant and irrelevant documents is given as [79]:

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

The Rocchio Algorithm [77] uses a modified query $\vec{q}_m$ given as:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \tag{2.40}$$

where, q_0 is the initial query vector. D_r and D_{nr} are the set of known relevant and irrelevant documents, and α, β, γ are the weights to each respectively. These weights control the balance of trusting the judged documents set against the query [77]. Starting from the initial query, each new query is expected to move towards the centroid of the relevant documents in the vector space. The positive quadrant of the vector space can be obtained by subtracting off an irrelevant documents vector [77].

Pseudo relevance feedback [80–82], provides a mechanism for automatic local analysis where the manual part of choosing the relevant documents is automated and has the advantage that assessors are not required. The aim of the mechanism is to perform normal information retrieval to find an initial set of most relevant documents, then choose the top k ranked documents as the most relevant, and finally to do relevance feedback under this assumption [83].

2.6 Summary and Conclusions

This chapter discussed Clustering along with basic clustering algorithms and few similarity methods. The chapter then presents the Stream-Dashboard framework and it's components like RINOStreams and TRACER. The chapter then discussed Topic Modeling techniques PLSI, LDA, Online LDA etc. The chapter then presented sentiment analysis, and relevance feedback to help the Compartmentalized Topic Modeling Framework to extract quality topics.

Latent Dirichlet Allocation (LDA) provides a sophisticated way of extracting topics from documents but suffers from the fact that it requires multiple passes over the dataset. The Online LDA with Variational Bayes provides a faster convergence compared to LDA. However, using Online LDA alone may not be suitable for streaming data. This is because, streaming data tends to become very diverse and heterogeneous over time, which in-turn makes pre-defining the number of topics challenging and may result in loss of information. Sentiment analysis can help understand the changes in the general public's perspective regarding a given topic, which can in turn help explain given topic in a better fashion. Query reformulation helps explore the unknown dimensions or terms of a given topic and thus narrow the focus or expand a given topic dynamically. Chapter 4 will present a new framework that uses streaming data to generate topics on the fly or per user queries. The proposed framework leverages the power of Stream-Dashboard, Topic Modeling, Sentiment Analysis, and Query Reformulation to extract a set of high quality topics and their trends.

CHAPTER 3

AFTER-STREAMS: A NEW STREAM CLUSTERING ALGORITHM WITH ADAPTIVE FORGETTING

In this chapter, we present a new stream clustering algorithm, that we call AFTER-Streams (**A**daptive **F**orgetting **T**ime-Decay **E**volving **R**obust Stream Clustering Algorithm). The algorithm employs *Adaptive/Dynamic Temporal Decay* instead of a constant decay factor that RINO-Streams [22] uses.

The rest of this chapter is organized as follows. Some definitions are first presented in Section 3.2, then the equations for incremental updates of the centroid and scale are presented in Section 3.2.1. Section 3.2.2 discusses the use of Chebyshev bounds to detect outliers, while merging and splitting clusters are discussed in Section 3.2.3, and the complete AFTER-Streams algorithm pseudo code is listed in Section 3.2.4. The time and memory complexities of AFTER-Streams are discussed in Section 3.2.5, and the relation between the proposed objective function and M-estimators is discussed in Section 3.2.6. Section 3.2.7 shows how AFTER-Streams complies with the requirements of online stream clustering algorithms. Section 3.2.8 analyzes the differences between AFTER-Streams and the most competitive stream clustering algorithms from the literature. Finally Section 3.3 presents our evaluation experiments.

3.1 Problem Statement

Given a data stream X, with new data points x_i arriving at time i (or more accurately, sequential/order), we are interested in detecting a set of evolving clusters ζ (i.e. a clustering model) that reflects the *evolving behavior* of the data stream, while being robust to outliers. Each cluster represents a portion of a the data stream seen so far, where the

data points in this portion are more similar to each other than data points in other clusters. Each cluster consists of a set of metrics/properties that distinguish it from other clusters. Such metrics include a cluster representative (i.e. cluster centroids), scales (i.e. the size of a cluster's influence area in terms of temporal and content space). Data streams are characterized by a dynamic nature, where new clusters emerge, old ones may undergo changes in their metrics (i.e. internal changes), merge together if they become very similar, or split if they become too general (i.e. external changes). Hence, the classical definition of a cluster needs to be modified to capture the reality of evolving clusters in a data stream.

Cluster partitions on evolving data streams are often computed based on certain time intervals. There are three popular models: landmark window, sliding window, and damped window. In the damped window model, the weight of each data point decreases exponentially with time t via a fading function $f(t) = 2^{-\lambda \cdot t}$, where $\lambda > 0$. The higher the value of λ, the lower the importance of the historical data compared to more recent data. The overall weight of the data stream is a constant $W = v\left(\sum_{t=0}^{t=t_c} 2^{-\lambda t}\right) = \frac{v}{1-2^{-\lambda}}$, where t_c is the current time and v denotes the speed of the stream (i.e. the number of points arrived in one unit time) [38].

In the following sections, we will extend the temporal importance (λ) to become (1) cluster-specific and (2) adtaive to the speed to the data streams.

3.2 Definitions

In the following, we present the definitions related to AFTER-Streams. We note that most definition as based on RINO-Streams [22] except for the introduction of a new temporal scale parameter which will affect all teh other parameter update equations and the objective functions. We start with viewing a data stream X as consisting of a set of data points (x_i) that are indexed based on the order of their arrival i (that we also call time stamp or timestamp) , and presented as: $x_1, x_2 ...., x_N$, where N is the size of the data stream and x_i is a d-dimensional data record (i.e. $x_i = \left(x_i^1, .., x_i^d\right)$). Each cluster i at time n (i.e. after receiving n points) is defined as follows:

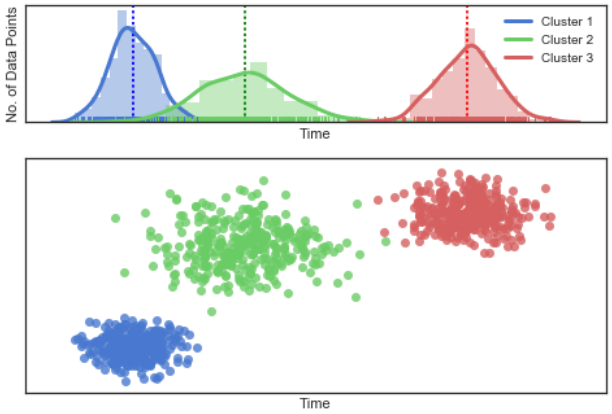


Figure 3.1: Sample data distribution with content space and temporal space. Dotted vertical lines show the temporal centroids.

Definition 3.2.1. *Cluster: The i^{th} cluster at time n ($C_{i,n}$) is defined using two spatial parameters: a centroid $c_{i,n}$ and a scale $\sigma^2_{i,n}$. Figure 3.1 shows the distribution of a sample dataset in both content space and temporal space. The centroid $c_{i,n}$ and scale $\sigma^2_{i,n}$ correspond to the bottom part of the figure. The data stream points may arrive in the order of the horizontal axis, or a random one. Two additional measures are used to keep track of stream-specific properties, namely the soft cardinality $W_{i,n}$ (the sum of the robust weights of the data points) and the age of the cluster, a_i. In addition a temporal scale parameter $\tau^2_{i,n}$ captures the speed of decay in $C_{i,n}$ with a temporal centroid $c'_{i,n}$.*

The centroid $c_{i,n}$ represents the location of the i^{th} cluster center at any time step (n), while ***the scale*** $\sigma^2_{i,n}$ represents the size of an influence zone around the centroid, where data from the stream has landed in the past. Both centroid and scale are affected by a robust weight function w_{ij} (See Definition 2) that is defined for each data point, relative to each cluster, and that decreases with the distance from the data instance to the cluster centroid, and also decreases with the timestamp of arrival of the data in the stream [22]. Hence, newer data points would have more impact on the model than older ones, which allows capturing the evolution of clusters over time. ***The soft cardinality*** W_i is the sum of the robust weights of each data point belonging to the cluster, and is one indicator of the quality of the cluster: a high cardinality means that the cluster represents a large portion of the data stream. The ***age*** a_i is defined as the difference between the current data arrival's

timestamp (n) and the time when the cluster was first created. In addition to keeping track of the cluster creation time, the age is used to compute a ***grace period*** a_{mature} for the cluster before it becomes eligible for testing its quality based on the minimum density threshold δ_{min}, which is an important criterion for cluster survival. Shielding new clusters from premature deletion serves to prevent deleting clusters that are still in infancy stage [22].

In addition to the centroid $c_{i,n}$ and scale $\sigma^2_{i,n}$ in content space, the cluster $C_{i,n}$ also uses two additional corresponding parameters in temporal space. One is an additional temporal centroid $c'_{i,n}$ and the other is a temporal scale $\tau^2_{i,n}$. Figure 3.1 (top part) shows the temporal centroid $c'_{i,n}$ and temporal scale (i.e. influence zone or width) $\tau^2_{i,n}$. In the process of clustering a new data stream point, both the temporal distance and the distance in content space are used to assign the robust weight as discussed in the next sections.

Definition 3.2.2. *Adaptive Robust Weight: At any given timestamp n (i.e. after the arrival of data points $x_1, x_2, ..., x_n$ in the stream), the robust weight of the j^{th} data point, arriving at timestamp j, $1 \leq j \leq n$, is defined as follows, for the i^{th} cluster, C_i, $i = 1, ..., K$*

$$w_{ij,n} = e^{-\left(\frac{d^2_{ij}}{\kappa\sigma^2_{i,n}} + \frac{n-j}{\tau}\right)} \tag{3.1}$$

where τ is an optional application-dependent parameter that controls the desired time decay or forgetting rate ($e^{-1/\tau}$) of the old data points (when needed, i.e. $\tau < \infty$), and how much importance is given to newer data points, d^2_{ij} is the distance from the j^{th} data point to the i^{th} cluster's centroid c_i, $\sigma^2_{i,n}$ is the scale of cluster i at timestamp n, and κ is a tuning constant that can be used for some applications, particularly for data with high dimensionality [22].

Without the time decay ($\tau \to \infty$), the robust weight $w_{ij,n}$ is essentially a Welsh estimator (Table 2.1) as we will show in Section 3.2.6. W-estimators are used to optimize the scaled residuals (i.e. $\frac{r^2}{\sigma}$), hence, the distances are divided by the scale of the distances as shown in (2.13). Moreover, in many robust estimators, a tuning constant (c) is used along with the scale, as shown in Table 2.1, thus the total scale is typically set to a constant

multiplier of MAD [44], i.e. $c \times MAD$, where MAD is the median of the absolute deviations of distances given in (2.16). However, in the context of data streams, the data point is seen only once and then discarded; hence, neither the past data nor their distances to the clusters are kept. We solve this problem by exploiting the fact that for normal data scaled squared Euclidean distances follow a Chi-Square distribution which variance equal to $2d$, where d is the number of dimensions. Hence, the distances can be normalized by the standard deviation of a Chi-Square distribution, which is equal to $\sigma_{\chi^2} = \sqrt{2d}$ (i.e $\kappa = \sqrt{2d}$) [22].

The second term ($e^{\frac{-(n-j)}{\tau}}$) represents a forgetting factor, that causes the weight of the data point j to decrease geometrically by the value of $n - j$. Hence a new data (with $j = n$) would have a higher weight since the forgetting factor would be close to 1, while an older data point (with $j \lll n$) would have a lower value for the forgetting factor (i.e. approaching 0 as n increases), which results in a smaller weight [22].

Assuming that the parameters of the model do not change significantly with every new point, then each old data point's weight, and thus its influence on the cluster parameter estimates, can be easily shown to decrease, after the arrival of each additional new data point, as follows [22]:

$$w_{ij,n} = e^{\frac{-1}{\tau}} w_{ij,n-1} \tag{3.2}$$

Thus the time forgetting factor controls the speed of forgetting older data. As $\tau \to \infty$, the time decay rate $1/\tau \to 0$, resulting in a maximal forgetting factor ($e^{\frac{-1}{\tau}} \to 1$), meaning that no forgetting occurs, and both the oldest and the most recent data would contribute equally to the parameter estimation [22].

Definition 3.2.3. *Temporal Aware Adaptive Robust Weights: At any given timestamp n (i.e. after the arrival of data points $x_1, x_2, ..., x_n$ in the stream), the content and temporal space robust weights of the j^{th} data point, arriving at timestamp j, $1 \leq j \leq n$, is defined as follows, for the i^{th} cluster, C_i, $i = 1, ..., K$*

$$w_{ij,n} = e^{-\left(\frac{d_{ij}^2}{\kappa \sigma_{i,n}^2}\right)} \tag{3.3}$$

$$f_{ij,n} = e^{-\left(\frac{\eta_{ij}^2}{\rho\tau_{i,n}^2}\right)} \tag{3.4}$$

where $\sigma_{i,n}^2$ and $\tau_{i,n}^2$ are the scales in content and temporal spaces of cluster i at timestamp n respectively. κ and ρ are tuning constants that can be used for some applications, particularly for data with high dimensionality. η_{ij}^2 is a unit temporal distance from the j^{th} data point to the i^{th} cluster's temporal centroid c_i'

The temporal robust weight $f_{ij,n}$ replaces the second term $e^{\frac{-(n-j)}{\tau}}$ from Definition 3.2.2, i.e. a dynamic temporal decay is employed instead of a constant forgetting factor. Thus, while using a constant forgetting factor, $f_{ij,n}$ can be interchangeably replaced with $e^{\frac{-(n-j)}{\tau}}$.

Definition 3.2.4. *Temporal Aware Sum Of Weights: For the i^{th} cluster, $C_i, i = 1, ..., K$, the sum of the robust weights of the data points $W_{i,n}$ at time n is defined as follows (using Definition 3.2.3):*

$$W_{i.n} = \sum_{j=1}^{n} w_{ij} f_{ij} \tag{3.5}$$

Given that the robust weights decrease with the distance from the respective cluster centroids of content and temporal spaces, the sum of weights will decrease for an older cluster, if no new data arrives to land in its influence zones.

Definition 3.2.5. *The Clustering Model: The clustering model at time n is defined as follows:*

$$\zeta_n = C_1 \cup C_2 \cdots \cup C_K \tag{3.6}$$

where, $C_i = (c_{i,n}, c_{i,n}'\sigma_{i,n}^2, \tau_{i,n}^2, a_i, W_{i,n})$

Also, $c_{i,n}'$ and $\tau_{i,n}^2$ are applicable only when a dynamic temporal decay is employed.

Definition 3.2.6. *Temporal Aware Density Optimization Function: After encountering n data points, we search for a maximum of K cluster centroids $c_{i,n}$, K cluster temporal centroids $c_{i,n}'$, and scales $\sigma_{i,n}$ and $\tau_{i,n}$, by optimizing the density objective function:*

$$\max_{c_{i,n},c'_{i,n},\sigma_{i,n},\tau_{i,n}} \left\{ \sum_{i=1}^{K} \delta_{i,n} \right\} i = 1, ..., K, \;\; \sigma_{i,n}^2 > 0 \;\; \tau_{i,n}^2 > 0 \tag{3.7}$$

where

$$\delta_{i,n} = \sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n}}{\sigma_{i,n}^2 \tau_{i,n}^2} \tag{3.8}$$

The robust weights $w_{ij,n}$ and $f_{ij,n}$ can be considered as the degrees of membership of the point j in the cluster i in content and temporal spaces respectively. The sum of the weights $W_{i.n}$ for each cluster represents the soft cardinality of that cluster in terms of both the content and temporal spaces.

The scales $\sigma_{i,n}$ and $\tau_{i,n}$ are related to the size of the influence zone of the cluster (i.e. all points inside that zone are considered part of the cluster). Hence, a small scale means that it is a good and compact cluster.

The density of the cluster, $\delta_{i,n}$ in (3.8), combines these metrics of the cluster, and hence it increases as the soft cardinality increases and the scale decreases. The advantage of optimizing the density, which combines the two metrics, is that judging the quality of the cluster using only the sum of weights (the numerator) is not enough, because a cluster with a large number of points is not desirable from the point of view of density, unless these data points are confined within a small influence zone [22].

Note that we added an inequality constraint on the scale in (3.7) to make sure that it remains greater than zero. Without this constraint, the optimization of the density with respect to the scale can lead to the scale shrinking to zero. This is because a zero scale would result in a singleton cluster which is a degenerate optimum of the objective function [22].

3.2.1 Incremental Optimization of the Cluster Density Criterion

Optimizing the density objective function is done using alternating optimization, where finding each parameter is done by fixing all the other parameters, and the same process is repeated for each parameter in an alternating fashion. This is the same optimization approach used in the Expectation Maximization (EM) algorithm [84].

The first step is to turn the objective function in (3.7) for each cluster C_i into a Lagrangian, to include the constraints as follows:

$$\mathcal{L} = \delta_i - \lambda\sigma_i^2 - \lambda\tau_i^2 = \sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n}}{\sigma_{i,n}^2 \tau_{i,n}^2} - \lambda\sigma_i^2 - \lambda\tau_i^2 \tag{3.9}$$

Now we find the optimal values by first setting the derivative of the Lagrangian with respect to the centroid to zero while fixing the scale. Then we set the derivative with respect to scale to zero while fixing the centroid. We need also to check for the Karush-Kuhn-Tucker conditions to handle the inequality condition on $\sigma_{i,n}^2$ and $\tau_{i,n}^2$.

Theorem 3.2.1. *Temporal Aware Optimal Incremental Centroid Update : Given the previous centroids, $c_{i,n-1}$, and assuming that the scales do not change much relative to the scale that resulted from the previous iteration, the new centroid that optimizes (3.9) after the arrival of the n^{th} data point is given by:*

$$c_{i,n} = \frac{c_{i,n-1} W_{i,n-1} + w_{in,n} f_{in,n} x_n}{W_{i,n-1} + w_{in,n} f_{in,n}} \tag{3.10}$$

Proof. The proof can be found in the Appendix (Theorem A.1.4). □

The centroid update in (3.10) can be applied to centroids in both the content and temporal spaces. The first term in the numerator (and the denominator) represents the previous knowledge about the location of the centroid obtained from the points $(x_1, .., x_{n-1})$. The second term in the numerator (and denominator) represents the new information obtained from the new data point x_n from the both the content perspective and the temporal perspective.

Theorem 3.2.2. *Temporal Aware Optimal Incremental Centroid Update : Given the previous temporal centroids, $c'_{i,n-1}$, and assuming that the temporal scales do not change much relative to the scale that resulted from the previous iteration, the new temporal centroid that optimizes (3.9) after the arrival of the n^{th} data point at time t_n is given by:*

$$c'_{i,n} = \frac{c'_{i,n-1} W_{i,n-1} + w_{in,n} f_{in,n} t_n}{W_{i,n-1} + w_{in,n} f_{in,n}} \tag{3.11}$$

Proof. The proof can be found in the Appendix (Theorem A.1.5). □

Theorem 3.2.3. *Temporal Aware Optimal Incremental Scale Update: Given the previous content space scale $\sigma^2_{i,n-1}$, the new scales that optimizes (3.7) after the arrival of the n^{th} data point is given by:*

$$\sigma^2_{i,n} = \frac{\kappa\left(\sigma^2_{i,n-1} W_{i,n-1}\right) + w_{in,n} f_{in,n} d^2_{ij}}{\kappa\left(W_{i,n-1} + w_{in,n} f_{in,n}\right)} \tag{3.12}$$

Proof. The proof can be found in the Appendix (Theorem A.1.6). □

Theorem 3.2.4. *Temporal Aware Optimal Incremental Scale Update: Given the previous temporal scale scale $\tau^2_{i,n-1}$, the new scale that optimizes (3.7) after the arrival of the n^{th} data point is given by:*

$$\tau^2_{i,n} = \frac{\rho\left(\tau^2_{i,n-1} W_{i,n-1}\right) + w_{in,n} f_{in,n} \eta^2_{ij}}{\rho\left(W_{i,n-1} + w_{in,n} f_{in,n}\right)} \tag{3.13}$$

Proof. The proof can be found in the Appendix (Theorem A.1.7). □

Similar to the centroid update equation, the first term in the numerator (and denominator) represents the sum of the contributions of all the previous data points $(x_1, .., x_{n-1})$. The second term represents the new information obtained from the new data point x_n.

3.2.2 Detecting outliers using Chebyshev bounds

Outliers are a common nuisance in raw data sets, and they can be due to many reasons such as Human error, machine error, or the randomness of a few data points that follow no cluster. Detecting outliers is a very challenging task in data mining, and is even more challenging in mining data streams. This is because in stream data mining, the data points are processed only once, and since there is no control over the flow of data, a data

point that is flagged as an outlier at the beginning of the data stream might turn out to be part of a cluster that evolves later in the data stream lifetime [22].

The proposed algorithm, AFTER-Streams, is resistant to outliers because its objective function is rooted in robust statistics (by virtue of using an objective function that resists outliers using robust weights) as shown in Section 3.2.6, and not in standard non-robust estimation methods that make rigid assumptions about the distribution of the data. An outlier is defined as a data point that does not belong to any of the existing clusters (i.e. not in their influence zone) and that does not form any cluster with other points. If the point is determined to be an outlier with respect to all existing clusters, then it will create a new cluster with the point itself being the centroid. This newly created cluster will be allowed a grace period, a_{mature}, and if after this threshold, it is still weak (it has a density less than a threshold δ_{min}), then it will be considered an outlying cluster and will be deleted [22].

In order to tackle outliers, we need to define the notion of a cluster's influence zone, which is an area around the centroid, that bounds the normal data inside the cluster. There exist some upper tail bounds in statistics, that bound the total probability that some random variable is in the tail of a distribution with some significant value (i.e. far from the mean) [43]. One of these bounds is the Chebyshev bound [85], which is a tight bound that, unlike bounds such as the Chernoff bounds for example, relies on no assumptions about the distribution of the data. The only assumption is that a reliable scale is available, which is available using AFTER-Streams by virtue of its robust estimation [22].

Definition 3.2.7. *Chebyshev Bounds : The Chebyshev bound for a random variable Y in a distribution with mean μ and standard deviation σ for any real number $t > 0$, is given by:*

$$Pr\{|Y - \mu| \geq t\sigma\} \leq \frac{1}{t^2} \tag{3.14}$$

The Chebyshev inequality can be rearranged as follows:

$$Pr\left\{|Y-\mu|^2 \geq t^2\sigma^2\right\} \leq \frac{1}{t^2} \tag{3.15}$$

For data with d dimensions, the bound can be written as

$$Pr\left\{\sum_{m=1}^{d}|Y_m-\mu_m|^2 \geq t^2\sum_{m=1}^{d}\sigma_m^2\right\} \leq \frac{1}{t^2} \tag{3.16}$$

which, in the simple case of $\sigma_m = \sigma$, becomes

$$Pr\left\{\sum_{m=1}^{d}|Y_m-\mu_m|^2 \geq t^2.d.\sigma^2\right\} \leq \frac{1}{t^2} \tag{3.17}$$

The Chebyshev bound allows us to design an outlyingness test for any new data point with respect to cluster C_i with significance probability $1/t^2$. The rearranged Chebyshev inequality in (3.15) can be applied directly on the robust weight as follows [22, 43]:

$$Pr\left\{e^{\frac{-|Y-\mu|^2}{\kappa\sigma^2}} \leq e^{\frac{-t^2}{\kappa}}\right\} \leq \frac{1}{t^2} \Longrightarrow Pr\left\{w_{ij} \leq e^{\frac{-t^2}{\kappa}}\right\} \leq \frac{1}{t^2}, \tag{3.18}$$

which means that if the robust weight w_{ij} of data point j with respect to cluster C_i is less than the constant value of $e^{\frac{-t^2}{2}}$, then point j is considered an outlier with respect to cluster C_i with a significance probability of $\frac{1}{t^2}$. This means that the probability that a good data point from that cluster gets incorrectly labeled as outlier (because its robust weight $w_{ij} \leq e^{\frac{-t^2}{\kappa}}$) is less than $\frac{1}{t^2}$. *This constitutes an error bound on the uncertainty of detecting outliers in the data stream* [22].

The notion of an outlier can be formalized using the following definition [22]:

Definition 3.2.8. *Outlier with Chebyshev probability $\frac{1}{t^2}$: the data point x_j is an outlier with respect to cluster C_i at time n with a significance probability of $\frac{1}{t^2}$ if [22, 43]:*

$$w_{ij,n} < e^{\frac{-t^2}{\kappa}} \tag{3.19}$$

From equations (3.3) and (3.4), Definition (3.2.8) can be extended to data in temporal space using the following definition:

Definition 3.2.9. *Outlier in content and temporal spaces with Chebyshev probability $\frac{1}{t^2}$: the data point x_j is an outlier with respect to cluster C_i at time n with a significance probability of $\frac{1}{t^2}$ if:*

$$w_{ij,n} f_{ij,n} < e^{\frac{-t^2}{\kappa\rho}} \tag{3.20}$$

3.2.3 Robust Cluster merging and splitting

The detected clusters in a real data stream typically evolve over time, thus besides giving more importance to newer data points, the online clustering algorithm should detect when two or more clusters become more similar to each other in order to merge them. Similarly, a cluster can become too diffuse and split into one or more sub-clusters [22].

To handle the merging of two clusters, AFTER-Streams uses the Chebyshev bound to design a compatibility test for merging clusters C_i and C_k. This is done by checking their mutual Chebyshev bounds (i.e. testing if each cluster's centroid can be considered as an outlier with respect to the other cluster) with significance probability $\frac{1}{t^2}$: Given the distance d_{ik} between the centroids c_i and c_k, then using (3.15), the clusters are merged if none of them is found to be an outlier with respect to the other cluster, i.e. [22],

$$d_{ik}^2 < t^2\sigma_i^2 \ \& \ d_{ik}^2 < t^2\sigma_k^2 \tag{3.21}$$

which means that the centroid c_i is not an outlier, and thus is inside the influence zone of cluster C_j with significance probability equal to $1-\frac{1}{t^2}$,. The same condition applies to centroid c_j with respect to cluster C_i. Equations (3.18) and (3.21) mean that the probability of incorrectly merging clusters is less than $\frac{1}{t^2}$. *This constitutes an error bound on the uncertainty of merging clusters.*

When clusters C_i and C_k are merged, the centroid of the new cluster becomes a weighted average of the two centroids as follows [22],

$$c_{new,n} = \frac{c_{i,n}W_{i,n} + c_{k,n}W_{k,n}}{W_{i,n} + W_{k,n}}, \tag{3.22}$$

and the new scale is also a weighted average as follows,

$$\sigma^2_{new,n} = \frac{\sigma^2_{i,n} W_{i,n} + \sigma^2_{k,n} W_{k,n}}{W_{i,n} + W_{k,n}} \tag{3.23}$$

Equations (3.22) and (3.23) preserve the optimal equations for the centroid and scale respectively, given the combination of points that contributed to each cluster before merging them, with only one assumption: that the old weights of the data points in the respective clusters are not very different from the new weights in the new merged clusters, an assumption that is reasonable given the similarity between the two clusters. Similarly the temporal scales can be merged as [22]:

$$\tau^2_{new,n} = \frac{\tau^2_{i,n} W_{i,n} + \tau^2_{k,n} W_{k,n}}{W_{i,n} + W_{k,n}} \tag{3.24}$$

The new age, a_{new}, for the new cluster is set as the maximum of the ages, a_i and a_k, of the merged clusters , i.e. $a_{new} = max(a_i, a_k)$, while the new sum of weights (soft cardinality) is simply the sum of the old sum of weights of the two merged clusters C_i and C_k:, i.e. $W_{new} = W_i + W_k$.

Splitting clusters in AFTER-Streams occurs naturally and does not require any special treatment. A cluster split occurs when points from one cluster bifurcate by evolving in two or more different directions, and hence their weights with respect to the original centroid would start decreasing to the point where they start being considered outliers, which continues until they form their own new clusters [22].

3.2.4 The Complete AFTER-Streams Algorithm

Following the update equations (3.12) and (3.10), AFTER-Streams updates the cluster parameters with the arrival of a new non-outlying data point incrementally, and keeps as a summary of the data stream only the centroids ($c_{i,n}$, $c'_{i,n}$), scales ($\sigma^2_{i,n}$, $\tau^2_{i,n}$), sum of weights ($W_{i,n} = \sum_{j=1}^{n} w_{ij,n} f_{ij,n}$) and the age a_i for each cluster. Moreover, a test is added to eliminate weak clusters whose density ($\delta_{i,n}$ falls below a minimum threshold (δ_{min}) and are mature enough (i.e. $a_i > a_{mature}$). The complete steps of AFTER-Streams are listed in Algorithm 3.1.

The input parameters to AFTER-Streams include the maximum number of clusters K_{max} which is a higher bound on the allowed number of clusters and is needed to control the memory space used to store the cluster models, the initial scale σ_0 which is assigned to the newly created cluster, the density threshold δ_{min} which is used to ensure that only good clusters with high density are kept, the maturity age a_{mature} which provides a newly created cluster with a grace period before testing its density quality. An initial temporal scale τ_0, if a dynamic temporal decay is employed, else the time decay τ which sets a constant forgetting factor $e^{\frac{-1}{\tau}}$ is used. This controls the decay of the data points' weights over time. A Chebyshev bound constant t from equations (3.18) and (3.21) to set the significance probabilities of the test is also used.

3.2.5 Complexity

3.2.5.1 Time Complexity

For each new data point, AFTER-Streams computes the distance and the weights with respect to all the clusters in the clustering model ζ, which is done in linear steps. Since the clustering model is updated incrementally, nothing is recomputed from scratch, and hence the computational complexity of AFTER-Streams is $O(NK^2)$ where N is the size of the data stream and K is the highest number of clusters throughout the stream clustering (which is a very small value compared to the size of the data stream, N). Note that the K^2 term is due to the pairwise-cluster compatibility tests for merging, and could be reduced to K by performing these pairwise tests only after every K data points have arrived instead of after each data point. Another way to reduce the complexity is by checking the pairwise compatibility only within local neighborhoods confined to surroundings of the cluster in which the current data point has landed.

3.2.5.2 Memory Requirements

The memory requirements of AFTER-Streams are linear with the number of clusters, because at any point in time, only the cluster model properties $(c_i, \sigma_i, \tau_i, W_i, a_i)$ are kept in

Algorithm 3.1 AFTER-Streams

Input: Maximum number of clusters (K_{max}), Initial scales (σ_0, τ_0), density threshold (δ_{min}), maturity age (a_{mature})

Output: Cluster model after n points $\zeta = C_1 \cup C_2.... \cup C_K$, where $C_i = (c_{i,n}, c'_{i,n}, \sigma^2_{i,n}, \tau^2_{i,n}, a_i, W_{i,n})$

$K = 0$
for $n = 1$ to K **do**
  Compute the distances: d^2_{in}, η^2_{in}, and robust weights: $w_{in,n}$, $f_{in,n}$ between x_n and clusters $C_i, \forall i = 1, .., K$ {single pass over the data stream of size N}
  if $K < K_{max}$ And x_n is an outlier with respect to all clusters in ζ (Definition 4) **then**
    $K = K + 1$ {Create a new cluster centered on x_n}
    $c_K = x_n$ {centroid}
    $\sigma_K = \sigma_0$ and $\tau_K = \tau_0$ {initial scales}
    $a_K = 0$ {initial age}
    $W_K = 1$ {initial sum of robust weights}
    $\delta_K = \frac{1}{\sigma_0^2 \tau_0^2}$ {initial density}
  end if
  for all Clusters (C_i, where $i = 1, .., K$) **do**
    if x_n is NOT an outlier with respect to cluster i **then**
      Update $c_{i,n}$ using equation (3.10)
      Update $c'_{i,n}$ as: $c'_{i,n} = \frac{c'_{i,n-1} W_{i,n-1} + w_{in,n} f_{in,n} t_n}{W_{i,n-1} + w_{in,n} f_{in,n}}$
      Update $\sigma^2_{i,n}$ using equation (3.12)
      Update $\tau^2_{i,n}$ as: $\tau^2_{i,n} = \frac{\rho\left(\tau^2_{i,n-1} W_{i,n-1}\right) + w_{in,n} f_{in,n} \eta^2_{ij}}{\rho(W_{i,n-1} + w_{in,n} f_{in,n})}$
      Update sum of weights using equation (3.5)
      Update density using equation (3.8)
    end if
    Update age $a_i = a_i + 1$
  end for
  for all Pairs of clusters $C_i \& C_k$, where $i, k = 1, .., K$ **do**
    if C_i and C_k are Chebyshev-compatible using equation (3.21) **then**
      Merge clusters C_i and C_k using equations (3.22) and (3.23)
    end if
  end for
  for all Clusters (C_i, where $i = 1, .., K$) **do**
    if $(a_i > a_{mature}) \,\&\, (\delta_i < \delta_{min})$ **then**
      $\zeta = \zeta - C_i$ {remove mature clusters that have low density}
      $K = K - 1$
    end if
  end for
end for

addition to the most recent data point. The memory requirements at time n (i.e. after the arrival of n data points in the stream) can be written as

$$M(n) = (4 + d) \times B \times K_n, \tag{3.25}$$

where B is the number of bytes needed to store one value (for simplicity, we assume that all cluster model properties are stored using the same number of bytes), d is the number of dimensions in the data, and K_n is the number of clusters at time n. The first term in (3.25) consists of the three scalar values (σ_i, W_i, a_i) and the d dimensions of each centroid vector c_i. The maximum memory requirements are therefore controlled by the number of clusters, K_n, at time stamp n.

Since the maximum number of clusters is limited by K_{max}, $K_n \leq K_{max}$, hence

$$M(n) \leq (4 + d) \times B \times K_{max}, \tag{3.26}$$

This gives an upper bound on the memory requirements at any point throughout the stream, and can be configured depending on the application. Furthermore, a limit on the number of time stamps a valid cluster is stored can be used to further reduce the memory requirements.

Theorem 3.2.5. *For a temporal aware cluster C_i (with dynamic temporal decay), which was valid at time step t, it is stored for a maximum of m time steps, where m is bounded as:*

$$m > a_{mature} \ \textit{and} \ \delta_{i,m} \leq \delta_{min} \tag{3.27}$$

When using a dynamic temporal decay, the temporal information is incorporated into the density function (see Equation (3.8)). A valid cluster may remain valid when the age of the cluster C_i at time step m, is less than the maturity age a_{mature}. If the cluster age exceeds the maturity age, the cluster is removed if the density of the cluster $\delta_{i,m} \leq \delta_{min}$.

Theorem 3.2.6. *For a cluster C_i (with constant forgetting factor τ), which was valid at time step t, it is stored for a maximum of m time steps if it was inactive, where m is bounded as:*

$$m > -\tau\, ln(\frac{\delta_{min}}{\delta_{i,t}}) \tag{3.28}$$

Proof. The proof can be found in the Appendix Theorem (A.1.3). □

This sets a horizon of life on any valid cluster that depends not only on δ_{min}, but also on the initial strength $\delta_{i,t}$ of the cluster before new data stopped arriving to the cluster, and depending on the decay rate τ. Depending on the application, one may want to set $m \in [m_{min}, m_{max}]$, which makes it possible to choose the values of δ_{min} and τ to control the survival horizon limits for a cluster. Indirectly, this can also control the maximum capacity of the clustering model, and thus the memory requirements. A small m leans toward newer clusters, and will quickly eliminate older ones.

3.2.6 The objective function as a robust M-estimator and W-estimator

The objective function in (3.7) can be written as an M-estimator (See (2.11)), as follows, where for simplicity, we ignore the decay term.

$$\max_{c_i,\sigma_i^2} \left\{ J = \sum_{j=1}^{N} \sum_{i=1}^{K} \rho(r_{ij}^2; c_i, \sigma_i^2) \right\} \tag{3.29}$$

where

$$r_{ij}^2 = \frac{-d_{ij}^2}{\kappa\sigma_i^2} = \frac{-(x_j - c_i)^t(x_j - c_i)}{\kappa\sigma_i^2} \tag{3.30}$$

$$\rho(r_{ij}^2; c_i, \sigma_i^2) = \frac{e^{\frac{-d_{ij}^2}{\kappa\sigma_i^2}}}{\sigma_i^2} \tag{3.31}$$

The parameters can be estimated by seeking the null points of the gradient as follows

$$\frac{\partial J}{\partial c_i} = \sum_{j=1}^{N} \psi(r_{ij}^2, c_i, \sigma_i^2) = 0 \tag{3.32}$$

where

$$\begin{aligned}
\psi(r_{ij}^2, c_i, \sigma_i^2) &= \frac{\partial \rho(r_{ij}^2, c_i, \sigma_i^2)}{\partial c_i} \\
&= \frac{\partial \rho(r_{ij}^2, c_i, \sigma_i^2)}{\partial r_{ij}} \times \frac{\partial r_{ij}}{\partial c_i} \\
&= \frac{-2r_{ij}}{\kappa \sigma_i^2} \times \frac{e^{-r_{ij}^2}}{\sigma_i^2} \times \frac{-1}{\sigma_i^2} \\
&= constant \times r_{ij} \times e^{-r_{ij}^2}
\end{aligned} \tag{3.33}$$

We can further interpret the objective function as a W-estimator by extracting the robust weight, in a similar fashion to (2.14)

$$\begin{aligned}
w(r_{ij}) &= \frac{\psi(r_{ij})}{r_{ij}} \\
&= constant \times e^{-r_{ij}^2}
\end{aligned} \tag{3.34}$$

which has the form of a Welsh estimator (see Table 2.1) [44].

Theoretical Resistance Properties

Having established that the clustering process performs robust estimation through the stream on multiple clusters, we conclude that the cluster centroid estimation benefits from the same advantages as the Welsch estimator in terms of its resistance to outliers. The Influence Curve (IC) [42] approach can be used to further illustrate resistance to outliers. The Influence curve (IC) tells us how an infinitesimal proportion of contamination affects the estimate in large samples, and has the same shape as the ψ-function (i.e.$IC = (constant \times r_{ij} \times e^{-r_{ij}^2})$ which is shown in Figure 3.2. This curve summarizes the influence of data points with given residuals on the resulting estimate. It can be inferred from IC that the influence is asymptotically zero at locations corresponding to infinite residuals, meaning that gross outliers have almost no effect on the estimate. Also, most importantly, at any point, the influence is bounded. This constitutes the most important resistance property of any robust estimator.

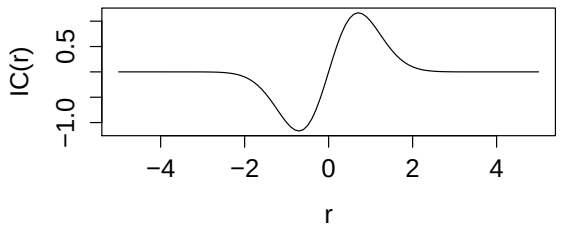


Figure 3.2: Influence Curve

3.2.7 Compliance with data stream clustering requirements

Below, we show how AFTER-Streams meets all the requirements of data stream clustering algorithms.

1. **Compactness of representation:**

   Each cluster is represented using four components: The centroid c_i in content space which is a vector of size equal to the dimensionality of data and the temporal centroid c_i', the scales σ_i and τ_i, the sum of weights W_i and the age a_i. Hence, the total memory requirement of the clustering model is given in (3.25), which is a very compact representation compared to the original data stream (Section 3.2.5.2).

2. **Fast processing of new data points:**

   The second requirement is also met since each new data point is compared against the existing clusters in a linear time $O(K)$, and is thereafter discarded (Section 3.2.5.1).

3. **Fast handling of outliers**:

   If the data point was determined to be an outlier (see Definition 7), then it is used to create a new cluster and this verification is also done in linear time with the number of clusters (Section 3.2.5.1).

4. **Integration of offline and online data:**

   The fourth requirement suggests the ability to store the clustering model offline and to access it easily, which is also met in AFTER-Streams because the clustering model

is very compact and can be stored in main memory. However, if needed, the clustering model at different time steps can also be stored offline in secondary memory, and can be easily accessed.

5. **Making no assumptions about the number of clusters:**

   AFTER-Streams does not assume a number of clusters in advance. Instead, it only requires a maximum number of clusters which is also used to control the memory space used (Section 3.2.5.2).

6. **Handling evolution:**

   AFTER-Streams handles the evolution of data by the very definition of its *dynamic* robust weights in (3.3), which uses a dynamic forgetting mechanism to give more emphasis to the newer data points, thus allowing adaptation to the changing nature of the data stream, and allowing old inactive clusters to disappear and new clusters to emerge.

3.2.8 Comparison with related work

Table (3.1) compares the proposed online clustering algorithm, AFTER-Streams, compared to some of the competing algorithms as discussed in Section 2.1, while Table (3.2b) explains the meaning of the symbols used in Table (3.1). The following conclusions can be drawn from the comparison:

- All algorithms provide an explicit way to detect outliers with the exception of CluStream, since it maintains a constant number of micro-clusters. Hence, it might mislabel outliers as valid micro-clusters.

- Both AFTER-Streams and TRAC-Streams have a low complexity (which depends on the number of points (N) as well the number of clusters (K)). On the other hand, the remaining algorithms have a complexity that includes the number of data points (N) as well as the number of micro-clusters (M_C), which is typically a much higher value than the number of regular clusters (K).

TABLE 3.1: Comparison between AFTER-Streams and most related stream clustering algorithms

(a) Comparison

Property	**RINO-Streams**	**DenStream**	**CluStream**	**TRAC-Streams**	**StreamKM++**	**AFTER-Streams**
Detecting Outliers	✓	✓	×	✓	×	✓
Complexity of online clustering	$O(NK^2)$	$O(NM_C)$	$O(NM_C^2)$	$O(NK^2)$	$O(dNm)$	$O(NK^2)$
Complexity of offline clustering	×	$O(M_C^2)$	$O(KM_Cd)$	×	×	×
Requiring specification of No. Clusters	×	×	✓	×	×	×
Requiring an offline step	×	yes (upon request)	yes (upon request)	×	×	×
Memory requirements	$(1d+3)*K$	$(2d+1)*M_C$	$S*M_C*(2d+3)$	$(1d+3)*K$	$\Theta(dm\log(N/m))$	$(1d+4)*K$
Needing special Initialization (typically Batch clustering)	×	Run DBSCAN on initial set of pts.	Run K-Means on initial set of pts.	×	Needs initialization via coreset size.	×
Built-in robustness to noise	✓	×	×	✓	×	✓
Built-in forgetting mechanism for old data	✓	✓	×	✓	×	✓
Adaptive Forgetting	×	×	×	×	×	✓
Reference	[22]	[86]	[87]	[43]	[88]	Our Proposed Algorithm

Symbol	**Meaning**
K	No. of clusters
N	No. of data points
M_C	No. of micro-clusters (or grids in the case of D-Stream) ($M_C \ggg K$)
S	No. of snapshots
d	No. of dimensions
m	No. of data points in the coreset

(b) Symbol meanings

- CluStream is the only algorithm that requires specifying the number of clusters in advance.
- The space costs of each cluster in AFTER-Streams ($1d + 4$) is small compared to the other algorithms. Moreover, the number of cluster they maintain (C) is much smaller than the number of micro-clusters (M_C) which is maintained by the rest of the algorithms. Hence, the memory requirement for AFTER-Streams and TRAC-Streams is small.
- AFTER-Streams and TRAC-Streams are less sensitive to the model assumption and data distributions, thanks to the use of robust statistics in their objective functions

via the robust weights.

- AFTER-Streams is a direct descendant of RINO-Streams [22] with the addition of dynamic, cluster-specific temporal scales to adapt the forgetting to each cluster's data arrival speed.

3.3 AFTER-Streams Experiments

In this section, we evaluate the performance of AFTER-Streams. The experimental settings are presented in Section 3.3.2, then we present a sensitivity analysis of the performance of AFTER-Streams with respect to the data stream properties as well as the AFTER-Streams parameter values in Section 3.3.3. We will present thorough comparisons between AFTER-Streams, RINO-Streams [89], CluStream [87], DenStream [86] and StreamKM++ [88] on synthetic and real datasets in Sections 3.3.4 and 3.3.5, respectively. The effectiveness of handling cluster splitting and merging is tested in Section 3.3.6.

Overall, our sensitivity experiments encompass close to a 1000 settings and 8640 synthetic data streams, in addition to real data streams with the biggest data stream consisting of around 500,000 data points (KDD Cup 99). In Section 3.3.3, we present the ANOVA tables used to analyze AFTER-Streams' performance sensitivity with respect to the properties of the datasets as well as the parameter inputs.

3.3.1 Evaluation Plan

The research contributions for AFTER-Streams can be outlined as shown in Table 3.2.

TABLE 3.2

Summary of Research Goals and Evaluation Plan

Framework Component	Goal	Methodology	Research Questions	Validation Metrics	Evaluation Section	Data	Baseline
Clustering evolving data streams	Adapt to different evolving cluster data arrival speeds	Dynamic Forgetting (Chapter 3)	RQ1.1, RQ1.2, RQ1.3	Internal and External cluster validation metrics, e.g. F1-Score, DB-Index etc.	Sec 3.3	Synthetic, TREC, KDDCUP99 (Sec 3.3.2.2)	RINO-Streams [22], CluStream [87], DenStream [86], StreamKM++ [88]

3.3.1.1 Research Questions

The goal of AFTER-Streams is to cluster data streams as described in Section 3.1. Solving the following research problems will show how the AFTER-Streams is an effective algorithm for clustering data streams.

Research Question 1.1: Which parameters of the RBF data stream generator show a significant effect on AFTER-Streams performance?

Research Question 1.2: Which parameters of the AFTER-Streams algorithm show a significant effect on its performance?

Research Question 1.3: Does the AFTER-Streams algorithm perform better than the baseline algorithms.

3.3.2 Experimental Settings

3.3.2.1 Evaluating Data Stream Clustering Results

Continuously validating the clustering model in live data streams is much more challenging compared to traditional clustering models for several reasons. First, the data points are viewed only once and then discarded, and the clustering model is just a summary of those data points. Hence, trying to use traditional validation metrics (e.g. purity) would

be impossible. Second, the clustering model is evolving over time, so a cluster at time t_1 might shift to a new location at time t_2. And finally, the clustering model cannot be predicted because the data stream is infinite, hence no realistic ground truth can be created.

For these reasons, in this work, we will follow two approaches to evaluate a data stream [22]:

(i) we find the evaluation metrics at predefined periods of time, and

(ii) we propose performing the validation hand in hand with detecting the clusters, and constructing behavioral profiles of clusters over time, that provide some form of online real-time validation.

3.3.2.2 Datasets

Synthetic Data Streams To emulate an infinite data stream in a typical real world scenario, we used the random Radial Basis Function (RBF) data stream generator provided as part of the Massive Online Analysis (MOA) stream data benchmarking framework [1] [2]. MOA is an open source framework designed to analyze massive streams of data. Using the data stream generators allows us to control all the aspects of a data stream to mimic realistic data streams. For example, we can control the frequency of merging clusters. RBF generates a continuous data stream following a normally distributed hypersphere. The parameters that control the RBF generator and their descriptions are listed in Table 3.3.

Real Datasets We will use several real text datasets and one network activity dataset. The real text datasets are provided with the CLUTO toolkit[2] [90] and are derived from the TREC collection[3]. The KDD Cup 99 dataset[4] represents network activity traces, collected over a period of nine weeks of normal activity interspersed with various attacks and intrusions simulated in a military network environment . We used the training dataset with the 33 continuous features. There are a total of 23 different attacks that fall into four main

[1] http://moa.cms.waikato.ac.nz/
[2] http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto
[3] http://trec.nist.gov
[4] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

TABLE 3.3: RBF Data Stream Generator Parameters [2].

Parameter	Description
Stream Length	Number of data points generated in the data stream
No. Clusters	Number of random centroids
No. Dimensions	Number of Dimensions
No. Clusters Range	Deviation of the number of centroids in the model
Radii	The average radii of the centroids in the model
Density Range	Offset of the average weight a cluster has. A value of 0 means all clusters contain the same amount of points
Speed	Kernels move a predefined distance of 0.01 every X points (X is the speed)
Noise	Noise level
Event Frequency	Frequency of events taking place (i.e. merging/splitting or emerging/disappearance)

TABLE 3.4: Real Text and Network Intrusion Detection Data Set Descriptions

Dataset	Source	Num. Points	Num. Dimensions	Num. Classes	Mean Class Size	Balance
tr11	TREC	414	6424	9	46	0.0455
tr12	TREC	313	5799	8	39	0.0968
tr23	TREC	204	5831	6	34	0.0659
tr31	TREC	927	10127	7	132	0.0057
tr41	TREC	878	7453	10	87	0.037
tr45	TREC	690	8261	10	69	0.0875
KDD CUP 99	KDD CUP 99	494021	33	23	21479	7e-06

categories.

The properties of the real datasets are listed in Table 3.4. The balance is the ratio of the smallest cluster (in terms of number of points) to the largest cluster.

3.3.2.3 Evaluation Metrics

As discussed in Section 3.3.2.1, evaluating data stream clusters is done at predefined periods of time since the data stream keeps evolving over time, hence, the final clustering model does not necessarily represent earlier data points. Hence, we performed the evaluation every 10% of the stream length, and then we computed the average, minimum and maximum metric values.

TABLE 3.5: RBF Data Stream Generator Parameters

Parameter	Range of values
Stream Length	1000, 5000, 10000
No. Clusters	20, 60, 80, 100
No. Dimensions	2, 5, 10
No. Clusters Range	5
Radii	0.05, 0.1
Density Range	0 , 0.1
Speed	500
Noise	0, 0.1
Event Frequency	100, 500, 1000

We compared the results using the following two internal validity metrics: Silhouette index [44] and Davies-Bouldin index [91]. In addition to the internal validity metrics, since class or cluster labels are provided with the data sets, we computed four external validity metrics: V-measure [92], Fowlkes-Mallows Index [93], Recall [94] and F1 Score [94].

3.3.2.4 Experimental Setup

For each of the baseline algorithms, we varied the parameter values and found the best results for each dataset, then we calculated the average performance over all the datasets. To ensure a fair and realistic comparison, we initialized the algorithms (with the exception of AFTER-Streams since it is not required) with the first data points, i.e. the centroids of the first K clusters were set to the values of the first K data points.

AFTER-Streams The parameters, along with their descriptions and values, are shown in Table 3.6. There is a total of 192 parameter settings.

Baseline Algorithm: RINO-Streams The parameters for RINO-Streams [22] are exactly the same as AFTER-Streams, except that the initial temporal scale τ_0 is considered as the forgetting lifetime $\frac{\tau}{|X|}$. The parameters, along with their descriptions and values, are shown in Table 3.6. There is a total of 192 parameter settings.

TABLE 3.6: AFTER-Streams Parameter Values

Parameter	Description	Values
$\frac{K_{max}}{K_G}$	The maximum number of clusters allowed (K_{max}) as a percentage of the real number of clusters in the ground truth (K_G).	50%, 100%, 150%
σ_0	Initial Scale	0.05, 0.1
τ_0	Intial Temporal Scale (or optional forgetting lifetime (τ) as a fraction of the data stream length $\|X\|$)	0.01 , 0.1, 0.2
$\frac{1}{t_{outlier}}$	Chebyshev constant for outlier detection	0.05, 0.1
$\frac{1}{t_{merging}}$	Chebyshev constant for the cluster merging test	0.05, 0.1
$\frac{a_{mature}}{\|X\|}$	The maturity age (a_{mature}) as a percentage of the data stream length ($\|X\|$)	1%, 2%

TABLE 3.7: CluStream Parameters

Parameter	Description	Values
$\frac{K_{macro}}{K_G}$	The number of macro clusters (K_{macro}) as a percentage of the real number of clusters in the ground-truth (K_G).	50%, 100%, 150%
$\frac{K_{micro} \times K_{macro}}{K_G}$	The number of micro clusters (K_{micro}) as a percentage of the number of macro clusters (K_{macro})	50%, 100%
$t_{recency}$	Threshold used to delete micro clusters when a new micro cluster is created	10, 20

Baseline Algorithm: CluStream The implementation of CluStream [87], provided by the MOA framework [2], was used in these experiments. CluStream incrementally updates a set of micro-clusters and generates the final clusters (i.e. macro-clusters) using K-means. The number of micro-clusters is usually higher than the final number of generated macro-clusters. In contrast, AFTER-Streams and RINO-Streams incrementally maintain the final clusters, which are considered the equivalent of CluStream's resulting macro clusters. Hence, to make a fair comparison, the evaluation metrics for CluStream are generated from the macro-clusters that are found using K-means at the end of each evaluation time period. The CluStream parameters, along with their description and values, are shown in Table 3.7

TABLE 3.8: DenStream Parameters

Parameter	Description	Values
ϵ	The epsilon neighborhood which is the maximal radius of micro-clusters.	0.02, 0.1
μ	The minimum points with which a core-micro-cluster needs to be created.	1, 10
β	A multiplier for μ to detect outlier micro-clusters.	0.05, 0.2
λ	The decay constant.	0.2, 0.5
$\frac{i}{\lvert X\rvert}$	The number of points to use for initialization as a percentage of the data stream length ($\lvert X\rvert$).	0.5%, 1%
X_{speed}	The number of incoming points per time unit.	1, 1000

TABLE 3.9: StreamKM++ Parameters

Parameter	Description	Values
$\frac{K}{K_G}$	The number of clusters (K) as a percentage of the real number of clusters in the ground truth (K_G).	50%, 100%, 150%
$\frac{s}{\lvert X\rvert}$	The size of the coreset as a percentage of the data stream length ($\lvert X\rvert$).	5%, 10%, 20%

Baseline Algorithm: DenStream The implementation of DenStream [86], provided by the MOA framework [2], was used in these experiments. DenStream requires multiple input parameters: the range of the window, epsilon neighborhood which is the maximal radius of micro-clusters, minimum points core-micro-cluster needs to be created with, decay constant, processing speed etc. These parameters, along with their description and values, are shown in Table 3.8.

Baseline Algorithm: StreamKM++ The implementation of StreamKM++ [88], provided by the MOA framework [2], was used in these experiments. StreamKM++ requires two input parameters: the number of clusters and size of coreset. These parameters, along with their description and values, are shown in Table 3.9.

3.3.3 Sensitivity Analysis

In this section, we will analyze AFTER-Streams' performance sensitivity with respect to the properties of the datasets as well as the parameter inputs to AFTER-Streams. We will use the synthetic datasets generated by the RBF data stream generator. The values of the different parameters controlled by the RBF generator are shown in Table 3.5. There are a total of 864 different experimental settings, and for each one of them, we generated 10 datasets. Hence, we have 8640 different datasets. We used the values in Table 3.6 for AFTER-Streams which resulted in 144 different settings. For each of the datasets we found the average over all 144 different AFTER-Stream settings, and then we found the average for every 10 datasets that have the same RBF data stream generator settings. To evaluate the quality of the clusters, we calculated the Davies-Bouldin index, Silhouette score and other metrics as shown in Figure 3.4.

3.3.3.1 Sensitivity based on the data stream properties

In this section, we will evaluate how AFTER-Streams behaves under different data stream conditions. We analyzed the results using an analysis of variance test (ANOVA) with the hypothesis that the RBF data stream generator parameters do not affect the quality of AFTER-Streams.

Hypothesis 1.1: From RQ1.1 (Sec 3.3.1.1), the RBF stream parameters do not have a significant effect on the quality of the clusters generated by AFTER-Streams.

If the *p-value* is less than α (i.e. 0.05) then the factor (i.e. one of the RBF stream parameters) has a significant effect on the quality of the clusters generated by AFTER-Streams.

The results of performing ANOVA with $\alpha = 0.05$ are shown in Table (3.11) for Davies-Bouldin, Silhouette index, and V-Measure (all rounded to two decimal points). The results show that the density range and the event frequency do not have a significant effect on the performance of AFTER-Streams based on all three quality measures. Moreover, the

radii of the clusters and the dimensionality of the data stream do not affect the Davies-Bouldin index. The rest of the RBF stream parameters have a significant effect on AFTER-Streams.

To further analyze the effect of the data stream properties on the performance of AFTER-Streams, we plotted the cluster quality, reflected by the metrics (as shown in Figure 3.4) versus the different values of seven of the parameters controlling the generation of the continuous RBF stream. The results can be summarized as follows:

- Density Range and Event Frequency do not significantly affect the quality of the clusters. Only Davies-Bouldin metric seems be have been affected by Density Range.
- The dimensionality and the cluster radius has seem to be having an affect on the metrics.
- The rest of the RBF stream parameters also affect the quality of the clusters generated by AFTER-Streams.

3.3.3.2 Sensitivity based on AFTER-Streams parameters

In this section, we will analyze the effect of using different parameter values for AFTER-Streams on the quality of the clustering models.

ANOVA

We analyzed the results using the analysis of variance test (ANOVA) with the hypothesis that AFTER-Streams' parameter settings do not affect the quality of the clustering model.

Hypothesis 1.2: From RQ1.2 (Sec 3.3.1.1), the AFTER-Streams parameters do not have a significant effect on the quality of the clusters generated.

If the *p-value* is less than α (i.e. 0.05) then the factor (i.e. one of the AFTER-Streams parameters) has a significant effect on the quality of the clusters generated.

The results are presented in Table (3.15). The results show that most AFTER-Streams' parameters have a significant effect on the quality of the clustering model. Only a few of the parameters show no significant effect on some of the quality metrics, namely, t_{merge}, $t_{outlier}$ and a_{mature}. The results can be summarized as follows:

- The Chebyshev constant (t_{merge}), used for merging (Section 3.2.3), does not have a significant effect on Davies-Bouldin and Silhouette scores.
- The Chebyshev constant ($t_{outlier}$), used to detect outliers (Section 3.2.2), and the minimum sum of weights (W_{min}) are inversely proportional to the quality of the clustering model.
- When using dynamic temporal scale, there is not significant effect of the initial temporal scale on the cluster quality as the clustering model automatically adjusts to the temporal distribution of the data points. RINO-Streams' forgetting factor (τ), affects the speed of decay of the data point weight (Section 3.2), is directly proportional to the quality of the clustering model. A lower value, emphasizing only newer clusters, generally leads to lower overall quality.
- The initial scale (σ_0) has a directly proportional to the quality of the clustering model based on Davies-Bouldin and Silhouette Index.
- The maturity age (a_{mature}), which provides a grace period for outliers (Section 3.2.2), shows no significant effect on the quality metrics.
- The maximum number of clusters allowed has a significant effect, it generates best results if it was closer to the real number of clusters.

Pareto Frontier

An important and difficult problem in data mining in general, is to find the best parameter values to maximize the quality of the model. However, analyzing the effect of the AFTER-Streams parameters on the quality of the clustering model showed that different

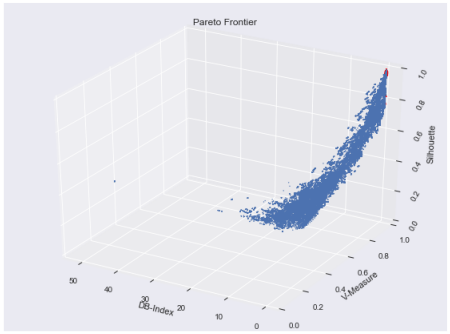


Figure 3.3: AFTER-Streams: Pareto Frontier

TABLE 3.10: MOA RBF Stream Generator Parameters' Significance p-values (Synthetic Data)

Parameter	F1-Score	Fowlkes-Mallows	Recall	V-Measure	DB-Index	Silhouette
Event Frequency	0.877	0.855	0.877	0.911	0.973	0.894
No. Clusters	0.000	0.000	0.000	0.000	0.000	0.000
Noise	0.000	0.000	0.000	0.000	0.000	0.000
Radii	0.000	0.000	0.000	0.000	0.000	0.000
No. Dimensions	0.000	0.000	0.000	0.000	0.000	0.000
Density Range	0.707	0.697	0.707	0.972	0.000	0.455
Random Seed	0.000	0.000	0.000	0.000	0.000	0.000

values can improve or reduce the quality of the generated clusters. Some, like maximum number of allowed clusters, can even have different effect on different measures of quality. Hence, there are trade-offs that we need to consider when choosing the parameter values.

To solve this problem, we perform a Pareto efficiency analysis [95]. Pareto efficiency deals with the problem of trade-offs between multiple solutions to a problem and it selects a set of *efficient* solutions which can not be further improved. These are called the Pareto Frontier and are shown in Figure 3.3 as red points. Analyzing the Pareto Frontier points shows that some parameters generate the best results when they are set to a specific value (e.g. $\frac{K_{max}}{K_G}$) while other parameters may need a specific combination with other parameters (e.g. σ_0).

TABLE 3.11: MOA RBF Stream Generator Parameters' Effect Size (Synthetic Data)

Parameter	F1-Score	Fowlkes-Mallows	Recall	V-Measure	DB-Index	Silhouette
Event Frequency	0.003	0.003	0.003	0.003	0.002	0.003
No. Clusters	0.689	0.726	0.689	0.702	0.208	0.718
Noise	0.032	0.041	0.032	0.103	0.125	0.091
Radii	0.23	0.227	0.23	0.259	0.288	0.294
No. Dimensions	6.645	5.808	6.645	4.403	2.077	4.388
Density Range	0.002	0.002	0.002	0.004	0.028	0.0
Random Seed	0.215	0.2	0.215	0.311	0.391	0.112

TABLE 3.12: AFTER-Streams compared to RINO-Streams, CluStream, DenStream, StreamKM++ (Synthetic Data, p-values)

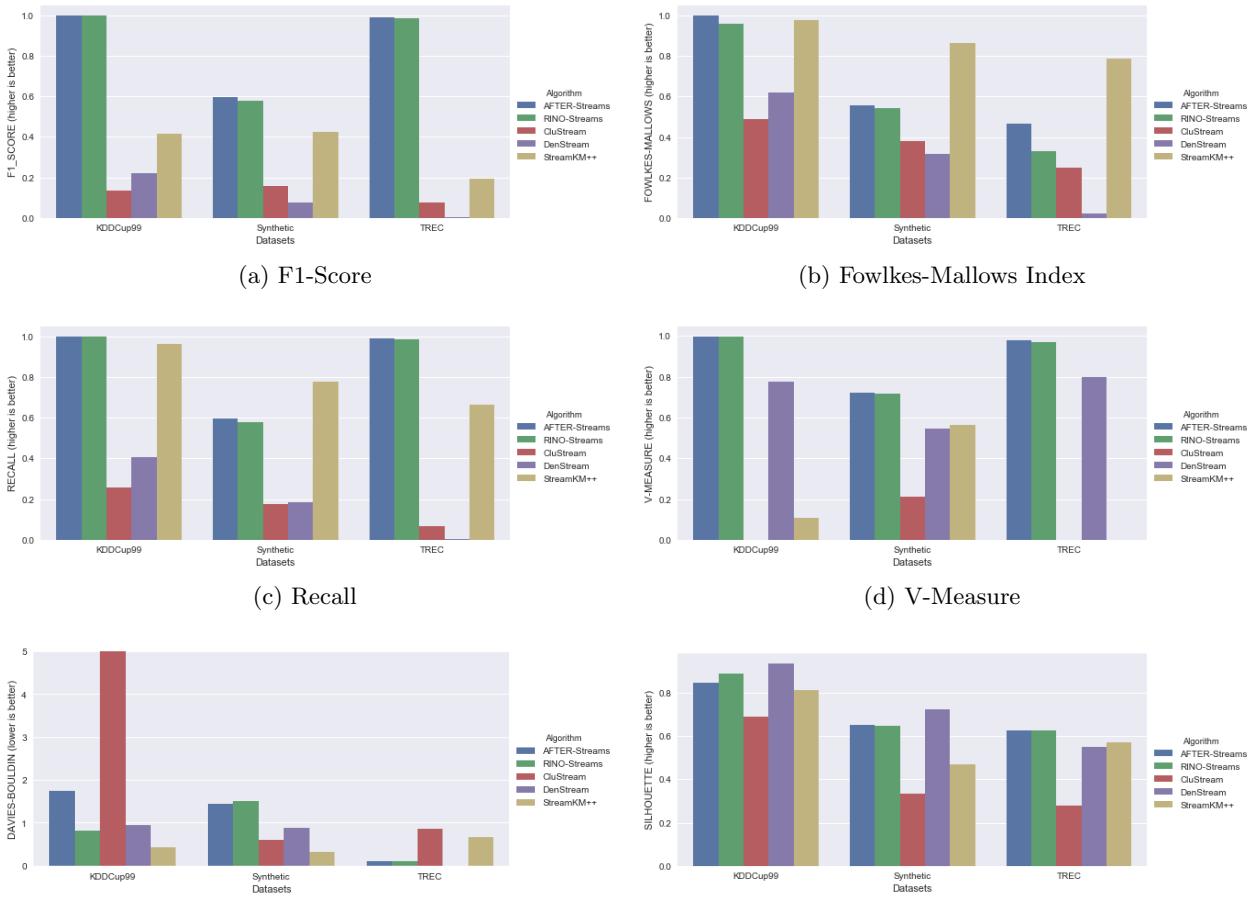
AFTER-Streams vs	F1-Score	Fowlkes-Mallows	Recall	V-Measure	DB-Index	Silhouette
RINOStreams	0.000	0.000	0.000	0.000	0.000	0.000
CluStream	0.000	0.000	0.000	0.000	0.000	0.000
DenStream	0.000	0.000	0.000	0.000	0.000	0.000
StreamKM++	0.000	0.000	0.000	0.000	0.000	0.000

TABLE 3.13: AFTER-Streams compared to RINO-Streams, CluStream, DenStream, StreamKM++ (Synthetic Data, Effect Size)

AFTER-Streams vs	F1-Score	Fowlkes-Mallows	Recall	V-Measure	DB-Index	Silhouette
RINOStreams	0.045	0.029	0.045	0.024	-0.047	0.022
CluStream	1.587	0.592	1.457	2.344	0.253	1.405
DenStream	1.861	0.711	1.248	0.693	0.470	-0.336
StreamKM++	0.585	-1.050	-0.570	0.653	0.953	0.764

TABLE 3.14: AFTER-Streams Parameters' Significance p-values (Synthetic Data)

Parameter	F1-Score	Fowlkes-Mallows	Recall	V-Measure	DB-Index	Silhouette
K_{max}	0.000	0.000	0.000	0.000	0.000	0.000
σ_0	0.000	0.000	0.000	0.000	0.000	0.000
τ_0	0.695	0.322	0.695	0.020	0.705	0.169
$\frac{1}{t_{outlier}}$	0.000	0.000	0.000	0.000	0.000	0.000
$\frac{1}{t_{merging}}$	0.000	0.000	0.000	0.000	0.424	0.413
a_{mature}	0.938	0.912	0.938	0.899	0.206	0.854



(a) F1-Score

(b) Fowlkes-Mallows Index

(c) Recall

(d) V-Measure

(e) Davies-Bouldin Index

(f) Silhouette Score

Figure 3.4: AFTER-Streams compared to RINO-Streams, CluStream, DenStream, StreamKM++

Default Parameter Values

Based on the ANOVA, sensitivity analysis and the Pareto Frontier results discussed above, we will use the parameter values in Table 3.16, unless stated otherwise, for AFTER-Streams parameters in the next experiments.

3.3.4 AFTER-Streams Performance (Synthetic Data Streams)

In this section, we present the results obtained from comparing AFTER-Streams with RINO-Streams, CluStream, DenStream, and StreamKM++ on the synthetic datasets generated by the RBF stream generator of the MOA framework.

TABLE 3.15: AFTER-Streams Parameters' Effect Size (Synthetic Data)

Parameter	F1-Score	Fowlkes-Mallows	Recall	V-Measure	DB-Index	Silhouette
K_{max}	0.184	0.254	0.184	0.127	0.429	0.364
σ_0	0.577	0.583	0.577	0.761	0.459	0.312
τ_0	0.005	0.009	0.005	0.017	0.012	0.006
$\frac{1}{t_{outlier}}$	0.159	0.111	0.159	0.246	0.067	0.045
$\frac{1}{t_{merging}}$	0.026	0.03	0.026	0.04	0.005	0.005
a_{mature}	0.0	0.001	0.0	0.001	0.001	0.007

TABLE 3.16: AFTER-Streams Default Parameter Values, resulting from sensitivity and Pareto efficiency analysis

Parameter	Default Value Used
$\frac{K_{max}}{K_G}$	100%
σ_0	0.05
τ_0	0.2
$\frac{1}{t_{outlier}}$	0.05
$\frac{1}{t_{merge}}$	0.1
$\frac{a_{mature}}{\|X\|}$	2%

Hypothesis 1.3: From RQ1.3 (Sec 3.3.1.1), clustering the data stream with AFTER-Streams produces clusters with the same quality as the baseline algorithms.

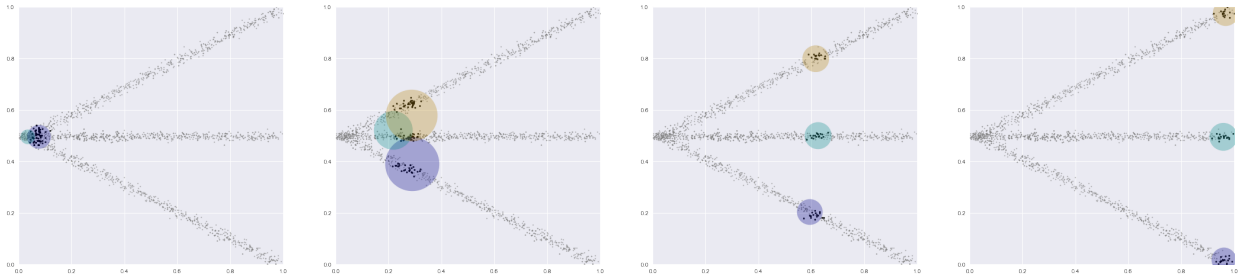
If the *p-value* is less than α (i.e. 0.05) then the quality of clusters generated by AFTER-Streams algorithm is significantly different from the quality of the clusters generated by the baseline algorithms. The results are summarized in the following sections.

3.3.4.1 Overall Performance

Figure 3.4 shows the results of the internal and external validity metrics. The statistical significance of the results is validated by finding the p-value as shown in Table

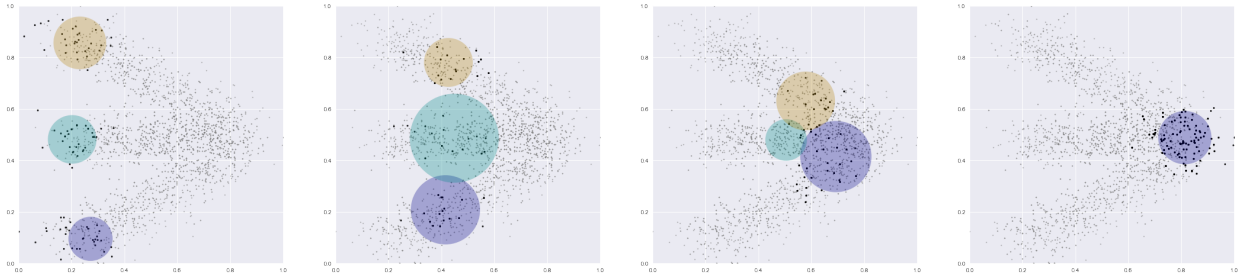
TABLE 3.17: Parameter configurations for cluster splitting and merging

Parameter	K_{max}	τ	σ_0	$\frac{1}{t^2}$	a_{mature}
Merging	15	5% of $\|X\|$	0.1	0.075	50
Splitting	15	2% of $\|X\|$	0.1	0.075	50



(a) At time=10% of $|X|$ (b) At time=30% of $|X|$ (c) At time=60% of $|X|$ (d) At time=100% of $|X|$

Figure 3.5: A cluster that gradually splits into three clusters over time



(a) At time=10% of —X— (b) At time=30% of —X— (c) At time=70% of —X— (d) At time=100% of —X—

Figure 3.6: AFTER-Streams: Three clusters that gradually merge into one cluster over time

3.13. The results show that AFTER-Streams and RINO-Streams outperform the benchmark algorithms in most cases. For F1-score and V-Measure, AFTER-Streams outperforms all the other benchmark algorithms. For Fowlkes-Mallows index and Recall, AFTER-Streams seems to be performing better than CluStream and DenStream. For Silhouette score, AFTER-Streams seems to be performing better than CluStream and StreamKM++.

3.3.4.2 Performance with respect to Stream Properties

We compared the performance of AFTER-Stream to that of RINO-Streams, CluStream, DenStream, and StreamKM++ with respect to some of the synthetic stream datasets properties (Table 3.3). Figure 3.4 show the validity metric values of the algorithms when varying the number of clusters, number of dimensions, noise level and stream length of the generated stream datasets.

3.3.5 AFTER-Streams Performance (Real Data Text and Intrusion Detection Data Sets)

In this section, we compare the performance of AFTER-Streams against RINO-Streams, CluStream, DenStream, and StreamKM++ using the 6 real text datasets (TREC) and network activity dataset (KDD CUP 99) listed in Table 3.4. For each of the experiments, we show the best results obtained from varying the parameter values for each algorithm as well as the significance of the difference using their p-values. The validation metrics are the same as the ones used in the previous section for the MOA synthetic stream data.

3.3.5.1 Results for TREC Text Datasets

Figures 3.4 show the best validity metric values, obtained from all the parameter configurations for the F1-score, V-Measure, cluster recall, Fowlkes-Mallows score, Davies-Bouldin index, Silhouette score etc.

The results show that AFTER-Streams (and RINO-Streams) significantly outperforms CluStream, DenStream, and StreamKM++ for the TREC datasets, except for Fowlkes-Mallows score, where StreamKM++ seems to be having a higher value.

3.3.5.2 Results for KDD CUP 99 Network Intrusion Data

Figure 3.4 shows the external validity metrics for AFTER-Streams, RINO-Streams, CluStream, DenStream, and StreamKM++. AFTER-Streams performs better for F1-Score, Fowlkes-Mallows index, Recall, V-Measure. StreamKM++ seems be performing better for Davies-Bouldin, and DenStreams seems to be having better Sihouette values.

3.3.6 Validating Cluster splitting and merging

To illustrate how clusters merge and split in AFTER-Streams, we designed two experiments where one cluster evolves into three different clusters to show cluster splitting, and one where three clusters evolve into one cluster to show cluster merging. Table 3.17 lists the configuration parameter values. Figure 3.5 shows the cluster output evolution at

five different time periods, where time is measured in terms of the number of data points that arrived relative to the data stream size ($|X|$). It can be seen that one cluster (cluster number 1) is detected at the beginning, and then, as the cluster splits, two more clusters are detected. Figure 3.6 illustrates the gradual merging of three different clusters, over five different time periods, into one cluster.

3.4 Summary and Conclusions

In this chapter, we presented an extension to RINO-Streams [22], where the clustering algorithm is equipped with automatic temporal scale estimation. The new algorithm can automatically cluster the data in both content space and temporal space. Our extensive experimental results showed that AFTER-Streams outperform the benchmark algorithms on TREC, KDDCUP99, and synthetic datasets. Our sensitivity analysis experiments have also validated the role of the different parameters and their impact on performance.

CHAPTER 4

COMPARTMENTALIZED ONLINE STREAM TOPIC MODELING

In this chapter, we present a new compartmentalized topic modeling framework which uses Stream-Dashboard, topic modeling, and pseudo relevance feedback to generate topics and new seed terms to adapt the initial filter query.

4.1 Introduction

Social media presents a very challenging scenario, where millions of users continuously produce huge amounts of streaming, diverse data. Social media data can be analyzed to extract stories as they evolve [96]. To handle the volume and unsupervised nature of social media data, we have employed a combination of *single-pass* stream clustering and topic modeling techniques. In particular, we have adopted Stream-Dashboard, a stream mining framework [23], which contains two components, RINO-Streams and TRACER, because it is the only available tool that allows simultaneous mining, tracking, and validation of evolving clusters in noisy data stream.

We also exploit topic modeling techniques that can learn unsupervised models of documents and words, simultaneously, such that each document and each term can be represented as a vector of topic proportions. Extracting topics, as the stream unfolds, can help extract stories from the social media data stream. Furthermore, using the Stream-Dashboard framework in combination with Topic Modeling, we created a compartmentalized framework that can extract, track, and validate stories from social media streams. To cope with large volumes of data, we use Online LDA which is faster than Gibbs sampling based LDA [4], while also relying on Stream-Dashboard's capability to perform single-pass stream clustering to discover clusters of similar tweets and to track their evolution. As a result, the

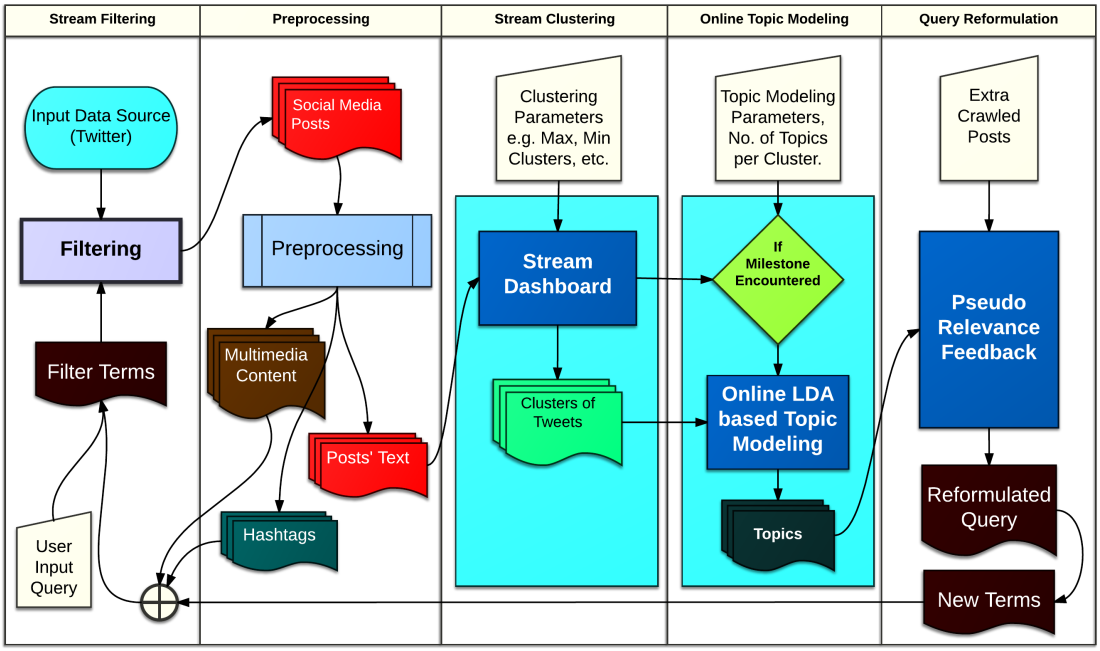


Figure 4.1: Compartmentalized Online Stream Topic Modeling Framework

compartmentalized framework extracts evolving topics on a real time basis. The following sections will discuss the proposed framework in more detail. Section 4.2 discusses the proposed framework, including online topic modeling, online joint sentiment topic modeling, the compartmentalized stream clustering and tracking, adaptive query reformulation, etc. Section 4.3 presents our evaluation results and Section 4.4 concludes the chapter.

4.2 A Compartmentalized Online Stream Topic Modeling Framework

The compartmentalized topic modeling approach handles the diversity of the data stream by partitioning the data in both content space and time space using the Stream-Dashboard framework into more homogeneous clusters. Extracting topics within specific clusters, and only at appropriate times, is expected to lead to faster, easier and better topic extraction compared to tracking, the entire data stream. Figure 4.1 shows the flow of the complete compartmentalized online stream topic modeling framework. The compartmentalized framework involves components to preprocess the social media posts, Stream-Dashboard-based clustering and milestone detection, online topic modeling, and finally

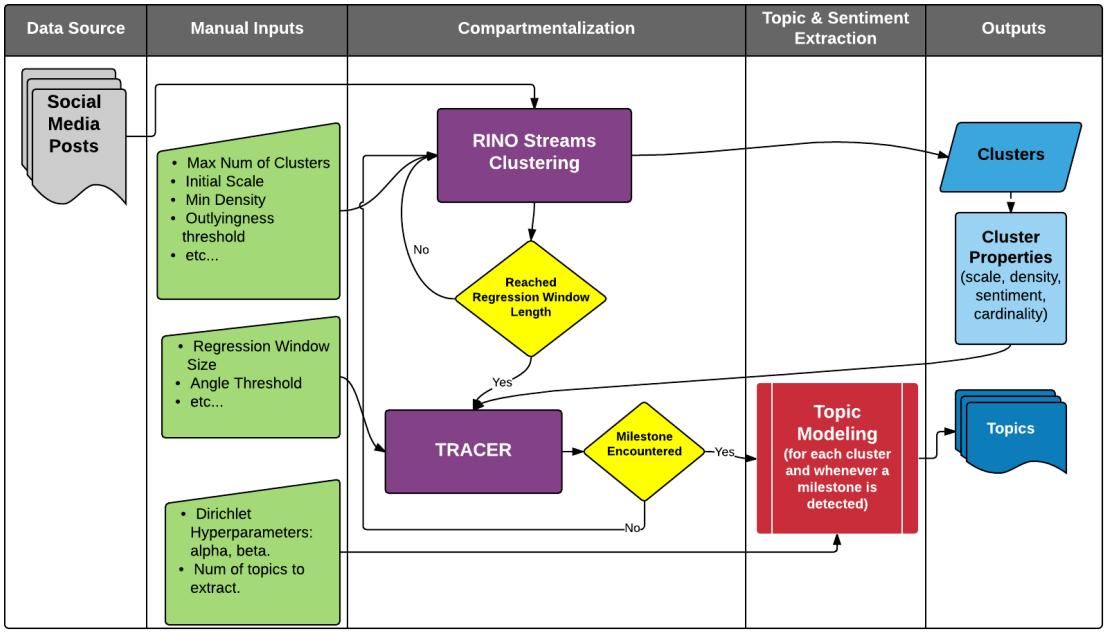


Figure 4.2: Stream-Dashboard Module for Compartmentalized Online Stream Topic Modeling

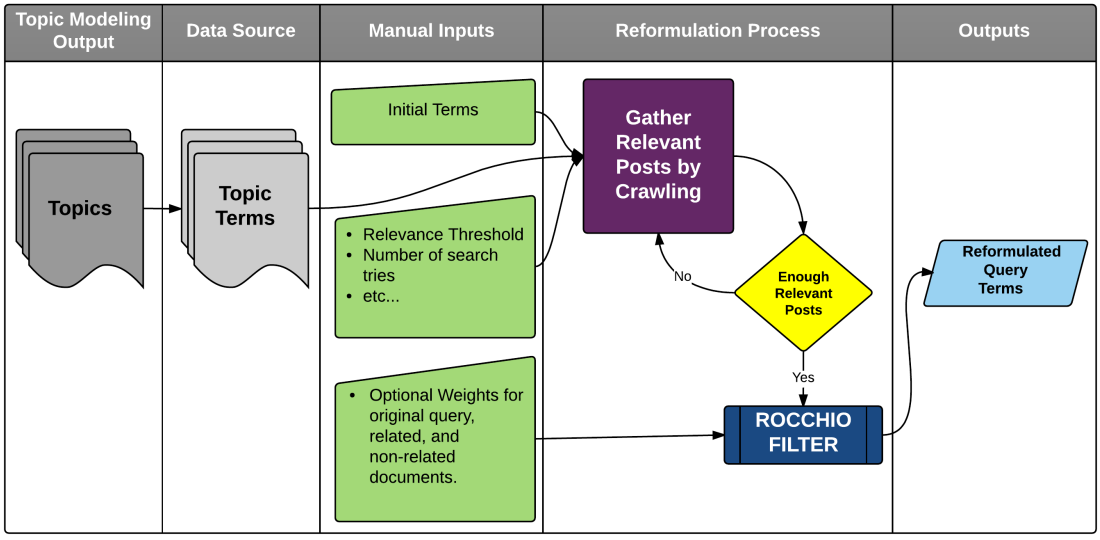


Figure 4.3: Relevance Feedback for Compartmentalized Online Stream Topic Modeling

query reformulation to help adapt the filtering of the input data stream to a particular user need or task.

4.2.1 Data Stream Source Filter

The Twitter micro-blog post stream source components play a vital role in choosing the data that is passed on to the succeeding stages of mining. The Stream-Dashboard and topic modeling components will later generate topics based on the data filtered through this component. The Twitter stream filter component accepts a stream of tweets, and filters it to include only data containing words that are entered by the user. The filter words are later refined using the query reformulation component, presented in Section 4.2.7. The Twitter stream filter component then sends the filtered posts to the next component, i.e. the pre-processing component, where the tweets will further be processed.

4.2.2 Pre-processing Component

The preprocessing component preprocesses the *tweets*, obtained from the filtered Twitter fire-hose API. The component extracts different kinds of information, e.g. URLs, Hashtags, multimedia content such as pictures and videos etc. This extra information can be useful to validate and enrich the topics in the next iterations. For example, posts can later be clustered/grouped together if they contain similar hashtags, URLs or multimedia content. This extra information, in addition to the tweet text, can help validate the quality of the topics.

4.2.3 Online Clustering and Topic Modeling Component

The online topic modeling component is the main component where the topics are extracted, while Stream-Dashboard [23] mines and keeps track of the clusters and their properties. Whenever a milestone is encountered, the topic modeling component is triggered. Stream-Dashboard then streamlines the data from each cluster to the topic modeling component, where the specialized topics are extracted. The topic modeling component accepts

Algorithm 4.1 Merging topics from different clusters for Online LDA.

Function: MergeTopicDistributions()

Input: *Two Topic* $\times$ *Word distributions,* λ *and* $\tilde{\lambda}$ *that are to be merged, where* $\lambda.N$ *denotes the number of words in the distribution* λ*, words in the distribution* λ *is given by* $\lambda.words$*, and* λ_w *denotes frequency distribution for word* w*. Weight of the information for a given batch/distribution is denoted by:* $\rho_t \in [0, 1]$.

Output: *A single unified topic distribution:* λ*, where the distribution* $\tilde{\lambda}$ *is merged into* λ *(i.e.* λ *is updated with values from* $\tilde{\lambda}$*).*

$W = \lambda.words \cup \tilde{\lambda}.words$ // Merge the words from both distributions.
$distinctWords = set\,(W)$ // Get unique words
$totalWords = |distinctWords|$ // Get total words, where N is the count of words for a given topic distribution.
Scale: $s = (1 - \rho_t) \times totalWords / \,(\lambda.N)$
Scale: $\tilde{s} = (\rho_t) \times totalWords / \left(\tilde{\lambda}.N\right)$
for all word w ***in*** $distinctWords$ **do**
  $newWordFreqDist = s \times \lambda_w + \tilde{s} \times \tilde{\lambda_w}$ // Merge the frequency distributions.
  if $newWordFreqDist >= 1$ **then**
    $\lambda_w = newWordFreqDist$
  else
    $\lambda_w = 0$
    Delete word w
  end if
  for all topic k in K number of topics **do**
    $newTopicFreqDist = s \times \lambda_{wk} + \tilde{s} \times \tilde{\lambda_{wk}}$ // Merge the frequency distributions for a word w and topic k.
    if $newTopicFreqDist >= 1$ **then**
      $\lambda_{wk} = newTopicFreqDist$ // Update frequency distribution for w.
    else
      $\lambda_{wk} = 0$
      Delete word w from topic k
    end if
  end for
end for

the number of topics to be extracted from each cluster as a user input this can also estimated from the cluster properties such as scale, density, cardinality etc.

Stream-Dashboard continuously receives new data from the stream, and updates the cluster model. Various actions are performed on the clusters when a milestone is encountered, such as merging, deletion etc. These updates are then pushed to the topic modeling component, which consequently merges topic models from merged clusters, deletes topic models from defunct clusters, etc. Algorithm 4.1 provides the pseudocode for merging two

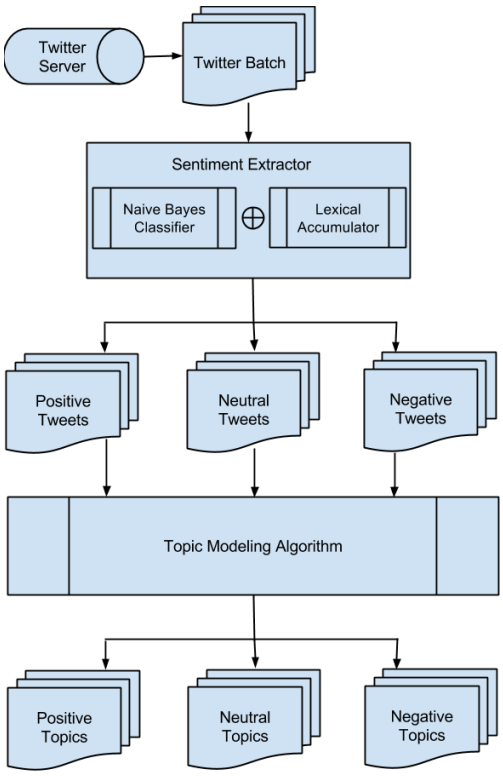


Figure 4.4: Topic Modeling Framework.

topic models. Since the framework is completely online, revisiting the old data is not a viable option, hence the need to merge the topic models themselves instead of re-extracting topics. Once the topics are extracted for each cluster, they are streamlined to the query reformulation component which will adapt the initial user queries, and as a result, improve the filter terms.

4.2.4 Online Joint Sentiment Topic Modeling

Our proposed framework uses sentiment analysis to facilitate the extraction of topics that are furthermore annotated with a sentiment polarity. This can help understand the driving factors and general public opinon on a given topic. We combined a probabilistic modeling framework based on Latent Dirichlet Allocation (LDA), called joint sentiment/topic model (JST) [75], which detects sentiments and topics simultaneously from text with Online LDA [4] to discover topics and sentiments in an online fashion.

The proposed technique, described in Algorithm 4.2, assumes a real time streaming

Algorithm 4.2 Online LDA Variational Bayes with Sentiment Analysis.

Function: OnlineLDAVBSentiments()
Input: *A list of documents, hyper-parameters α and β. Document set/batch D with sentiment labels. Sentiment Label set $S \ni \{positive, negative, neutral\}$*
Output: *Topic and Word distributions.*

for all s in S **do**
  Define $\rho_t \triangleq (\tau_0 + t)^{-k}$
  Initialize λ randomly
  for t=0 to ∞ **do**
    Initialize $\gamma_{tks} = 1$
    repeat
      Set $\phi_{twks} \propto \exp\{\mathbb{E}_q[\log \theta_{tks}] + \mathbb{E}_q[\log \beta_{kws}]\}$
      Set $\gamma_{tks} = \alpha + \sum_w \phi_{twks} n_{tws}$
    until $\frac{1}{K}\sum_k |change\ in\ \gamma_{tks}| < 0.00001$
    Compute $\tilde{\lambda}_{kws} = \eta + D n_{tws} \phi_{twks}$
    Set $\lambda_s = (1 - \rho_t)\lambda_s + \rho_t \tilde{\lambda}_s$
  end for
end for

data input and is replicated using process calls to the database/storage records containing the posts. For LDA, each post is considered as a single document. The stages include the following:

Stage-1: The documents are first passed through a data pre-processing pipeline to strip unwanted data, such as foreign language phrases, special characters, etc. The stop words are currently retained especially for sentiment extraction. The documents are converted to lower case for uniformity.

Stage-2: As an optional stage, the sentiment polarity for each document is extracted using a pre-trained Naive Bayes classifier, or a Lexicon based technique such as Vader [74].

Stage-3: The resulting documents are then passed through to the topic modeling process.

Part of our research explores the quality of the topic models that are extracted through Sentiment enabled Online LDA that receives the streaming documents that are labeled and grouped together with respect to their sentiment. Since Online LDA is considerably faster, labeling the documents before the topic modeling process should result in a faster technique compared to JST. Figure 4.4 depicts this combined framework. Once the posts are obtained from the Twitter API, each one is tagged, then the posts with the same sentiment level are

grouped together as a batch which is then independently passed to the Online LDA topic modeling process. The latter will result in topics extracted from documents with the same sentiment. This is comparable to the JST results but is expected to converge faster and to handle a growing number of streaming documents.

The JST technique uses Gibbs sampling along with the sentiment labels to extract topics such that all the documents in a given topic must share the same sentiment. Figure 4.4 shows the proposed framework for Online LDA. Note that the JST model does not require a separate sentiment analysis layer . In Stage-2 above, once the tweets arrive as batches, either a Naive Bayes classifier or Lexicon based techniques (e.g. Vader [74]) can be used to extract the sentiment levels. The tweets are then regrouped based on the sentiments and the topic modeling is applied, resulting in topics that are associated with a sentiment.

4.2.5 Stream Clustering and Tracking

Algorithm 4.3 Pseudo-Code for Stream Clustering Component Adopted from Stream-Dashboard [23].

Input: *Data Stream* $X = \{x_j = (x_j^1, ..., x_j^d), \forall j = 1, ..., N\}$
Output: Clustering Model $\zeta = \{C_i, \forall i = 1, ..., K\}$, Regression Models Ξ_P

1: **for** $j = 1$ to N {Loop through the data stream X} **do**
2: $\zeta = StreamClustering(x_j)$ {Update clustering model ζ, using Modified AFTER-Streams; Algorithm 4.4}
3: **if** $mod(j, \Delta_{Reg}) = 0$ {The size of the Regression Window (data points) was encountered.} **then**
4: $\Xi_P = TRACER(P_\zeta)$ {Call *TRACER* module with input $=\Delta_{Reg}$ metric values for each cluster in ζ. }
5: **end if**
6: **end for**

The stream clustering component essentially invokes an adaptation of the Stream-Dashboard algorithm [23] (see Algorithm 4.3). The component utilizes a stream clustering algorithm, with the arrival of each new data point, updates the clustering model. When a new cluster is created by the clustering algorithm, it is added to a stream cluster genealogy, and will be tracked over time. When a predefined regression window size of data has been processed, the TOPIC-TRACER [23] component is invoked.

Algorithm 4.4 Modified AFTER-Streams: Stream Clustering with Online Topic Modeling.

Function: StreamClustering(x_n)
Input: Maximum number of clusters (K_{max}), Initial scales (σ_0, τ_0), density threshold (δ_{min}), maturity age (a_{mature})
Output: Cluster model after n points $\zeta = C_1 \cup C_2 .... \cup C_K$, where $C_i = (c_{i,n}, c'_{i,n}, \sigma^2_{i,n}, \tau^2_{i,n}, a_i, W_{i,n})$

$K = 0$
for $n = 1$ to K **do**
  Compute the distances: d^2_{in}, η^2_{in}, and robust weights: $w_{in,n}$, $f_{in,n}$ between x_n and clusters $C_i, \forall i = 1, .., K$ {single pass over the data stream of size N}
  if $K < K_{max}$ And x_n is an outlier with respect to all clusters in ζ (Definition 4) **then**
    $K = K + 1$ {Create a new cluster centered on x_n}
    $c_K = x_n$ {centroid}
    $\sigma_K = \sigma_0$ and $\tau_K = \tau_0$ {initial scales}
    $a_K = 0$ {initial age}
    $W_K = 1$ {initial sum of robust weights}
    $\delta_K = \frac{1}{\sigma_0^2 \tau_0^2}$ {initial density}
  end if
  for all Clusters (C_i, where $i = 1, .., K$) **do**
    if x_n is NOT an outlier with respect to cluster i **then**
      Update $c_{i,n}$ using equation (3.10)
      Update $c'_{i,n}$ as: $c'_{i,n} = \frac{c'_{i,n-1} W_{i,n-1} + w_{in,n} f_{in,n} t_n}{W_{i,n-1} + w_{in,n} f_{in,n}}$
      Update $\sigma^2_{i,n}$ using equation (3.12)
      Update $\tau^2_{i,n}$ as: $\tau^2_{i,n} = \frac{\rho\left(\tau^2_{i,n-1} W_{i,n-1}\right) + w_{in,n} f_{in,n} \eta^2_{ij}}{\rho(W_{i,n-1} + w_{in,n} f_{in,n})}$
      Update sum of weights using equation (3.5)
      Update density using equation (3.8)
    end if
    Update age $a_i = a_i + 1$
  end for
  for all Pairs of clusters $C_i \& C_k$, where $i, k = 1, .., K$ **do**
    if C_i and C_k are Chebyshev-compatible using equation (3.21) **then**
      Merge clusters C_i and C_k using equations (3.22) and (3.23)
      Merge their topic models T_i and T_k {***Call***: **MergeTopicDistributions()** in Algorithm **4.1**}
    end if
  end for
  for all Clusters (C_i, where $i = 1, .., K$) **do**
    if $(a_i > a_{mature}) \,\&\, (\delta_i < \delta_{min})$ **then**
      $\zeta = \zeta - C_i$ {remove mature clusters that have low density}
      $K = K - 1$
    end if
  end for
end for

Algorithm 4.5 TOPIC-TRACER (At time period t) [23].

Function: TRACER(P_ζ)
Input: Cluster metric values P_i stored after receiving the last Δ_{Reg} data points
Output: Regression Models $\Xi_{P_i,[t-1,t]}$ and Behavioral Profiles $\mathcal{H}_i$

```
for all cluster C_i , i = 1, .., K do
  for all metric P_i do
    if a milestone is detected then
      Invoke Topic Modeling { Call: OnlineLDAVBSentiments() for all sentiments, Algorithm: 4.2 }
      Create a new regression model for P_i
    else
      Update the regression model for P_i
    end if
  end for
  Find the cluster transitions using the cluster rules [23].
  Update the behavioral profile H_i [23].
  Update Stream Genealogy graph where transitions took place [23].
end for
```

The stream clustering algorithm can be any generic online clustering algorithm that returns basic cluster metrics such as centroid, scale, and density, even though our research has exclusively used AFTER-Streams [97] and RINO-Streams [22]. The pseudocode for a modified AFTER-Streams is described in Algorithm 4.4. The modification reflects the need for merging the topic models when any two clusters are merged by AFTER-Streams. AFTER-Streams creates new clusters when a new data point is not an outlier. As these clusters grow and whenever a milestone is encountered by the TOPIC-TRACER component, the topic modeling component gets invoked and topics are extracted. AFTER-Streams handles the cluster merging and removal of matured clusters automatically. Algorithm 4.5 provides the pseudocode for TOPIC-TRACER which is modified from [23] to invoke topic modeling, where for each cluster, the metrics are updated, and whenever a milestone gets detected, the Online LDA topic modeling is invoked.

4.2.6 Topic Agglomeration

The topics generated by the topic modeling component can be merged into a smaller set of topics. This provides a reduced set of keywords for Query Reformulation. The topic

agglomeration can also provide a more concise summary of the discovered topics as we have done in the SNOW competition [98], and a good way for validating the generated topics. Section 4.3.8 presents a set of initial experiments with topic agglomeration.

4.2.7 Query Reformulation

The topic modeling component provides knowledge feedback in terms of new terms that can be adapted by query reformulation techniques [78, 99]. The query reformulation component uses pseudo relevance feedback to improve the query that the user had provided. After a user provides an initial query, the user is presented with the results and along with that, the query is reformulated with new terms from the current data and new data crawled from Twitter Advanced Search web pages[1]. Section 2.5 provided an overview of the relevance feedback mechanism. Recall that the pseudo relevance feedback mechanism uses the current data and the newly crawled data to generate a new set of terms, that are passed to the initial tweet filter, which will in turn help collect more data that is relevant to a specific topic(s), that the user has been exploring. Figure 4.3 shows an overview of the reformulation process. Once the initial terms/query and the topic terms from the topic modeling component are provided to the reformulation component. Meanwhile, a new set of documents are collected from the Twitter stream data source. The ROCCHIO approach is finally applied on this collection to compute a reformulated query. The new terms in this query are then used as a new filter, to obtain a more relevant set of documents.

4.2.7.1 Pseudo Relevance Feedback

The *pseudo-relevance feedback* [80–82] procedure involves taking the very top results returned by an initial query as relevant results, and then selecting the top n terms from these documents using for instance tf-idf weights. *Query Reformulation* is finally performed, by adding these terms to the previous query which is then submitted again to find more relevant documents. This automated feedback technique has been shown to work well within

[1] `https://twitter.com/search-advanced?lang=en`

Algorithm 4.6 Query Reformulation with Pseudo-Relevance Feedback

Input: Initial query $\vec{q}_{current}$, Precision threshold P_t
Output: Reformulated query $\vec{q}_{opt}$

$P_{current} = 0$
while $P_{current} < P_t$ **do**
  Get a new set of documents $D_{current}$ with the query $\vec{q}_{current}$ for the timestamps of the documents used for the topic model.
  Get relevant (D_r) and non-relevant (D_{nr}) documents with Implicit/Explicit/Blind feedback. {Relevant documents (D_r) have a cosine similarity of greater than 0.1 with the top topic terms.}
  Get current precision $P_{current} = \frac{|D_r|}{|D_{current}|}$
  Index and weigh current posts set.
  if $P_{current} = 0$ **then**
    Exit; {Cannot reformulate any more.}
  end if
  if $P_{current} < P_t$ **then**
    Optimize query $\vec{q}_{current}$ with Equation: $\vec{q}_{current} = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$
  end if
end while

the context of text/web search [100]. Through a query expansion, some relevant documents that have been missed in the initial round, can then be retrieved to improve the overall performance. Algorithm 4.6 presents the pseudocode for the query reformulation.

Definition 4.2.1. *Current Precision for Psuedo-Relevance Feedback: For a given set of queried/retrieved documents $D_{current}$ and a set of relevant documents D_r within $D_{current}$, the precision is defined as:*

$$P_{current} = \frac{|D_r|}{|D_{current}|} \tag{4.1}$$

Precision is calculated implicitly based on the number of relevant documents from a batch of queried/retrieved documents using the top topic terms of a topic.

The relevant documents are computed by using *cosine* similarity, where if a retrieved document has a similarity of greater than 0.1 with the top topic terms, the document is deemed as relevant else non-relevant.

4.3 Experiments

Topic modeling is generally applied on a collection of documents. Our research deals primarily with social media data such as Twitter. Each tweet from a user can only have a maximum length of 140 characters, this implies that the length of these documents is very small compared to typical text collections. Our dataset consists of tweets that were acquired from Twitter by continuous querying using the Twitter API (Twitter Firehose [1]). Section 4.3.2 provides detailed information about the datasets and the pre-processing step.

4.3.1 Evaluation Plan

The research contribution for the Compartmentalized Framework can be outlined as shown in Table 4.1.

TABLE 4.1

Summary of Research Goals and Evaluation Plan

Framework Component	Goal	Methodology	Research Questions	Validation Metrics	Evaluation Section	Data	Baseline
Topic Mining in evolving data streams	Obtain pure topics	Compatmentalized Framework (Sec 4.2)	RQ2.1, RQ2.2	Perplexity and Coherence	Sec 4.3	Trump, Hurricanes 4.3.2	Online LDA [4]
	Reduce vocabulary size						
	Detect milestones						
	Adapt stream data filters to evolving topics.	Query Reformulation (Sec 4.2.7)	RQ2.3, RQ2.4	New filter terms and unseen topics, Topic Agglomeration (qualitat -ive evaluation).	Sec 4.3.8	Trump, Hurricanes 4.3.2	Google Trends
	Discover new topics						

4.3.1.1 Research Questions

The goal of the Compartmentalized Framework is to mine topics from an evolving data stream, detect milestones, adapt-to and discover new and evolving topics. Solving the following research problems will show how the Compartmentalized Framework is an effective method to mine topics in evolving data streams.

[1]Twitter Firehose: \url{https://dev.twitter.com/streaming/firehose}

Research Question 2.1: Does auto-tuning the Online LDA hyperparameters have a significant effect on the quality of the topics?

Research Question 2.2: Does the Compartmentalized Framework perform better than the baseline algorithm?

Research Question 2.3: Does the Compartmentalized Framework with Query Reformulation perform better than the baseline algorithm?

Research Question 2.4: Does the Compartmentalized Framework with Query Reformulation produce more diverse set of topics?

4.3.2 Datasets

TABLE 4.2

Twitter Dataset Details

Dataset Name	# of Tweets	Vocabulary Size	Filtering Keywords	Data Range
Hurricanes	454,109	18,288	*hurricane, harvey, irma*	Aug 25th, 2017 to Sep 19th, 2017
Trump	646,470	21,306	*trump*	May 1st, 2016 to Nov 24th, 2017

Experiments were performed to extract topics from two datastreams, one on President Donald Trump, and the other related to Hurricane Harvey and Irma. The data/tweets were collected using the filter words: *Trump* and {*Hurricane, Harvey, Irma*} respectively. Table 4.2 provides the dataset descriptions.

4.3.2.1 Dataset Preprocessing

The datasets were preprocessed by removing stopwords, numbers, terms of a single character. We also added bi-grams at the end of the document (i.e. tweet) if the bi-gram appeared in atleast 20 other documents of the dataset. We also removed words that appeared in less than 20 documents. The dataset vocabulary details (including bi-grams) are provided in Table 4.2. Hashing can be used to convert text to feature vectors for clustering in the stream clustering component. The document words are hashed to specific locations of the feature vector. This provides a solution for building feature vectors when the vocabulary of the dataset or the data stream is not available. For experimental purposes we used a regular count based feature vector for the stream clustering component.

4.3.3 Evaluation Metrics

To evaluate the topic models we used two metrics namely, Perplexity [3] and Topic Coherence [101]. For a good topic model, the Perplexity value is expected to be lower, while the Topic Coherence (UMass) should be higher [101]. Perplexity and Topic Coherence (UMass) are given as:

$$perplexity\left(D_{test}\right) = \exp\left\{-\frac{\sum_{d=1}^{M}\log p\left(w_d\right)}{\sum_{d=1}^{M} N_d}\right\} \tag{4.2}$$

where M is the number of documents in the document set D_{test}, w_d represents the words in document d, and N_d is the count of words in document d.

$$Coherence_{UMass} = \frac{2}{N\cdot(N-1)}\sum_{i=2}^{N}\sum_{j=1}^{i-1}\log\left(\frac{p\left(w_i, w_j\right)+\epsilon}{p\left(w_j\right)}\right) \tag{4.3}$$

where N is the number of top words of a topic, w_i is a term from the top topic terms list, $p(w_i)$ is the probability of topic containing term w_i, ϵ is a constant. Word probabilities are estimated based on document frequencies of the original documents used for learning the topics [101, 102].

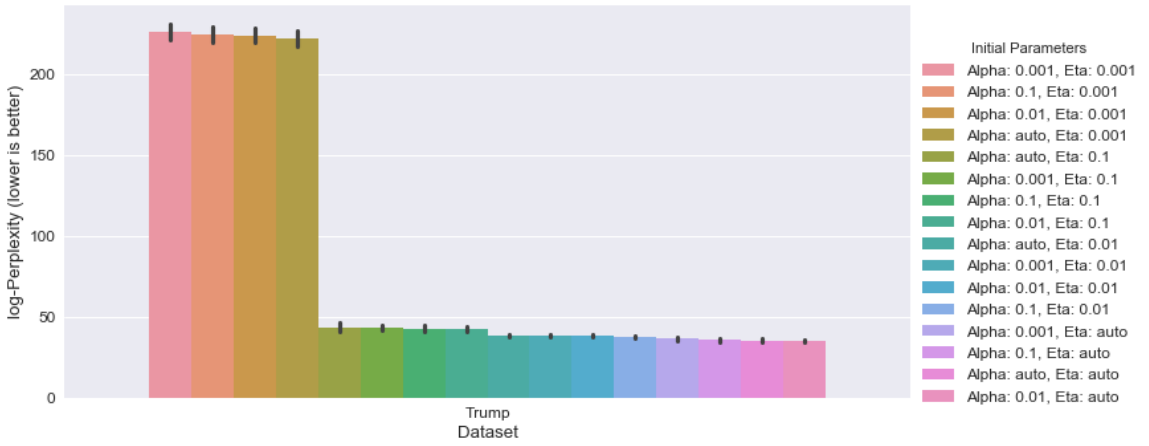


Figure 4.5: *Dataset: Trump.* Performance of Online LDA [4] with and without *automatic* hyperparameter estimation. Lower Perplexity values are better.

4.3.4 Online LDA Hyperparameter Estimation

The Dirichlet distribution is a multivariate distribution [3]. LDA assumes that a single document may contain multiple topics, and a single topic is spread over a number of vocabulary words. Section 2.3.4 introduced the hyperparameter estimation for LDA.

Hypothesis 2.1: From RQ2.1 (Sec 4.3.1.1), Online LDA produces topics of the same quality regardless of whether the hyperparameters are auto-tuned or are set to a constant value.

If the *p-value* is less than α (i.e. 0.05) then the factor (i.e. one of the LDA hyperparameters) has a significant effect on the quality of the topics generated. The results can be summarized as:

Figure 4.5 shows the Perplexity trends for the Online LDA algorithm [4] under different hyperparameter settings. The quality of the topics extracted is higher (i.e. lower Perplexity) when the hyperparameters are autotuned using Newton's method [3,65]. When using an auto-tuned hyper parameter the quality of topics is significantly higher with a *p-value* of **0.0001** and with a moderate Effect Size (Cohen's d) of **-0.505**.

Figure 4.6 shows the Perplexity evaluations for the Online LDA with the *infinite vocabulary* algorithm [5] under different hyperparameter settings. When using an auto-

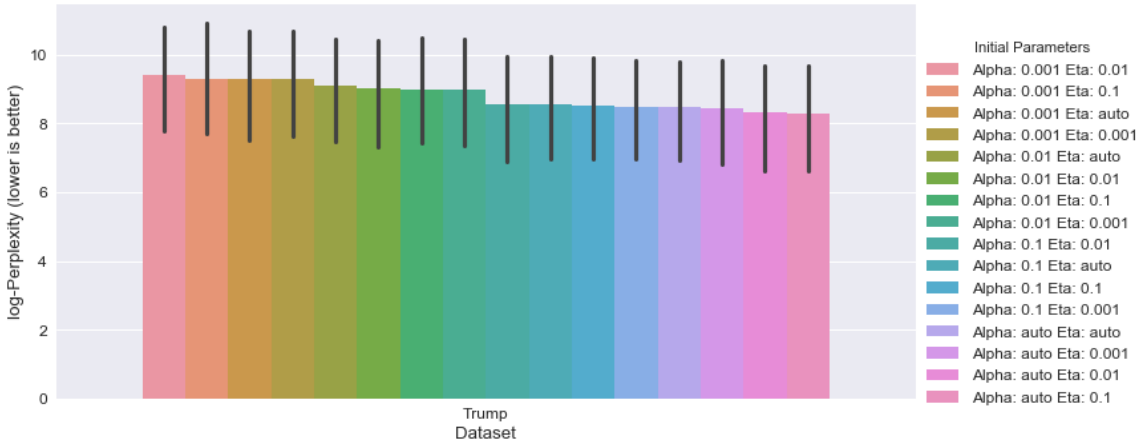


Figure 4.6: *Dataset: Trump.* Performance of Online LDA with infinite vocabulary [5] with and without *automatic* hyperparameter estimation.

tuned hyper parameter the quality of topics (i.e. lower Perplexity) is significantly higher with a *p-value* of **0.0043** but with a very small Effect Size (Cohen's d) of **-0.078**.

4.3.5 Compartmentalized Framework Performance

Hypothesis 2.2: From RQ2.2 (Sec 4.3.1.1), the Compartmentalized Framework produces topics of same quality when compared to Online LDA.

If the *p-value* is less than α (i.e. 0.05) then the Compartmentalized Framework has significant effect on the quality of topics. The results can be summarized as:

Figure 4.7 shows the Perplexity and Topic Coherence evaluation for the Compartmentalized Framework with AFTER-Streams and RINO-Streams compared to the Online LDA algorithm [4] while using only *autotuned* hyperparamters. The quality of the topics extracted using AFTER-Streams are higher (i.e. lower Perplexity and higher Topic Coherence) when compared to RINO-Streams and Online LDA.

Tables 4.3 shows the Effect Size and *p-value* (*Cohen's d*) for ANOVA performed on the Perplexity and Topic Coherence results. The values convey that the Compartmentalized Framework has a moderate to significant effect on the quality of the topics compared to Online LDA.

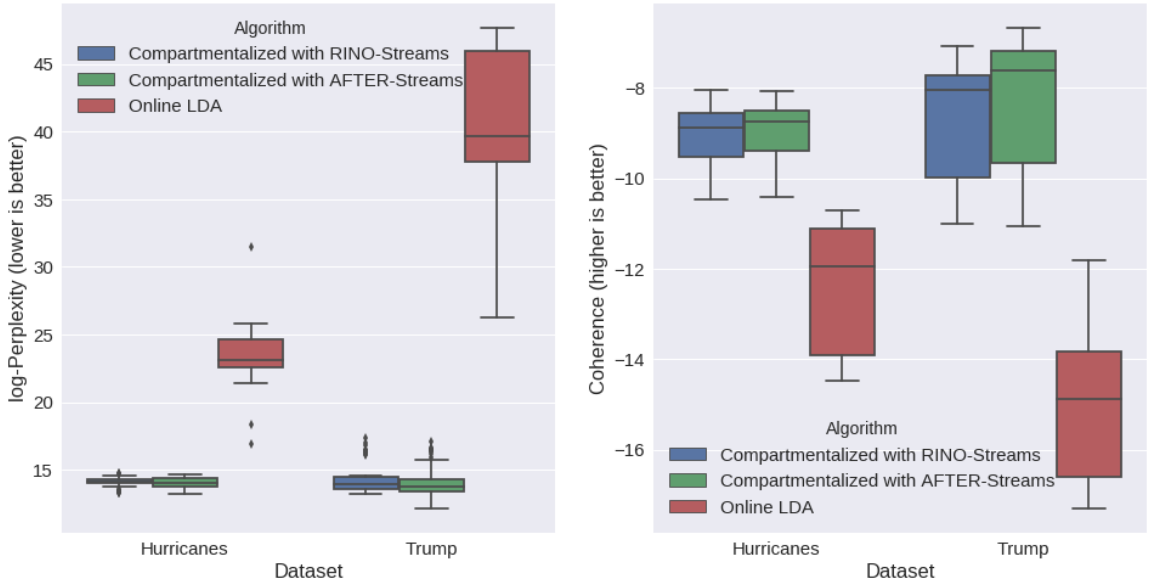


Figure 4.7: *Datasets: Trump and Hurricanes.* Performance of Compartmentalized Framework with AFTER-Streams and RINO-Streams compared to Online LDA [4].

TABLE 4.3: Compartmentalized Framework with AFTER-Streams and RINO-Streams compared to Online LDA. Cohen's d Effect Size (p-*value*).

AFTER-Streams vs	Perplexity	Topic Coherence
RINOStreams	-0.13 (0.04)	0.67 (0.0001)
Online LDA	-1.06 (0.000)	2.22 (0.000)

4.3.6 Online Joint Sentiment Topic Modeling

Figure 4.8 shows the top 3 topics of positive and negative sentiments extracted.

4.3.7 Compartmentalized Framework

The Online Stream Topic Modeling framework can be evaluated using the Perplexity metric. We have performed experiments comparing the benchmark Online LDA [4], with the proposed Compartmentalized topic modeling framework. Our experiments have consistently shown that the proposed methodology produces better perplexity results compared to Online LDA. Figure 4.7 and Table 4.3 show the performance evaluation of the Compartmentalized Framework using Perplexity and Topic Coherence metrics. The plots show that the Compartmentalized Framework outperforms conventional Online LDA. Compartmentalized

Positive	Negative
god president love news bless god_bless	news donald mexico world troop war president raid
supporter job good president yes gain patriot	syria war north_korea administration people donald bad
government united congress best winning state	voter fraud people donald investigation

Figure 4.8: *Datasets: Trump.* Top 3 topics of positive and negative sentiments extracted using the Compartmentalized Framework.

Framework also helps reduce vocabulary size of the topic models [103].

4.3.7.1 Query Reformulation with Pseudo-Relevance Feedback

Hypothesis 2.3: From RQ2.3 (Sec 4.3.1.1), the Compartmentalized Framework with Query Reformulation produces topics of the same quality when compared to Online LDA.

If the *p-value* is less than α (i.e. 0.05) then the Compartmentalized Framework has significant effect on the quality of topics. The results can be summarized as:

Figures 4.9, 4.10 and 4.11 show the top topics and their respective reformulated terms. These new terms are added to the data stream filter which facilitates the capture of new evolving topics. The distinctiveness of the topics extracted while using query reformulation is higher as discussed in Section 4.3.8.

Figure 4.12 shows the performance of the Compartmentalized framework while using Query Reformulation. The figure shows that the Compartmentalized framework yields better quality topics with respect to Perplexity and Topic Coherence. Tables 4.4 and 4.5 shows the Effect Size and *p-value* for ANOVA performed on the Perplexity and Topic Coherence results. The values convey that the Compartmentalized Framework with Query Reformulation achieves significant effect on the quality of the topics compared to Online

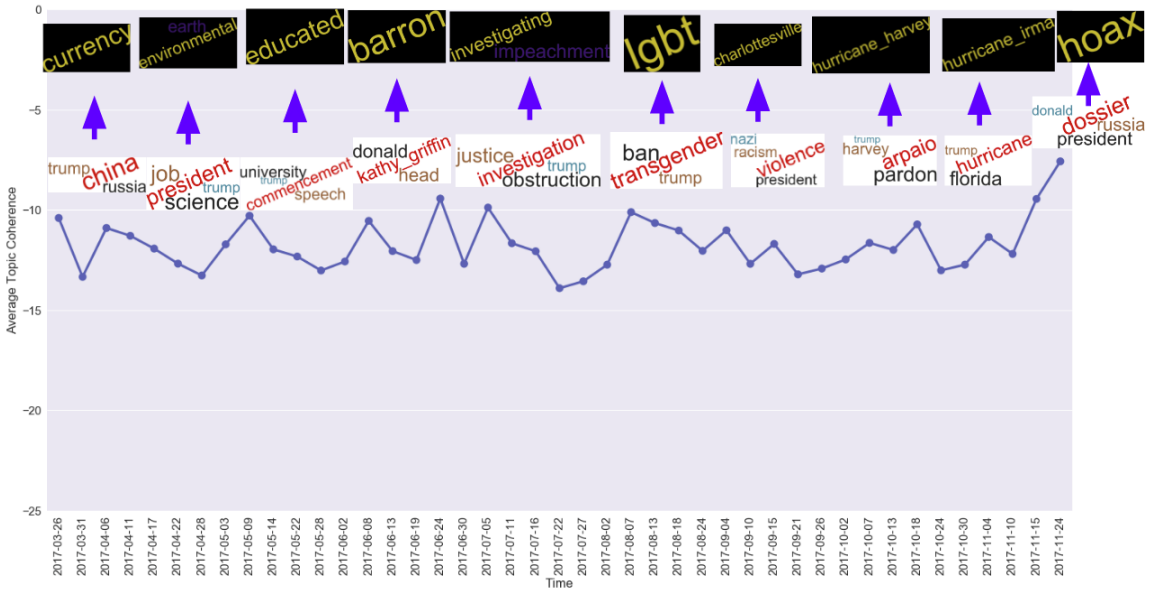


Figure 4.9: *Datasets: Trump.* Top topics extracted using the Compartmentalized Framework, and their respective new reformulated terms. The wordclouds with white background are the topic terms, and the ones with black backgrounds are new reformulated terms. The word clouds are ordered from left to right.

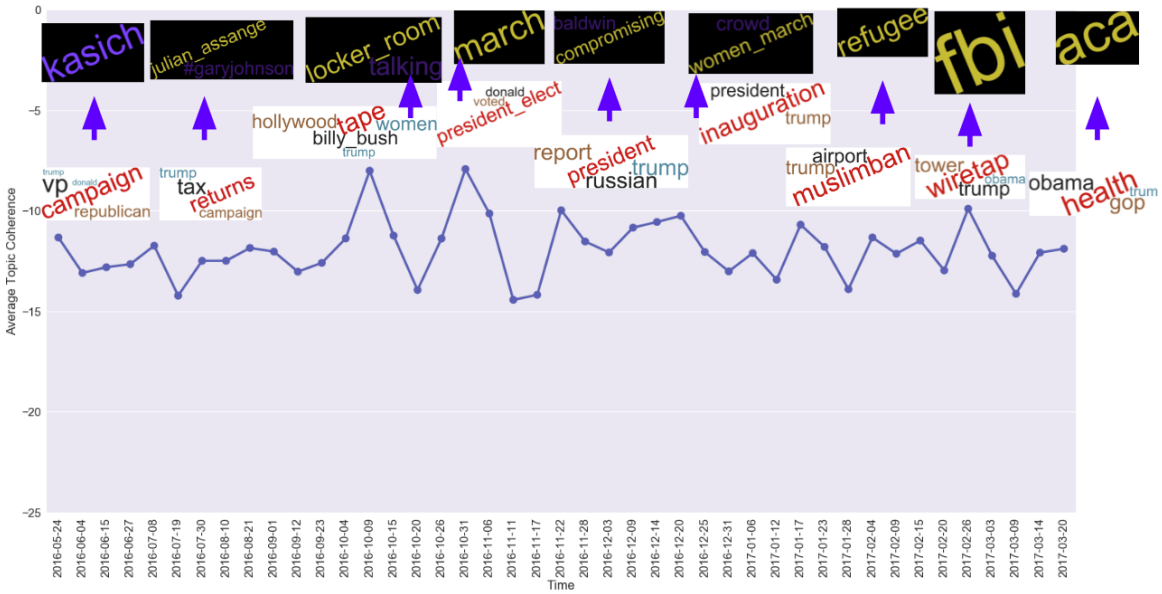


Figure 4.10: *Datasets: Trump.* Top topics extracted using the Compartmentalized Framework, and their respective new reformulated terms. The wordclouds with white background are the topic terms, and the ones with black backgrounds are new reformulated terms. The word clouds are ordered from left to right.

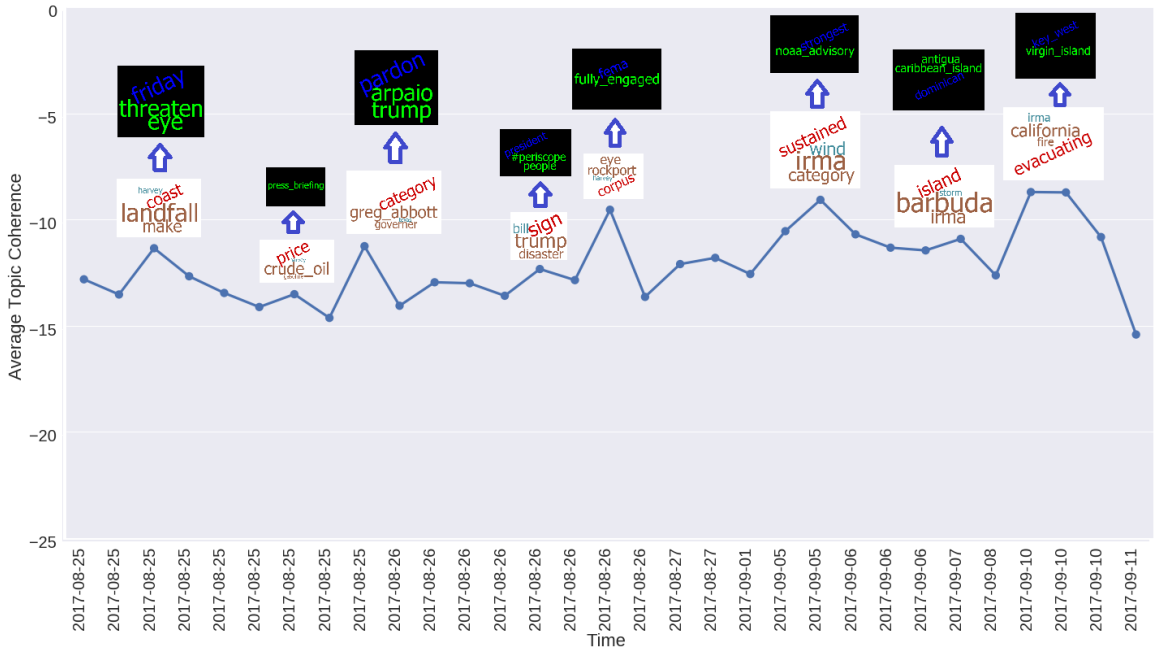


Figure 4.11: *Datasets: Hurricanes.* Top topics extracted using the Compartmentalized Framework, and their respective new reformulated terms. The wordclouds with white background are the topic terms, and the ones with black backgrounds are new reformulated terms. The word clouds are ordered from left to right.

LDA.

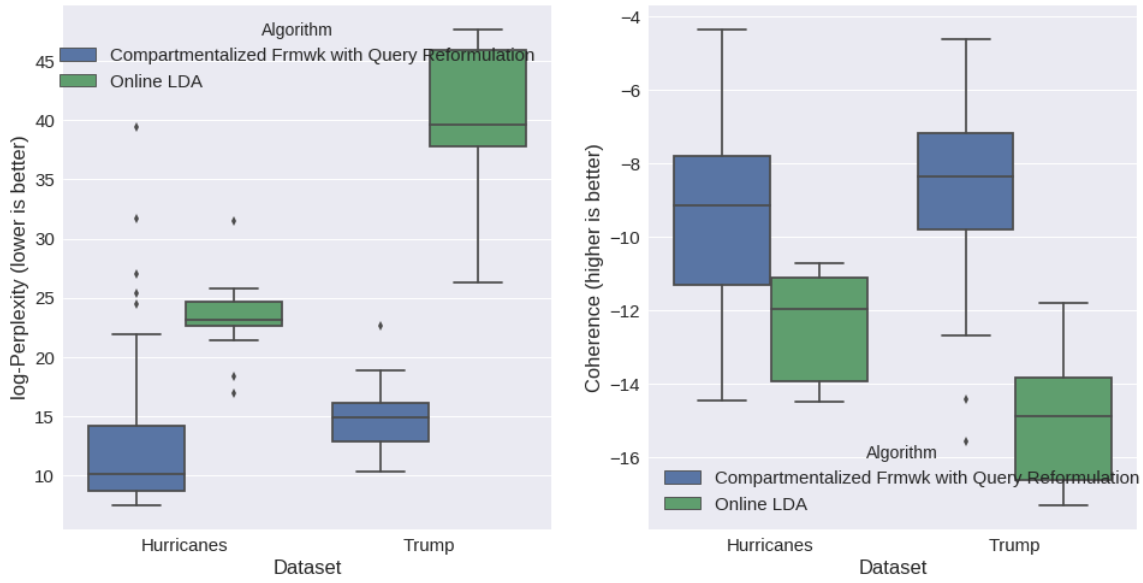


Figure 4.12: *Datasets: Hurricanes, Trump.* Performance of Compartmentalized Framework with Query Reformulation compared to Online LDA [4].

TABLE 4.4: *Dataset: Hurricane.* Compartmentalized Framework with Query Reformulation compared to Online LDA. Cohen's d Effect Size (*p-value*).

Compartmentalized vs	Perplexity	Topic Coherence
Online LDA	-2.003 (0.000)	1.563 (0.000)

TABLE 4.5: *Dataset: Trump.* Compartmentalized Framework with Query Reformulation compared to Online LDA. Cohen's d Effect Size (*p-value*)

Compartmentalized vs	Perplexity	Topic Coherence
Online LDA	-6.127 (0.000)	3.284 (0.000)

If the *p-value* is less than α (i.e. 0.05) then the Compartmentalized Framework has significant effect on the quality of topics. The results can be summarized as:

Figures 4.13 to 4.15 show the Google Trends corresponding to few of the topics in Figures 4.9, 4.10, and 4.11. The Google Trend figures show that the topics were extracted or picked up at a similar time when there was a general public interest. This is one way to verify the validity of the discovered topics and their timeliness.

4.3.8 Topic Agglomeration with Compartmentalized Framework

Hypothesis 2.4: From RQ2.4 (Sec 4.3.1.1), the Compartmentalized Framework with Query Reformulation produces more diverse topics when compared to Online LDA.

One way to both visualize, summarize and evaluate the quality of the mined topics is to apply Hierarchical Agglomerative Clustering [104] to see how the extracted topics are related. A good set of topics is expected to have topics that are internally consistent and

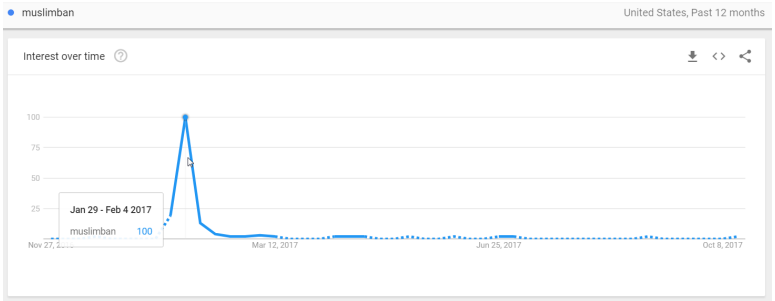


Figure 4.13: *Dataset: Trump.* Google Trend for one topic's top term(s): *muslimban.*

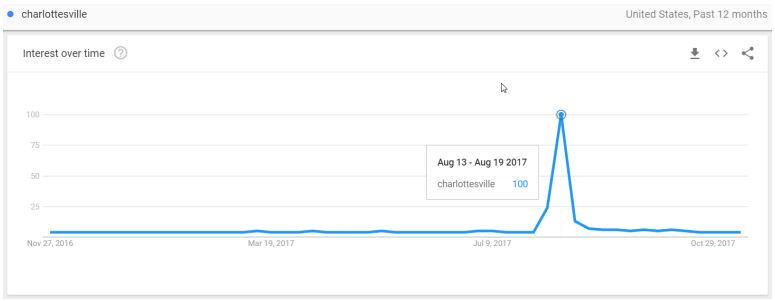


Figure 4.14: *Dataset: Trump.* Google Trend for one topic's top term(s): *charlottesville.*

Figure 4.15: *Dataset: Hurricanes.* Google Trend for one topic's top term(s): *gas price harvey.*

compact as already evaluated. In addition, topics should be distinct to cover a diverse range of non-redundant topics in the data stream using Perplexity and Coherence. Topic Agglomeration can be used as a tool to help organize, understand, and visualize the mined topics whose number can be very large when mining real-life social data streams [98].

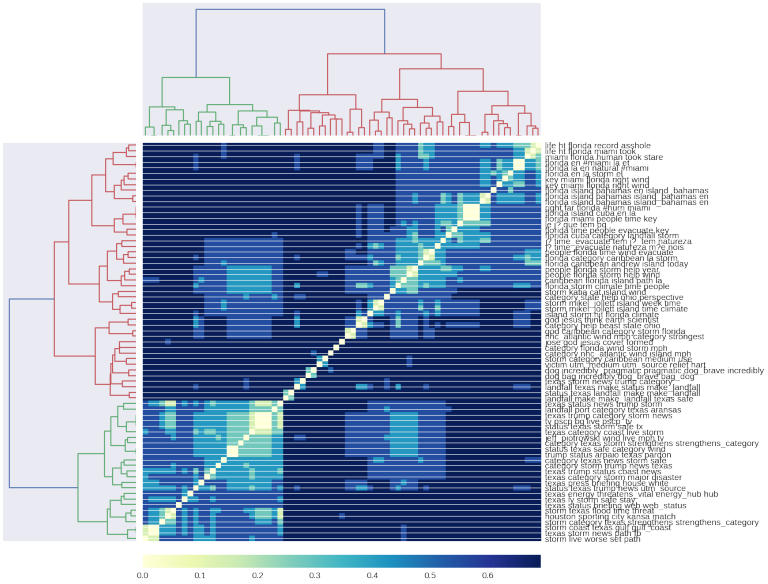
Figures 4.16 and 4.17 show the results of topic agglomeration using Ward link Hierarchical Agglomerative Clustering [104] and Jensen-Shannon Divergence [105] as distance measure for the top 60 topics (i.e. highest Topic Coherence) for the Trump and Hurricanes datasets respectively. The Jensen-Shannon Divergence is given by:

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2} \times D_{KL}(P \parallel M) + \frac{1}{2} \times D_{KL}(Q \parallel M)$$

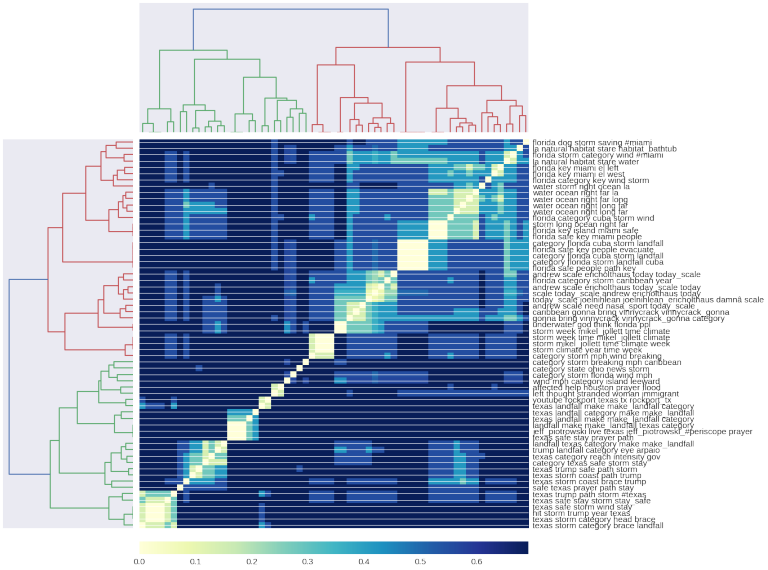
where, P and Q are two probability distributions, $M = \frac{1}{2}(P + Q)$, and $D_{\mathrm{KL}}(P\|M)$ is the KL-Divergence [106] given by:

$$D_{\mathrm{KL}}(P\|M) = -\sum_i P(i) \log \left\{ \frac{M(i)}{P(i)} \right\}$$

The topics are listed in the order of extraction and the agglomeration of these topics resulted in grouping the topics from the same clusters first, which means that the topics



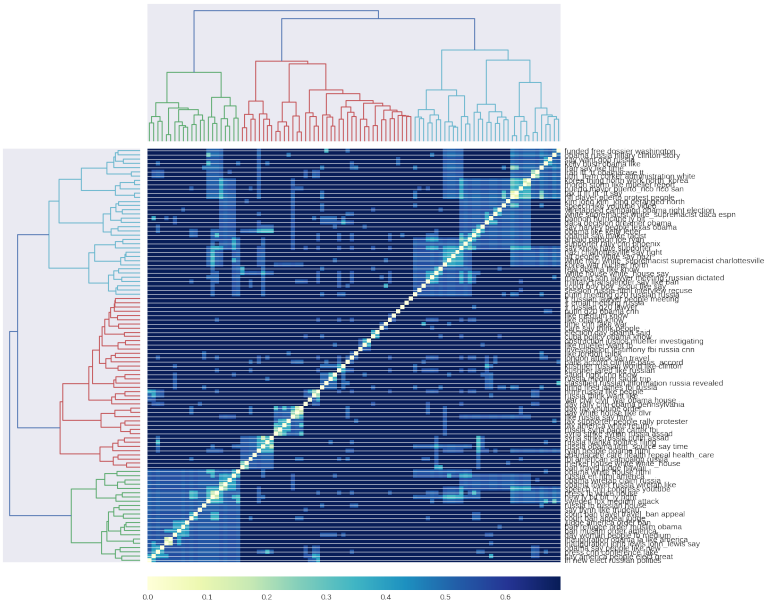
(a) Topic Agglomeration **With Query Reformulation**



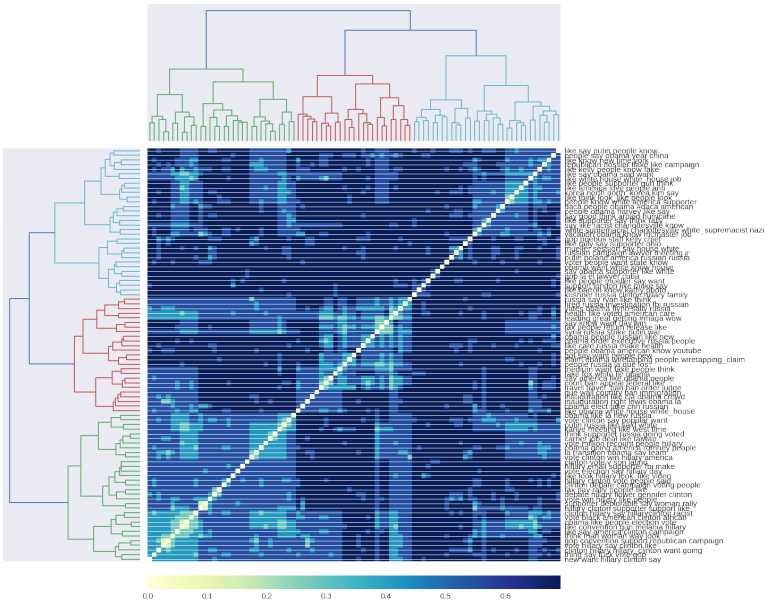
(b) Topic Agglomeration **Without Query Reformulation**

Figure 4.16: Topic Agglomeration using Jensen-Shannon divergence for topics with and without Query Reformulation on *Dataset Hurricanes*. The blocks along the diagonal of the distance heatmap are smaller when using Query Reformulation, indicating that the consecutive top topics are diverse. Bigger blocks indicate overlap of the top topic terms.

are very different for each cluster and are very granular. This contributes to the validation of the results.



(a) Topic Agglomeration **With Query Reformulation**



(b) Topic Agglomeration **Without Query Reformulation**

Figure 4.17: Topic Agglomeration using Jensen-Shannon divergence for topics with and without Query Reformulation on *Dataset Trump*. The blocks along the diagonal of the distance heatmap are smaller when using Query Reformulation, indicating that the consecutive top topics are diverse. Bigger blocks indicate overlap of the top topic terms.

4.4 Summary and Conclusions

This chapter presented the proposed Compartmentalized Topic Modeling framework which uses the modified Stream-Dashboard with AFTER-Streams to initially cluster the data stream and then applies Topic Modeling to extract finer quality topics. The framework also uses Query Reformulation to control the filtering of the input data stream in such a way that the quality of topics is improved.

The chapter also presented experiments on several datasets. Section 4.3 presented a set experiments on different datasets while using the conventional Online LDA. Even though Online LDA has a comparatively low time complexity, looking at the perplexity plots, there is a great room for improvement in terms of the quality of the topics extracted. Section 4.3.6 discussed experiments while using Online LDA, along with sentiment analysis. The results show that the quality of the topics was comparatively better than the conventional Online LDA. The proposed framework used Stream-Dashboard, Online LDA with Sentiment Analysis, and Query Reformulation to provide an integrated platform to extract and track high quality topics. The results showed that the proposed framework can extract higher quality topics than Online LDA with different sentiment values. Using Query Reformulation, the framework can relay new seed information to the initial data filter which can help explore evolving topics.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this dissertation, we proposed a new stream clustering methodology based on the AFTER-Streams clustering algorithm, which extends the RINO-Streams clustering algorithm with an automatic, dynamic, and cluster-specific temporal scale estimation. The proposed algorithm can automatically cluster the data in both the content and temporal spaces. Our extensive experimental results showed that AFTER-Streams outperforms the competitive baseline stream clustering algorithms on TREC (text data), KDDCUP99 (network intrusion logs), and a large battery of synthetic datasets controlling critical parameters that characterize the difficulties associated with a data stream (size, velocity, noise, evolution patterns, etc).

We have also proposed a Compartmentalized Topic Mining framework which uses Stream-Dashboard to initially cluster the data stream and then applies Topic Modeling to extract finer and better quality topics. The framework also uses Query Reformulation to control the input data stream to improve the quality and novelty of topics that are discovered.

The proposed compartmentalized framework clusters the stream data, extracts the topics and seed words from each cluster, and tracks the topics over time. The initial stage involves using a modified Stream Dashboard framework which clusters the data stream points (i.e. documents) and the second stage involves invoking the topic modeling component. From the topics that are extracted, several potential future seed words are extracted and are relayed back to the data stream filters to help focus the discovery on a specific set of topics. We also presented an approach to automatically optimize the hyper-parameters used for topic modeling.

The evaluation experiments confirmed the quality of the topics extracted using topic modeling techniques, then finer topics are extracted using the integrated *Stream-Dashboard + Topic Modeling + Query Reformulation* Framework. The validation results support our claim that when using *Stream-Dashboard + Topic Modeling + Query Reformulation* framework, the topics extracted are of higher quality.

Limitations and Future Work

Demographics can play a critical role in the context of social media data and sentiment analysis of any given discussion. Extracting topics for users in a given location or demographic cross-section may provide a more targeted and possibly more accurate and useful results, depending on the application goals.

The compartmentalized framework uses Query Reformulation to set the data stream filter to extract topics of evolving interest. The topics extracted are dependent on this initial data stream filter. The user should be able to specify additional parameters for this filter e.g. location, query terms, a specific sentiment, etc. The user may also choose some topics of interest in an interactive manner which can later be used to guide subsequent topic discovery.

REFERENCES

[1] Victor J. Yohai Ricardo A. Maronna, Douglas R. Martin, *Robust Statistics*, Wiley-Interscience, 2006.

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer, "Moa: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

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

[4] Matthew Hoffman, David M Blei, and Francis Bach, "Online learning for latent dirichlet allocation," *Advances in Neural Information Processing Systems*, vol. 23, pp. 856–864, 2010.

[5] Ke Zhai and Jordan Boyd-Graber, "Online topic models with infinite vocabulary," in *International Conference on Machine Learning*, 2013.

[6] Jim Gray and Andreas Reuter, *Transaction processing*, Morgan Kaufíann Publishers, 1993.

[7] Jürgen Beringer and Eyke Hüllermeier, "Online clustering of parallel data streams," *Data & Knowledge Engineering*, vol. 58, no. 2, pp. 180–204, 2006.

[8] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[9] Arnold Priguna Boedihardjo, "Efficient algorithms for mining data streams," 2010.

[10] Joseph Weizenbaum, "Eliza-a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[11] Kevin Gimpel, "Modeling topics," 2006.

[12] David M Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[13] David L Marvit, Jawahar Jain, Stergios Stergiou, Alex Gilman, B Thomas Adler, John J Sidorowich, and Yannis Labrou, "Modeling topics using statistical distributions," Oct. 1 2008, US Patent App. 12/243,267.

[14] Thomas Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999, pp. 50–57.

[15] Mark Steyvers and Tom Griffiths, "Probabilistic topic models," *Handbook of latent semantic analysis*, vol. 427, no. 7, pp. 424–440, 2007.

[16] Daniel B Neill, Andrew W Moore, Maheshkumar Sabhnani, and Kenny Daniel, "Detection of emerging space-time clusters," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 218–227.

[17] Michael Steinbach, George Karypis, Vipin Kumar, et al., "A comparison of document clustering techniques," in *KDD workshop on text mining*. Boston, MA, 2000, vol. 400, pp. 525–526.

[18] Jürgen Beringer and Eyke Hüllermeier, "Fuzzy clustering of parallel data streams," *Advances in Fuzzy Clustering and Its Application*, pp. 333–352, 2007.

[19] Antje Düsterhöft and Bernhard Thalheim, *Natural Language Processing and Information Systems*, Ges. für Informatik, 2003.

[20] David M Blei and Jon D McAuliffe, "Supervised topic models," *arXiv preprint arXiv:1003.0783*, 2010.

[21] Charu C Aggarwal and ChengXiang Zhai, *Mining text data*, Springer Science & Business Media, 2012.

[22] Basheer Hawwash and Olfa Nasraoui, "Robust clustering of data streams using incremental optimization.," in *AMW*. Citeseer, 2010.

[23] Basheer Hawwash and Olfa Nasraoui, "Stream-dashboard: a framework for mining, tracking and validating clusters in a data stream," in *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. ACM, 2012, pp. 109–117.

[24] A.K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[25] Yi-Kuei Lin, Yu-Chung Tsao, and Shi Woei Lin, *Proceedings of the Institute of Industrial Engineers Asian Conference 2013*, Springer Science & Business Media, 2013.

[26] John A Hartigan and Manchek A Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.

[27] James C Bezdek, Robert Ehrlich, and William Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, 1984.

[28] Dan Pelleg, Andrew W Moore, et al., "X-means: Extending k-means with efficient estimation of the number of clusters.," in *ICML*, 2000, pp. 727–734.

[29] Todd K Moon, "The expectation-maximization algorithm," *Signal processing magazine, IEEE*, vol. 13, no. 6, pp. 47–60, 1996.

[30] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, 2000, pp. 58–64.

[31] Patrick JF Groenen and Krzysztof Jajuga, "Fuzzy clustering with squared minkowski distances," *Fuzzy Sets and Systems*, vol. 120, no. 2, pp. 227–237, 2001.

[32] Catherine A Sugar and Gareth M James, "Finding the number of clusters in a dataset," *Journal of the American Statistical Association*, vol. 98, no. 463, 2003.

[33] Padhraic Smyth, "Model selection for probabilistic clustering using cross-validated likelihood," *Statistics and Computing*, vol. 10, no. 1, pp. 63–72, 2000.

[34] Kenneth P Burnham and David R Anderson, "Multimodel inference understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.

[35] Katsuhiro Nishinari, Martin Treiber, and Dirk Helbing, "Interpreting the wide scattering of synchronized traffic data by time gap statistics," *Physical Review E*, vol. 68, no. 6, pp. 067101, 2003.

[36] Douglas H. Fisher, "Iterative optimization and simplification of hierarchical clusterings," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 147–179, 1996.

[37] S. Zhong, "Efficient online spherical k-means clustering," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. Ieee, 2005, vol. 5, pp. 3180–3185.

[38] F. Cao, M. Ester, 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*, 2006, pp. 328–339.

[39] D. Barbará, "Requirements for clustering data streams," *ACM sIGKDD Explorations Newsletter*, vol. 3, no. 2, pp. 23–27, 2002.

[40] Chris AJ Klaassen, "On an inequality of chernoff," *The Annals of Probability*, pp. 966–974, 1985.

[41] Olfa Nasraoui, "A brief overview of robust clustering techniques," in *Online]. Available: http://www. louisville. edu/ o0nasr01/Websites/tutorials. html*. Citeseer, 1999.

[42] Peter J. Huber, *Robust Statistics*, Wiley-Interscience, 1981.

[43] O. Nasraoui and C. Rojas, "Robust clustering for tracking noisy evolving data streams," in *Proc. 2006 SIAM Conf. on Data Mining (SDM 2006)*, 2006, pp. 80–99.

[44] Peter J Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[45] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[46] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu, "A framework for projected clustering of high dimensional data streams," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 852–863.

[47] Olfa Nasraoui, Cesar Cardona Uribe, Carlos Rojas Coronel, and Fabio Gonzalez, "Tecno-streams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 235–242.

[48] Charu C Aggarwal, *Data streams: models and algorithms*, vol. 31, Springer Science & Business Media, 2007.

[49] Spiros Sirmakessis, *Adaptive and personalized semantic web*, vol. 14, Springer Science & Business Media, 2006.

[50] Ricardo A Maronna, R Douglas Martin, and Victor J Yohai, "Robust statistics: Theory and methods. 2006," *J. Wiley*.

[51] Albert W Marshall and Ingram Olkin, "Multivariate chebyshev inequalities," *The Annals of Mathematical Statistics*, pp. 1001–1014, 1960.

[52] Wassily Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[53] Basheer Hawwash, *Stream-Dashboard: A big data stream clustering framework with applications to social media streams*, University of Louisville, 2013.

[54] Steven P Crain, Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha, "Dimensionality reduction and topic modeling: From latent semantic indexing to latent dirichlet allocation and beyond," in *Mining Text Data*, pp. 129–161. Springer, 2012.

[55] Shokri Z Selim and Mohamed A Ismail, "Soft clustering of multidimensional data: a semi-fuzzy approach," *Pattern Recognition*, vol. 17, no. 5, pp. 559–568, 1984.

[56] James E Mosimann, "On the compound multinomial distribution, the multivariate β-distribution, and correlations among proportions," *Biometrika*, pp. 65–82, 1962.

[57] S Dumais, G Furnas, T Landauer, S Deerwester, S Deerwester, et al., "Latent semantic indexing," in *Proceedings of the Text Retrieval Conference*, 1995.

[58] L De Lathauwer, B De Moor, J Vandewalle, and Blind Source Separation by Higher-Order, "Singular value decomposition," in *Proc. EUSIPCO-94, Edinburgh, Scotland, UK*, 1994, vol. 1, pp. 175–178.

[59] Hanna M Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 977–984.

[60] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[61] Mark D Smucker and James Allan, "An investigation of dirichlet prior smoothing's performance advantage," Tech. Rep., Citeseer, 2005.

[62] Jun S Liu, "The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem," *Journal of the American Statistical Association*, vol. 89, no. 427, pp. 958–966, 1994.

[63] Limin Yao, David Mimno, and Andrew McCallum, "Efficient methods for topic model inference on streaming document collections," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 937–946.

[64] W Keith Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[65] Gerd Ronning, "Maximum likelihood estimation of dirichlet distributions," *Journal of statistical computation and simulation*, vol. 32, no. 4, pp. 215–221, 1989.

[66] Kendall E Atkinson, *An introduction to numerical analysis*, John Wiley & Sons, 2008.

[67] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno, "Evaluation methods for topic models," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1105–1112.

[68] Alex Wright, "Mining the web for feelings, not facts," *New York Times*, vol. 24, 2009.

[69] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment.," *ICWSM*, vol. 10, pp. 178–185, 2010.

[70] Erik Cambria, Bjorn Schuller, Yunqing Xia, and Catherine Havasi, "New avenues in opinion mining and sentiment analysis," *IEEE Intelligent Systems*, , no. 2, pp. 15–21, 2013.

[71] Andrew Ortony, Gerald L Clore, and Allan Collins, *The cognitive structure of emotions*, Cambridge university press, 1990.

[72] Soo-Min Kim and Eduard Hovy, "Identifying and analyzing judgment opinions," in *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 200–207.

[73] Erik Cambria and Amir Hussain, *Sentic computing: Techniques, tools, and applications*, vol. 2, Springer Science & Business Media, 2012.

[74] CJ Hutto and Eric Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[75] Chenghua Lin and Yulan He, "Joint sentiment/topic model for sentiment analysis," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 375–384.

[76] D Manning Christopher, Raghavan Prabhakar, and SCHTZE Hinrich, "Introduction to information retrieval," *An Introduction To Information Retrieval*, vol. 151, pp. 177, 2008.

[77] Joseph John Rocchio, "Relevance feedback in information retrieval," 1971.

[78] Thorsten Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.," Tech. Rep., DTIC Document, 1996.

[79] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al., *Introduction to information retrieval*, vol. 1, Cambridge university press Cambridge, 2008.

[80] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al., *Modern information retrieval*, vol. 463, ACM press New York, 1999.

[81] Rong Yan, Alexander Hauptmann, and Rong Jin, "Multimedia search with pseudo-relevance feedback," in *Image and Video Retrieval*, pp. 238–247. Springer, 2003.

[82] Marc Caillet, Jean-François Pessiot, Massih-Reza Amini, and Patrick Gallinari, "Unsupervised learning with term clustering for thematic segmentation of texts," in *Coupling approaches, coupling media and coupling languages for information retrieval*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2004, pp. 648–657.

[83] Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma, "Improving pseudo-relevance feedback in web information retrieval using web page segmentation," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 11–18.

[84] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977.

[85] Albert W. Marshall and Ingram Olkin, "Multivariate chebyshev inequalities," *The Annals of Mathematical Statistics*, vol. 31, pp. 1001–1014, 1960.

[86] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou, "Density-based clustering over an evolving data stream with noise," in *In 2006 SIAM Conference on Data Mining*, Bethesda, Maryland, 2006, pp. 328–339, SIAM.

[87] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu, "A framework for clustering evolving data streams," in *In VLDB*, Berlin, Germany, 2003, pp. 81–92, VLDB Endowment.

[88] Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler, "Streamkm++: A clustering algorithm for data streams," *Journal of Experimental Algorithmics (JEA)*, vol. 17, pp. 2–4, 2012.

[89] Basheer Hawwash and Olfa Nasraoui, "Robust clustering of data streams using incremental optimization," in *The IVth Alberto Mendelzon International Workshop on Foundations of Data Management. AMW*, Buenos Aires, Argentina, 2010, CEUR-WS.org.

[90] George Karypis, "Cluto-a clustering toolkit," Tech. Rep., DTIC Document, 2002.

[91] David L Davies and Donald W Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 2, no. 2, pp. 224–227, 1979.

[92] Andrew Rosenberg and Julia Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure.," in *EMNLP-CoNLL*, 2007, vol. 7, pp. 410–420.

[93] Edward B Fowlkes and Colin L Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American statistical association*, vol. 78, no. 383, pp. 553–569, 1983.

[94] Cornelis Joost van Rijsbergen, "A theoretical basis for the use of co-occurrence data in information retrieval," *Journal of documentation*, vol. 33, no. 2, pp. 106–119, 1977.

[95] Hsiang-Tsung Kung, Fabrizio Luccio, and Franco P Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM*, vol. 22, no. 4, pp. 469–476, 1975.

[96] Tai-hoon Kim Carlos Ramos, Jemal Abawajy Byeong-Ho Kang, and Dominik Slezak Hojjat Adeli, "Computer applications for modeling, simulation, and automobile," .

[97] Gopi Chand Nutakki and Olfa Nasraoui, "Clustering data streams with adaptive forgetting," in *Big Data (BigData Congress), 2017 IEEE International Congress on*. IEEE, 2017, pp. 494–497.

[98] Gopi Chand Nutakki, Olfa Nasraoui, Behnoush Abdollahi, Mahsa Badami, and Wenlong Sun, "Distributed lda-based topic modeling and topic agglomeration in a latent space.," in *SNOW-DC@ WWW*, 2014, pp. 17–24.

[99] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay, "Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search," *ACM Transactions on Information Systems (TOIS)*, vol. 25, no. 2, pp. 7, 2007.

[100] Jinxi Xu and W Bruce Croft, "Query expansion using local and global document analysis," in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1996, pp. 4–11.

[101] David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum, "Optimizing semantic coherence in topic models," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 262–272.

[102] Michael Röder, Andreas Both, and Alexander Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the eighth ACM international conference on Web search and data mining*. ACM, 2015, pp. 399–408.

[103] Gopi Chand Nutakki and Olfa Nasraoui, "Compartmentalized adaptive topic mining on social media streams," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 992–997.

[104] William HE Day and Herbert Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of classification*, vol. 1, no. 1, pp. 7–24, 1984.

[105] Jianhua Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[106] Thomas M Cover and Joy A Thomas, "Elements of information theory," 2012.

APPENDIX A

Proofs

A.1 AFTER-Streams

In this section, we will present the proofs for the theorems presented in Chapter 3.

Theorem A.1.1. *Optimal Incremental Centroid Update : Given the previous centroids, $c_{i,n-1}$, and assuming that the scales do not change much relative to the scale that resulted from the previous iteration, the new centroid that optimizes density function in the manuscript after the arrival of the n^{th} data point is given by [22]:*

$$c_{i,n} = \frac{e^{\frac{-1}{\tau}} c_{i,n-1} W_{i,n-1} + w_{in,n} x_n}{e^{\frac{-1}{\tau}} W_{i,n-1} + w_{in,n}} \tag{A.1}$$

Proof. Since the time dependency has been absorbed into the weight function, and by fixing the previous centroid $c_{i,n-1}$, scale $\sigma_{i,n-1}$ and weight sums $W_{i,n-1}$, the equations for the center updates are found by finding the derivative of the Lagrangian of the density $\delta_{i,n}$ with respect to the centroid $c_{i,n}$, while all the other parameters are held constant as follows [22]:

$$\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{1}{\sigma_{i,n}^2} \times \sum_{j=1}^{n} w_{ij,n} \frac{\partial d_{ij}^2}{\partial c_{i,n}} - \frac{\partial \left(\lambda \sigma_{i,n}^2\right)}{\partial c_{i,n}} = \mathbf{0} \tag{A.2}$$

In case an inner norm inducing metric is used such as $d_{ij}^2 = (x_j - c_{i,n})^t A (x_j - c_{i,n})$, where A is a positive semi-definite matrix (A is the identity matrix for the Euclidean norm), it is easy to show that $\frac{\partial d_{ij}^2}{\partial \mathbf{c}_{i,n}} = -2A(x_j - c_{i,n})$. For the case of $A = I$ (identity matrix), we have

$$
\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{1}{\sigma_{i,n}^2} \times \sum_{j=1}^{n} \frac{\partial w_{ij,n}}{\partial c_{i,n}} - \frac{\partial\left(\lambda \sigma_{i,n}^2\right)}{\partial c_{i,n}} \\
&\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{1}{\sigma_{i,n}^2} \times \sum_{j=1}^{n} w_{ij,n} \times \frac{\partial(\frac{-(x_j - c_{i,n})^t (x_j - c_{i,n})}{\kappa \sigma_{i,n}^2})}{\partial c_{i,n}} - 0 \\
&\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{1}{\sigma_{i,n}^2} \times \sum_{j=1}^{n} w_{ij,n} \times \left\{ \frac{\frac{\partial(-(x_j - c_{i,n})^t (x_j - c_{i,n}))}{\partial c_{i,n}} \times \kappa \sigma_{i,n}^2 - \frac{\partial(\kappa \sigma_{i,n}^2)}{\partial c_{i,n}} \times -(x_j - c_{i,n})^t (x_j - c_{i,n})}{(\kappa \sigma_{i,n}^2)^2} \right\} \\
&\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{1}{\sigma_{i,n}^2} \times \sum_{j=1}^{n} w_{ij,n} \times \left\{ \frac{-2(x_j - c_{i,n}) \times \kappa \sigma_{i,n}^2 - 0 \times -(x_j - c_{i,n})^t (x_j - c_{i,n})}{\kappa^2 \sigma_{i,n}^4} \right\} \\
&\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{1}{\sigma_{i,n}^2} \times \sum_{j=1}^{n} \frac{-2 w_{ij} (x_j - c_{i,n})}{\kappa \sigma_{i,n}^2} \\
&\frac{\partial \mathcal{L}}{\partial c_{i,n}} = \frac{2}{\kappa \sigma_{i,n}^4} \times \left\{ \sum_{j=1}^{n} w_{ij,n} c_{i,n} - \sum_{j=1}^{n} w_{ij,n} x_j \right\} = 0, \\
&Therefore, \\
&\sum_{j=1}^{n} w_{ij,n} c_{i,n} = \sum_{j=1}^{n} w_{ij,n} x_j \\
&c_{i,n} = \frac{\sum_{j=1}^{n} w_{ij,n} x_j}{\sum_{j=1}^{n} w_{ij,n}}
\end{aligned}
\tag{A.3}
$$

Given the previous centroids, $c_{i,n-1}$, and assuming that the scales do not change much relative to the scale that resulted from the previous point, the new centroid that optimizes (15) in the manuscript after the arrival of the n^{th} data point, and by penalizing the previous information as in (11) in the manuscript, is given by [22]:

$$
\begin{aligned}
c_{i,n} &= \frac{\sum_{j=1}^{n} w_{ij,n} x_j}{\sum_{j=1}^{n} w_{ij,n}} \\
c_{i,n} &= \frac{\sum_{j=1}^{n-1} w_{ij,n} x_j + w_{in,n} x_n}{\sum_{j=1}^{n} w_{ij,n} + w_{in,n}} \\
c_{i,n} &= \frac{\sum_{j=1}^{n-1} (e^{\frac{-1}{\tau}} w_{ij,n-1}) x_j + w_{in,n} x_n}{\sum_{j=1}^{n} (e^{\frac{-1}{\tau}} w_{ij,n-1}) + w_{in,n}} \\
c_{i,n} &= \frac{e^{\frac{-1}{\tau}} \sum_{j=1}^{n-1} w_{ij,n-1} x_j + w_{in,n} x_n}{e^{\frac{-1}{\tau}} \sum_{j=1}^{n} w_{ij,n-1} + w_{in,n}} \\
c_{i,n} &= \frac{e^{\frac{-1}{\tau}} \left(c_{i,n-1} \sum_{j=1}^{n} w_{ij,n-1} \right) + w_{in,n} x_n}{e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n} w_{ij,n-1} \right) + w_{in,n}} \\
c_{i,n} &= \frac{e^{\frac{-1}{\tau}} c_{i,n-1} W_{i,n-1} + w_{in,n} x_n}{e^{\frac{-1}{\tau}} W_{i,n-1} + w_{in,n}}
\end{aligned}
\tag{A.4}
$$

□

Theorem A.1.2. *Optimal Incremental Scale Update: Given the previous scale $\sigma_{i,n-1}^2$, the new scale that optimizes the density function in the manuscript after the arrival of the n^{th} data point is given by [22]:*

$$\sigma_{i,n}^2 = \frac{\kappa e^{\frac{-1}{\tau}} \left(\sigma_{i,n-1}^2 W_{i,n-1}\right) + w_{in,n} d_{ij}^2}{\kappa \left(e^{\frac{-1}{\tau}} W_{i,n-1} + w_{in,n}\right)} \tag{A.5}$$

Proof. For the cluster C_i at time n, we find the derivative of the Lagrangian of the density $\delta_{i,n}$ with respect to the centroid $\sigma_{i,n}^2$, while all other parameters are held constant, giving

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \sigma_{i,n}^2} &= \sum_{j=1}^{n} \frac{\frac{\partial w_{ij,n}}{\partial \sigma_{i,n}^2} \times \sigma_{i,n}^2 - w_{ij,n} \times \frac{\partial (\sigma_{i,n}^2)}{\partial \sigma_{i,n}^2}}{(\sigma_{i,n}^2)^2} - \frac{\partial \left(\lambda \sigma_{i,n}^2\right)}{\partial \sigma_{i,n}^2} \\
&= \sum_{j=1}^{n} \frac{w_{ij,n} \times \frac{\partial (\frac{-d_{ij}^2}{\kappa \sigma_{i,n}^2})}{\partial \sigma_{i,n}^2} \times \sigma_{i,n}^2 - w_{ij,n} \times 1}{\sigma_{i,n}^4} - \lambda \\
&= \sum_{j=1}^{n} \frac{w_{ij,n} \times \left\{ \frac{\frac{\partial (-d_{ij}^2)}{\partial \sigma_{i,n}^2} \times \kappa \sigma_{i,n}^2 - \frac{\partial (\kappa \sigma_{i,n}^2)}{\partial \sigma_{i,n}^2} \times -d_{ij}^2}{(\kappa \sigma_{i,n}^2)^2} \right\} \times \sigma_{i,n}^2 - w_{ij,n}}{\sigma_{i,n}^4} - \lambda \\
&= \sum_{j=1}^{n} \frac{w_{ij,n} \times \left\{ \frac{0 \times \kappa \sigma_{i,n}^2 - \kappa \times -d_{ij}^2}{\kappa^2 \sigma_{i,n}^4} \right\} \times \sigma_{i,n}^2 - w_{ij,n}}{\sigma_{i,n}^4} - \lambda \\
&= \sum_{j=1}^{n} \frac{\frac{w_{ij,n} d_{ij}^2}{\kappa \sigma_{i,n}^2} - w_{ij.n}}{\sigma_i^4} - \lambda = 0
\end{aligned} \tag{A.6}$$

The Karush-Kuhn-Tucker conditions are necessary (but not sufficient) for the scale to be maximum. The conditions are:

$$\lambda \geq 0 \ \& \ \lambda \sigma_{i.n}^2 = 0 \tag{A.7}$$

Which means that we have two cases. In the first case, $\lambda = 0$, and the scale can be found by setting the gradient

$$\sum_{j=1}^{n} \frac{\frac{w_{ij,n} d_{ij}^2}{\kappa \sigma_{i,n}^2} - w_{ij.n}}{\sigma_i^4} - \lambda = \sum_{j=1}^{n} \frac{\frac{w_{ij,n} d_{ij}^2}{\kappa \sigma_{i,n}^2} - w_{ij.n}}{\sigma_i^4} = 0 \tag{A.8}$$

Thus

$$\begin{aligned}
\sum_{j=1}^{n} \frac{w_{ij,n} d_{ij}^2}{\kappa \sigma_{i,n}^2} &= \sum_{j=1}^{n} w_{ij,n} \\
\sum_{j=1}^{n} w_{ij,n} d_{ij}^2 &= \kappa \sigma_{i,n}^2 \sum_{j=1}^{n} w_{ij,n} \\
\sigma_{i,n}^2 &= \frac{\sum_{j=1}^{n} w_{ij,n} d_{ij}^2}{\kappa \sum_{j=1}^{n} w_{ij,n}}
\end{aligned} \tag{A.9}$$

In the second case, $\lambda > 0$, thus $\sigma_{i,n}^2 = 0$. Hence, we need to have a test to make sure that the scale does not become zero. This can be done by checking whether the value of λ is non zero, in this case, we set the scale to the initial value (i.e. $\sigma_{i,n}^2 = \sigma_0^2$). The value of λ

is derived from

$$\sum_{j=1}^{n} \frac{\frac{w_{ij,n} d_{ij}^2}{\kappa \sigma_{i,n}^2} - w_{ij.n}}{\sigma_i^4} - \lambda = 0 \tag{A.10}$$

Thus

$$\lambda = \sum_{j=1}^{n} \frac{\frac{w_{ij,n} d_{ij}^2}{\kappa \sigma_{i,n}^2} - w_{ij.n}}{\sigma_i^4} \tag{A.11}$$

Given the previous scales, $\sigma_{i,n-1}^2$, the new scale that optimizes (15) in the manuscript after the arrival of the n^{th} data point, can also be rewritten as follows, which explicitly shows the penalizing effect of the forgetting mechanism on the previous information via the weight decay expression in (11) in the manuscript,

$$\begin{aligned}
\sigma_{i,n}^2 &= \frac{\sum_{j=1}^{n} w_{ij,n} d_{ij}^2}{\kappa \sum_{j=1}^{n} w_{ij,n}} \\
\sigma_{i,n}^2 &= \frac{\sum_{j=1}^{n-1} w_{ij,n} d_{ij}^2 + w_{in,n} d_{ij}^2}{\kappa \left(\sum_{j=1}^{n-1} w_{ij,n} + w_{in,n} \right)} \\
\sigma_{i,n}^2 &= \frac{\sum_{j=1}^{n-1} (e^{\frac{-1}{\tau}} w_{ij,n-1}) d_{ij}^2 + w_{in,n} d_{ij}^2}{\kappa \left(\sum_{j=1}^{n-1} (e^{\frac{-1}{\tau}} w_{ij,n-1}) + w_{in,n} \right)} \\
\sigma_{i,n}^2 &= \frac{e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} d_{ij}^2 \right) + w_{in,n} d_{ij}^2}{\kappa \left(e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} \right) + w_{in,n} \right)} \\
\sigma_{i,n}^2 &= \frac{e^{\frac{-1}{\tau}} \left(2\sigma_{i,n-1}^2 \sum_{j=1}^{n-1} w_{ij,n-1} \right) + w_{in,n} d_{ij}^2}{\kappa \left(e^{\frac{-1}{\tau}} \left(\sum_{j=1}^{n-1} w_{ij,n-1} \right) + w_{in,n} \right)} \\
\sigma_{i,n}^2 &= \frac{\kappa e^{\frac{-1}{\tau}} \left(\sigma_{i,n-1}^2 W_{i,n-1} \right) + w_{in,n} d_{ij}^2}{\kappa \left(e^{\frac{-1}{\tau}} W_{i,n-1} + w_{in,n} \right)}
\end{aligned} \tag{A.12}$$

□

Theorem A.1.3. *For a cluster C_i, which was valid at time step t, it is stored for a maximum of m time steps if it was inactive, where m is bounded as follows [22]:*

$$m > -\tau \, ln(\frac{\delta_{min}}{\delta_{i,t}}) \tag{A.13}$$

Proof. For simplicity, assume that cluster C_i was valid at time stamp t. Assuming no new points are assigned to C_i during m time stamps, after time stamp t, the density is given by

$$\delta_{i,m} = \sum_{j=1}^{m} \frac{w_{ij,m}}{\sigma_{i,m}^2}$$
$$\delta_{i,m} = \sum_{j=1}^{m} \frac{e^{(-\frac{1}{\tau})^m} w_{ij,t}}{\sigma_{i,m}^2}$$
$$\delta_{i,m} = \frac{e^{(-\frac{1}{\tau})^m}}{\sigma_{i,m}^2} \sum_{j=1}^{m} w_{ij,t}$$

Hence,

$$\sigma_{i.m}^2 = \frac{e^{-\frac{m}{\tau}}}{\delta_{i,m}} \sum_{j=1}^{m} w_{ij,t} \tag{A.14}$$

Since C_i does not get updated for m steps, its scale at time m decreases over time using the update equation, and hence:

$$\sigma_{i,m}^2 \leq \sigma_{i,t}^2$$
$$\frac{e^{-\frac{m}{\tau}}}{\delta_{i,m}} \sum_{j=1}^{m} w_{ij,t} \leq \sigma_{i,t}^2$$
$$e^{-\frac{m}{\tau}} \frac{\sum_{j=1}^{m} w_{ij,t}}{\sigma_{i,t}^2} \leq \delta_{i,m}$$
$$e^{-\frac{m}{\tau}} \delta_{i,t} \leq \delta_{i,m}$$

Cluster C_i will become invalid and thus be eliminated when $\delta_{i,m} = \delta_{min}$, and by using (15):

$$e^{-\frac{m}{\tau}} \delta_{i,t} \leq \delta_{min}$$
$$e^{-\frac{m}{\tau}} \leq \frac{\delta_{min}}{\delta_{i,t}}$$
$$-\frac{m}{\tau} \leq ln(\frac{\delta_{min}}{\delta_{i,t}})$$
$$m > -\tau\, ln(\frac{\delta_{min}}{\delta_{i,t}})$$

□

Theorem A.1.4. *Optimal Incremental Centroid Update : Given the previous centroids, $c_{i,n-1}$, and assuming that the scales do not change much relative to the scale that resulted from the previous iteration, the new centroid that optimizes (3.8) in the manuscript after the arrival of the n^{th} data point is given by:*

$$c_{i,n} = \frac{c_{i,n-1}W_{i,n-1} + w_{in,n}f_{in,n}x_n}{W_{i,n-1} + w_{in,n}f_{in,n}} \tag{A.15}$$

Proof.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial c_{i,n}} &= \frac{\partial\left(\sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma_{i,n}^2} \times \frac{f_{ij,n}}{\tau_{i,n}^2}\right)}{\partial c_{i,n}} - \frac{\partial\left(\lambda\sigma_i^2\right)}{\partial c_{i,n}} - \frac{\partial\left(\lambda\tau_{i,n}^2\right)}{\partial c_{i,n}} \\
&= \frac{\partial\left(\sum_{j=1}^{n} \frac{e^{\left(\frac{-d_{ij}^2}{\kappa\sigma_{i,n}^2}\right)}}{\sigma_{i,n}^2} \times \frac{f_{ij,n}}{\tau_{i,n}^2}\right)}{\partial c_{i,n}} - 0 - 0 \\
&= \frac{f_{ij,n}}{\sigma_{i,n}^2\tau_{i,n}^2} \times \frac{\partial\left(\sum_{j=1}^{n} e^{\left(\frac{-d_{ij}^2}{\kappa\sigma_{i,n}^2}\right)}\right)}{\partial c_{i,n}} \\
&= \frac{f_{ij,n}}{\sigma_{i,n}^2\tau_{i,n}^2} \times \sum_{j=1}^{n} w_{ij,n}\left(\frac{-2(x_j - c_{i,n}) \times \kappa\sigma_{i,n}^2 - 0 \times (x_j - c_{i,n})^T (x_j - c_{i,n})}{\kappa^2\sigma_{i,n}^4}\right) \\
&= \frac{f_{ij,n}}{\sigma_{i,n}^2\tau_{i,n}^2} \times \sum_{j=1}^{n} \frac{-2w_{ij,n}(x_j - c_{i,n})}{\kappa\sigma_{i,n}^2} \\
&= \frac{2 \times f_{ij,n}}{\kappa\sigma_{i,n}^4\tau_{i,n}^2} \times \left(\sum_{j=1}^{n} w_{ij,n}c_{i,n} - \sum_{j=1}^{n} w_{ij,n}x_j\right) = 0
\end{aligned}$$

Therefore,

$$
\begin{aligned}
c_{i,n} &= \frac{\sum_{j=1}^{n} w_{ij,n} f_{ij,n} x_j}{\sum_{j=1}^{n} w_{ij,n} f_{ij,n}} \\
&= \frac{\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} x_j + w_{in,n} f_{in,n} x_n}{\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n}} \\
&= \frac{c_{i,n-1} \sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} x_n}{\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n}} \\
&= \frac{c_{i,n-1} \sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} x_n}{\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n}} \\
&= \frac{c_{i,n-1} W_{i,n-1} + w_{in,n} f_{in,n} x_n}{W_{i,n-1} + w_{in,n} f_{in,n}}
\end{aligned}
$$

□

Theorem A.1.5. *Optimal Incremental Temporal Centroid Update : Given the previous temporal centroids, $c_{i,n-1}$, and assuming that the scales do not change much relative to the scale that resulted from the previous iteration, the new temporal centroid that optimizes (3.8) in the manuscript after the arrival of the n^{th} data point at time t_n is given by:*

$$
c'_{i,n} = \frac{c'_{i,n-1} W_{i,n-1} + w_{in,n} f_{in,n} t_n}{W_{i,n-1} + w_{in,n} f_{in,n}} \tag{A.16}
$$

Proof.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial c'_{i,n}} &= \frac{\partial\left(\sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma^2_{i,n}} \times \frac{f_{ij,n}}{\tau^2_{i,n}}\right)}{\partial c'_{i,n}} - \frac{\partial\left(\lambda\sigma_i^2\right)}{\partial c'_{i,n}} - \frac{\partial\left(\lambda\tau^2_{i,n}\right)}{\partial c'_{i,n}} \\
&= \frac{\partial\left(\sum_{j=1}^{n} \frac{e^{\left(\frac{-\eta^2_{ij}}{\rho\tau^2_{i,n}}\right)}}{\tau^2_{i,n}} \times \frac{w_{ij,n}}{\sigma^2_{i,n}}\right)}{\partial c'_{i,n}} - 0 - 0 \\
&= \frac{w_{ij,n}}{\sigma^2_{i,n}\tau^2_{i,n}} \times \frac{\partial\left(\sum_{j=1}^{n} e^{\left(\frac{-\eta^2_{ij}}{\rho\tau^2_{i,n}}\right)}\right)}{\partial c'_{i,n}} \\
&= \frac{w_{ij,n}}{\sigma^2_{i,n}\tau^2_{i,n}} \times \sum_{j=1}^{n} f_{ij,n}\left(\frac{-2(t_j - c'_{i,n}) \times \rho\tau^2_{i,n} - 0 \times \left(t_j - c'_{i,n}\right)^T\left(t_j - c'_{i,n}\right)}{\rho^2\tau^4_{i,n}}\right) \\
&= \frac{w_{ij,n}}{\sigma^2_{i,n}\tau^2_{i,n}} \times \sum_{j=1}^{n} \frac{-2f_{ij,n}(t_j - c'_{i,n})}{\rho\tau^2_{i,n}} \\
&= \frac{2 \times w_{ij,n}}{\rho\sigma^4_{i,n}\tau^2_{i,n}} \times \left(\sum_{j=1}^{n} f_{ij,n}c'_{i,n} - \sum_{j=1}^{n} f_{ij,n}t_j\right) = 0
\end{aligned}$$

Therefore,

$$\begin{aligned}
c'_{i,n} &= \frac{\sum_{j=1}^{n} w_{ij,n}f_{ij,n}t_j}{\sum_{j=1}^{n} w_{ij,n}f_{ij,n}} \\
&= \frac{\sum_{j=1}^{n-1} w_{ij,n-1}f_{ij,n-1}t_j + w_{in,n}f_{in,n}t_n}{\sum_{j=1}^{n-1} w_{ij,n-1}f_{ij,n-1} + w_{in,n}f_{in,n}} \\
&= \frac{c'_{i,n-1}\sum_{j=1}^{n-1} w_{ij,n-1}f_{ij,n-1} + w_{in,n}f_{in,n}t_n}{\sum_{j=1}^{n-1} w_{ij,n-1}f_{ij,n-1} + w_{in,n}f_{in,n}} \\
&= \frac{c'_{i,n-1}\sum_{j=1}^{n-1} w_{ij,n-1}f_{ij,n-1} + w_{in,n}f_{in,n}t_n}{\sum_{j=1}^{n-1} w_{ij,n-1}f_{ij,n-1} + w_{in,n}f_{in,n}} \\
&= \frac{c'_{i,n-1}W_{i,n-1} + w_{in,n}f_{in,n}t_n}{W_{i,n-1} + w_{in,n}f_{in,n}}
\end{aligned}$$

□

Theorem A.1.6. *Optimal Incremental Scale Update: Given the previous scale $\sigma^2_{i,n-1}$, the new scale that optimizes (3.8) in the manuscript after the arrival of the n^{th} data point is given by:*

$$\sigma_{i,n}^2 = \frac{\kappa\sigma_{i,n-1}^2 W_{i,n-1} + w_{in,n} f_{in,n} d_{ij}^2}{\kappa\left(W_{i,n-1} + w_{in,n} f_{in,n}\right)} \tag{A.17}$$

Proof.

$$\begin{aligned}
\frac{\partial \delta_{i,n}}{\partial \sigma_{i,n}^2} &= \frac{\partial\left(\sum_{j=1}^{n} \frac{f_{ij,n} \times e^{\left(\frac{-d_{ij}^2}{\kappa\sigma_{i,n}^2}\right)}}{\sigma_{i,n}^2 \tau_{i,n}^2} - \lambda\sigma_i^2 - \lambda\tau_{i,n}^2\right)}{\partial\sigma_{i,n}^2} \\
&= \frac{\partial\left(\sum_{j=1}^{n} \frac{f_{ij,n} \times e^{\left(\frac{-d_{ij}^2}{\kappa\sigma_{i,n}^2}\right)}}{\sigma_{i,n}^2 \tau_{i,n}^2}\right)}{\partial\sigma_{i,n}^2} - \frac{\partial\lambda\sigma_i^2}{\partial\sigma_{i,n}^2} - \frac{\partial\lambda\tau_{i,n}^2}{\partial\sigma_{i,n}^2} \\
&= \frac{\left(\sum_{j=1}^{n} \frac{f_{ij,n}}{\tau_{i,n}^2} \times \left(w_{ij,n} \times \frac{\partial\left(\frac{-d_{ij}^2}{\kappa\sigma_{i,n}^2}\right)}{\partial\sigma_{i,n}^2} \times \sigma_{i,n}^2 - w_{ij,n} \times 1\right)\right)}{\sigma_{i,n}^4} - \lambda \\
&= \frac{\left(\sum_{j=1}^{n} \frac{f_{ij,n}}{\tau_{i,n}^2} \times \left(w_{ij,n} \times \frac{\frac{\partial\left(-d_{ij}^2\right)}{\partial\sigma_{i,n}^2} \times \kappa\sigma_{i,n}^2 - \frac{\partial\left(\kappa\sigma_{i,n}^2\right)}{\partial\sigma_{i,n}^2} \times \left(-d_{ij}^2\right)}{\left(\kappa\sigma_{i,n}^2\right)^2} \times \sigma_{i,n}^2 - w_{ij,n} \times 1\right)\right)}{\sigma_{i,n}^4} - \lambda \\
&= \frac{\left(\sum_{j=1}^{n} \frac{f_{ij,n}}{\tau_{i,n}^2} \times \left(w_{ij,n} \times \frac{0 \times \kappa\sigma_{i,n}^2 - \kappa \times \left(-d_{ij}^2\right)}{\left(\kappa\sigma_{i,n}^2\right)^2} \times \sigma_{i,n}^2 - w_{ij,n}\right)\right)}{\sigma_{i,n}^4} - \lambda \\
&= \frac{\left(\sum_{j=1}^{n} \frac{f_{ij,n}}{\tau_{i,n}^2} \times \left(\frac{w_{ij,n} d_{ij}^2}{\kappa\sigma_{i,n}^2} - w_{ij,n}\right)\right)}{\sigma_{i,n}^4} - \lambda = 0
\end{aligned}$$

The Karush-Kuhn-Tucker conditions are necessary (but not sufficient) for the scale to be maximum. The conditions are:

$$\lambda \geq 0 \;\&\; \lambda\sigma_{i.n}^2 = 0 \tag{A.18}$$

Which means that we have two cases. In the first case, $\lambda = 0$, and the scale can be found by setting the gradient. In the second case, $\lambda > 0$, thus $\sigma_{i,n}^2 = 0$. Hence, we need to have a test to make sure that the scale does not become zero.

When $\lambda = 0$, we get:

$$
\begin{aligned}
\sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n}}{\tau_{i,n}^2} &= \sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n} d_{ij}^2}{\kappa \tau_{i,n}^2 \sigma_{i,n}^2} \\
\sigma_{i,n}^2 &= \frac{\sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n} d_{ij}^2}{\kappa \tau_{i,n}^2}}{\sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n}}{\tau_{i,n}^2}} \\
&= \frac{\sum_{j=1}^{n} w_{ij,n} f_{ij,n} d_{ij}^2}{\kappa \sum_{j=1}^{n} w_{ij,n} f_{ij,n}} \\
&= \frac{\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} d_{ij}^2 + w_{in,n} f_{in,n} d_{ij}^2}{\kappa \left(\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} \right)} \\
&= \frac{2\sigma_{i,n-1}^2 \sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} d_{ij}^2}{\kappa \left(\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} \right)} \\
&= \frac{\kappa \sigma_{i,n-1}^2 W_{i,n-1} + w_{in,n} f_{in,n} d_{ij}^2}{\kappa \left(W_{i,n-1} + w_{in,n} f_{in,n} \right)}
\end{aligned}
$$

□

Theorem A.1.7. *Optimal Incremental Temporal Scale Update: Given the previous scale* $\tau_{i,n-1}^2$*, the new scale that optimizes (3.8) in the manuscript after the arrival of the* n^{th} *data point is given by:*

$$
\tau_{i,n}^2 = \frac{\rho \tau_{i,n-1}^2 W_{i,n-1} + w_{in,n} f_{in,n} \eta_{ij}^2}{\rho \left(W_{i,n-1} + w_{in,n} f_{in,n} \right)} \tag{A.19}
$$

Proof.

$$
\begin{aligned}
\frac{\partial \delta_{i,n}}{\partial \tau_{i,n}^2} &= \frac{\partial \left(\sum_{j=1}^{n} \frac{w_{ij,n} \times e^{\left(\frac{-\eta_{ij}^2}{\rho \tau_{i,n}^2} \right)}}{\sigma_{i,n}^2 \tau_{i,n}^2} - \lambda \sigma_i^2 - \lambda \tau_{i,n}^2 \right)}{\partial \tau_{i,n}^2} \\
&= \frac{\partial \left(\sum_{j=1}^{n} \frac{w_{ij,n} \times e^{\left(\frac{-\eta_{ij}^2}{\rho \tau_{i,n}^2} \right)}}{\sigma_{i,n}^2 \tau_{i,n}^2} \right)}{\partial \tau_{i,n}^2} - \frac{\partial \lambda \sigma_i^2}{\partial \tau_{i,n}^2} - \frac{\partial \lambda \tau_{i,n}^2}{\partial \tau_{i,n}^2} \\
&= \frac{\left(\sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma_{i,n}^2} \times \left(f_{ij,n} \times \frac{\partial \left(\frac{-\eta_{ij}^2}{\rho \tau_{i,n}^2} \right)}{\partial \tau_{i,n}^2} \times \tau_{i,n}^2 - f_{ij,n} \times 1 \right) \right)}{\tau_{i,n}^4} - \lambda \\
&= \frac{\left(\sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma_{i,n}^2} \times \left(f_{ij,n} \times \frac{\frac{\partial \left(-\eta_{ij}^2 \right)}{\partial \tau_{i,n}^2} \times \rho \tau_{i,n}^2 - \frac{\partial \left(\rho \tau_{i,n}^2 \right)}{\partial \tau_{i,n}^2} \times \left(-\eta_{ij}^2 \right)}{\left(\rho \tau_{i,n}^2 \right)^2} \times \tau_{i,n}^2 - f_{ij,n} \times 1 \right) \right)}{\tau_{i,n}^4} - \lambda \\
&= \frac{\left(\sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma_{i,n}^2} \times \left(f_{ij,n} \times \frac{0 \times \rho \tau_{i,n}^2 - \rho \times \left(-\eta_{ij}^2 \right)}{\left(\rho \tau_{i,n}^2 \right)^2} \times \tau_{i,n}^2 - f_{ij,n} \right) \right)}{\tau_{i,n}^4} - \lambda \\
&= \frac{\left(\sum_{j=1}^{n} \frac{w_{ij,n}}{\sigma_{i,n}^2} \times \left(\frac{f_{ij,n} \eta_{ij}^2}{\rho \tau_{i,n}^2} - f_{ij,n} \right) \right)}{\tau_{i,n}^4} - \lambda = 0
\end{aligned}
$$

The Karush-Kuhn-Tucker conditions are necessary (but not sufficient) for the scale to be maximum. The conditions are:

$$\lambda \geq 0 \ \& \ \lambda \sigma_{i.n}^2 = 0 \tag{A.20}$$

Which means that we have two cases. In the first case, $\lambda = 0$, and the scale can be found by setting the gradient. In the second case, $\lambda > 0$, thus $\sigma_{i,n}^2 = 0$. Hence, we need to have a test to make sure that the scale does not become zero.

When $\lambda = 0$, we get:

$$
\begin{aligned}
\sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n}}{\sigma_{i,n}^2} &= \sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n} \eta_{ij}^2}{\rho \tau_{i,n}^2 \sigma_{i,n}^2} \\
\tau_{i,n}^2 &= \frac{\sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n} \eta_{ij}^2}{\rho \sigma_{i,n}^2}}{\sum_{j=1}^{n} \frac{w_{ij,n} f_{ij,n}}{\sigma_{i,n}^2}} \\
&= \frac{\sum_{j=1}^{n} w_{ij,n} f_{ij,n} \eta_{ij}^2}{\rho \sum_{j=1}^{n} w_{ij,n} f_{ij,n}} \\
&= \frac{\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} \eta_{ij}^2 + w_{in,n} f_{in,n} \eta_{ij}^2}{\rho \left(\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} \right)} \\
&= \frac{2\tau_{i,n-1}^2 \sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} \eta_{ij}^2}{\rho \left(\sum_{j=1}^{n-1} w_{ij,n-1} f_{ij,n-1} + w_{in,n} f_{in,n} \right)} \\
&= \frac{\rho \tau_{i,n-1}^2 W_{i,n-1} + w_{in,n} f_{in,n} \eta_{ij}^2}{\rho \left(W_{i,n-1} + w_{in,n} f_{in,n} \right)}
\end{aligned}
$$

□

A.2 ANOVA Tables

TABLE A.1

ANOVA Table for RBF Parameter Sensitivity on Davies-Bouldin Index Evaluation Metric.

Source	Sum Sq. Errors.	Degrees of Freedom	Mean Sq.	F-statistic	p-value
Event Frequency	0.148	2	0.074	0.027	0.973
No. Clusters	2164.019	3	721.340	267.580	0.000
Noise	1251.508	1	1251.508	462.937	0.000
Radii	6660.964	1	6660.964	2506.008	0.000
No. Dimensions	99519.243	2	49759.621	26489.519	0.000
Density Range	62.302	1	62.302	22.961	0.000
Random Seed	9801.957	2	4900.978	1862.321	0.000
Error	203816.162	119107	1.71120221		
Total	323276.303	119119			

TABLE A.2

ANOVA Table for RBF Parameter Sensitivity on Silhouette Index Evaluation Metric.

Source	Sum Sq. Errors.	Degrees of Freedom	Mean Sq.	F-statistic	p-value
Event Frequency	0.011	2	0.006	0.112	0.894
No. Clusters	357.420	3	119.140	2482.139	0.000
Noise	12.612	1	12.612	247.809	0.000
Radii	128.219	1	128.219	2568.378	0.000
No. Dimensions	3396.926	2	1698.463	75549.211	0.000
Density Range	0.000	1	0.000	0.001	0.972
Random Seed	13.914	2	6.957	136.727	0.000
Error	2165.76	119107	0.018		
Total	6074.859	119119			

TABLE A.3

ANOVA Table for RBF Parameter Sensitivity on V-Measure Evaluation Metric.

Source	Sum Sq. Errors.	Degrees of Freedom	Mean Sq.	F-statistic	p-value
Event Frequency	0.016	2	0.008	0.094	0.911
No. Clusters	535.377	3	178.459	2236.336	0.000
Noise	26.709	1	26.709	317.707	0.000
Radii	165.754	1	165.754	1999.407	0.000
No. Dimensions	5964.429	2	2982.215	87144.346	0.000
Density Range	0.047	1	0.047	0.558	0.455
Random Seed	160.384	2	80.192	966.787	0.000
Error	3188.082	119107	0.027		
Total	10040.798	119119			

TABLE A.4

ANOVA Table for AFTER-Streams Parameter Sensitivity on Davies-Bouldin Index Evaluation Metric.

Source	Sum Sq. Errors.	Degrees of Freedom	Mean Sq.	F-statistic	p-value
K_{max}	6898.274	2	3449.137	1298.607	0.000
σ_0	7693.783	1	7693.783	2904.052	0.000
τ_0	1.900	2	0.950	0.350	0.705
$\frac{1}{t_{outlier}}$	164.165	1	164.165	60.521	0.000
$\frac{1}{t_{merging}}$	1.737	1	1.737	0.640	0.424
a_{mature}	4.347	1	4.347	1.602	0.206
Error	308512.097	119111	2.590		
Total	323276.303	119119			

TABLE A.5

ANOVA Table for AFTER-Streams Parameter Sensitivity on Silhouette Index Evaluation Metric.

Source	Sum Sq. Errors.	Degrees of Freedom	Mean Sq.	F-statistic	p-value
K_{max}	174.173	2	87.087	1758.016	0.000
σ_0	303.473	1	303.473	6263.504	0.000
τ_0	0.181	2	0.091	1.779	0.169
$\frac{1}{t_{outlier}}$	6.765	1	6.765	132.806	0.000
$\frac{1}{t_{merging}}$	0.034	1	0.034	0.671	0.413
a_{mature}	0.002	1	0.002	0.034	0.854
Error	5590.231	119111	0.047		
Total	6074.859	119119			

TABLE A.6

ANOVA Table for AFTER-Streams Parameter Sensitivity on V-Measure Evaluation Metric.

Source	Sum Sq. Errors.	Degrees of Freedom	Mean Sq.	F-statistic	p-value
K_{max}	26.708	2	13.354	158.845	0.000
σ_0	1268.712	1	1268.712	17228.110	0.000
τ_0	0.656	2	0.328	3.889	0.020
$\frac{1}{t_{outlier}}$	149.272	1	149.272	1797.598	0.000
$\frac{1}{t_{merging}}$	4.020	1	4.020	47.714	0.000
a_{mature}	0.001	1	0.001	0.016	0.899
Error	8591.429	119111	0.072		
Total	10040.798	119119			

CURRICULUM VITAE

NAME: Gopi Chand Nutakki

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.S., Computer Science
December 2010
Western Kentucky University, *Bowling Green, Kentucky*

B.Tech., Information Technology
April 2007
Acharya Nagarjuna University, *Vijayawada, India*

CONFERENCE PUBLICATIONS:

1. **Nutakki, G. C.**, Nasraoui, O. (2017, December). AFTER-Streams: A stream clustering algorithm. Pattern recognition. **(journal yet to be submitted)**

2. **Nutakki, G. C.**, Nasraoui, O. (2017, December). A Compartmentalized Framework for adaptive topic mining. Pattern recognition. **(journal yet to be submitted)**

3. **Nutakki, G. C.**, Nasraoui, O. (2017, June). Clustering Data Streams with Adaptive Forgetting. In Big Data (BigData Congress), 2017 IEEE International Congress on (pp. 494-497). IEEE.

4. **Nutakki, G. C.**, Nasraoui, O. (2016, December). Compartmentalized adaptive topic mining on social media streams. In Big Data (Big Data), 2016 IEEE International Conference on (pp. 992-997). IEEE.

5. Sanders, W. S., **Nutakki, G. C.**, Nasraoui, O. (2016, July). Testing the Application of Warranting Theory to Online Third Party Marketplaces: The Effects of Information Uniqueness and Product Type. In Proceedings of the 7th 2016 International Conference on Social Media & Society (p. 16). ACM.

6. Abdollahi, B., Badami, M., **Nutakki, G. C.**, Sun, W., Nasraoui, O. (2014, October). A two step ranking solution for twitter user engagement. In Proceedings of the 2014 Recommender Systems Challenge (p. 35). ACM. Chicago

7. **Nutakki, G. C.**, Nasraoui, O. (2014, June). Online sentiment-based topic modeling for continuous data streams. In Proceedings of the 2014 ACM conference on Web science (pp. 259-260). ACM.

8. **Nutakki, G. C.**, Nasraoui, O., Abdollahi, B., Badami, M., Sun, W. (2014). Distributed LDA-based Topic Modeling and Topic Agglomeration in a Latent Space. In SNOW-DC WWW (pp. 17-24).