FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

TweeProfiles4: a weighted multidimensional stream clustering algorithm

Luís Miguel Azevedo Pereira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Soares (PhD)

July 30, 2015

TweeProfiles4: a weighted multidimensional stream clustering algorithm

Luís Miguel Azevedo Pereira

Mestrado Integrado em Engenharia Informática e Computação

July 30, 2015

Abstract

The emergence of social media made it possible for users to easily share their thoughts on different topics, which constitutes a rich source of information for many fields. Microblogging platforms experienced a large and steady growth over the last few years. Twitter is the most popular microblogging site, making it an interesting source of data for pattern extraction. One of the main challenges of analyzing social media data is its continuous nature, which makes it hard to use traditional data mining approaches. Therefore, mining stream data has also received a lot of attention recently.

TweeProfiles is a data mining tool for analyzing and visualizing Twitter data over four dimensions: spatial (the location of the tweet), temporal (the timestamp of the tweet), content (the text of the tweet) and social (relationships graph). This is an ongoing project with many interesting challenges. For instance, it was recently improved by replacing the original clustering algorithm which could not handle the continuous flow of data with a streaming method.

The goal of this dissertation is to continue the development of TweeProfiles. First, the stream clustering process is improved by proposing a new algorithm. The new algorithm is incremental and supports multi-dimensional streaming data. Moreover, it allows the user to dynamically change the relative importance of each dimension in the clustering. Additionally, a more thorough empirical evaluation is carried out using suitable measures to evaluate the extracted patterns.

The proposed algorithm has been applied in the context of Twitter data and has been evaluated in both quantitative and qualitative terms. Its performance has also been measured and compared to the approach used in the previous version of the project.

Resumo

O aparecimento das redes sociais abriu aos utilizadores a possibilidade de facilmente partilharem as suas ideias a respeito de diferentes temas, o que constitui uma fonte de informação enriquecedora para diversos campos. As plataformas de *microblogging* sofreram um grande crescimento e de forma constante nos últimos anos. O Twitter é o *site* de *microblogging* mais popular, tornando-se uma fonte de dados interessante para extração de conhecimento. Um dos principais desafios na análise de dados provenientes de redes sociais é o seu fluxo, o que dificulta a aplicação de processos tradicionais de *data mining*. Neste sentido, a extração de conhecimento sobre fluxos de dados tem recebido um foco significativo recentemente.

O TweeProfiles é a uma ferramenta de *data mining* para análise e visualização de dados do Twitter sobre quatro dimensões: espacial (a localização geográfica do *tweet*), temporal (a data de publicação do *tweet*), de conteúdo (o texto do *tweet*) e social (o grafo dos relacionamentos). Este é um projeto em desenvolvimento com muitos desafios interessantes. Uma das recentes melhorias inclui a substituição do algoritmo de *clustering* original, o qual não suportava o fluxo contínuo dos dados, por um método de *streaming*.

O objetivo desta dissertação passa pela continuação do desenvolvimento do TweeProfiles. Em primeiro lugar, é proposto um novo algoritmo de *clustering* para fluxos de dados com o objetivo de melhorar o existente. O novo algoritmo é incremental e suporta fluxos de dados multidimensionais. Esta abordagem permite ao utilizador alterar dinamicamente a importância relativa de cada dimensão do processo de *clustering*. Adicionalmente, é feita uma avaliação empírica dos resultados mais completa através da identificação e implementação de medidas adequadas de avaliação dos padrões extraídos.

O algoritmo proposto foi aplicado no contexto do Twitter e foi avaliado tanto em termos quantitativos como qualitativos. O desempenho do mesmo também foi medido e comparado com a abordagem utilizada na versão anterior do projeto.

Acknowledgements

I would like to thank Prof. Carlos Soares for his supervision in this project. His ideas and guidance enabled the accomplishment of the goals of this dissertation.

I also thank Tiago Cunha and André Maia for their support and insights on the theme, which promoted a leaner integration and development process.

I would like to thank my family, specially my parents, for all their support, motivation and for believing in my success during the development of this work.

Finally, I thank my girlfriend, Inês de Sá, and my friends with a special remark for those from college. All of you made this journey a better experience and it would not have been the same without you.

Luís Miguel Azevedo Pereira

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

HMC	Hybrid micro cluster
MC	Micro cluster
PMC	Potential micro cluster
OMC	Outlier micro cluster
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points to Identify the Clustering Structure
REST	Representational state transfer
API	Application programming interface
HTTP	Hypertext Transfer Protocol
NMI	Normalized Mutual Information
TFIDF	Term frequency-inverse document frequency
IDF	Inverse document frequency

Chapter 1

Introduction

Social networks are a source of ever growing data being used by many people to share firsthand information. As such, they are regarded as timely and cost-effective source of spatio-temporal information [Lee12]. These social media services not only influence how individuals communicate from a personal perspective, but also how companies define their marketing strategies.

Twitter is one of the most popular social networking sites. It has not only gained world-wide popularity but has also been increasing its user activity with over 300 million monthly active users generating 500 million tweets per day [Twi15]. It is considered a microblogging platform for its short message broadcasting features, which allied to the proliferation of this service, makes it an interesting instrument for research studies. These include topic summarization [YZF12, SWCC13], event detection [BF10, BHP11, KBQ14] and sentiment analysis [Cor12, Lee12, BNG11, BL12, CTB+12]. Journalism is one of the most affected businesses, taking advantage of social networks to follow trending topics, information spreading and public opinion on several affairs.

TweeProfiles [Cun13] is a data mining tool which allows the analysis and visualization of patterns extracted from Twitter data. The initial version used an offline clustering algorithm to identify patterns over four dimensions: spatial, temporal, social and content. However, it had the shortcoming of only supporting static data and, as such, it was unable to capture emerging trends from a data stream. It was further extended in TweeProfiles2 [Per14], Olhó-Passarinho [Mot14] and TweeProfiles3 [Mai15]. TweeProfiles2 improved the clustering process by introducing an algorithm capable of handling streaming data. Olhó-Passarinho extended the clustering process to consider images as part of the content a tweet, besides the text. TweeProfiles3 focused on improving the visualization of the results and on the integration with SocialBus [BOM+12], a platform for collecting and processing data from social networks for helping researchers build social network datasets.

1.1 Motivation and Objectives

In spite of the progress, some aspects of TweeProfiles can still be improved. As it evolves, it becomes necessary to evaluate the produced results. This is not only important for the ability to validate the clustering approaches, but also because it facilitates tuning the parameters required by the algorithms. Moreover, the user is currently unable to dynamically change the relative importance given to each dimension, since this parametrization is only allowed in the beginning of the process. This makes it difficult for the user to perform sensitivity analysis regarding the weighted combination of the dimensions. Whenever a different combination is required, the clustering algorithm must be restarted from the beginning of the stream, which is not practical.

This dissertation aims to continue the development of TweeProfiles by improving several aspects. The first will be the proposal and implementation of a new algorithm for clustering multi-dimensional streaming data. This novel approach will allow the user to alter the relative weight of each dimension in the clustering process and have results in real-time. The second goal involves the identification and implementation of suitable measures for the evaluation of the resulting clusterings.

An empirical methodology will be performed based on the extracted patterns obtained from the developed platform. The platform will be served input data as tweets acquired from a Twitter data collector.

1.2 Document Structure

This document is organized as follows: Chapter 2 summarizes the state of the art of the scientific fields related to this project, namely: stream clustering algorithms, distance measures, evaluation measures and research done using Twitter. Chapter 3 describes the developed tool in terms of the architecture and explains the clustering and evaluation tasks. Chapter 4 presents the experimental setup and analyzes the obtained results. Chapter 5 concludes the achievements and discusses the work to be done.

Chapter 2

State of the art

In this chapter, the state of the art in the domain of the project is reviewed. Section 2.1 details technical aspects of the clustering process with focus on the streaming paradigm and on the evaluation measures. In Section 2.2, an overview of the similarity functions is provided in the context of multidimensional clustering. Twitter is described in Section 2.3 and an overview of data mining research done on this microblogging platform is covered in section 2.3.3. The current state of TweeProfiles is described in sections 2.4 and 2.5 with focus on the relevant parts for this work.

2.1 Clustering

Data mining is the process of discovering interesting patterns from massive amounts of data [HK06]. This knowledge discovery process comprises the following steps [HK06]:

1. Data cleaning
2. Data integration
3. Data selection
4. Data transformation
5. Data mining
6. Pattern evaluation
7. Knowledge presentation

The first four steps are data preparation tasks which are responsible for making the data for being mined. This process is followed by the application of appropriate algorithms to the data so as to retrieve interesting patterns. These patterns are then assessed based on evaluation measures and, finally, a representation of the mined knowledge is constructed for visualization and decision support purposes.

Clustering is an unsupervised data mining task, whose goal is to group unlabeled data into meaningful groups [JMF99]. The data is partitioned by maximizing the similarity between objects in the same cluster, while minimizing the similarity between objects from distinct clusters. The assessment of the similarity is computed using distance functions, which are explained in more detail in section 2.2. Clustering methods can be classified into five categories [HK06]: partitioning, hierarchical, density-based, grid-based and model-based. In section 2.1.1, stream clustering algorithms are presented according to these categories.

K-Means is one of the most common clustering algorithms, which fits in the partitioning category. This algorithm tries to find partitions such that the squared error between the points in a cluster and its center is minimized. This is known to be a NP-hard problem. Let us consider a dataset X of d-dimensional data points X_i, K clusters C_i with centroids c_i. The squared error is defined as:

$$E = \sum_{i=1}^{K} \sum_{x \in C_i} dist(x, c_i)^2 \tag{2.1}$$

The main steps of the algorithm are the following [Jai10]:

1. Arbitrarily select an initial partition with K clusters.

2. Generate a new partition by assigning each object to its closest cluster center (most similar cluster)

3. Update the cluster centers

Steps 1 and 2 are repeated until a predefined limit number of iterations is reached or until the partitioning does not change in two consecutive iterations. One drawback of the algorithm is the fact that is requires the user to establish the number of clusters K. The minimization of the squared error can only be applied for a fixed number of clusters since the error is inversely proportional to it. As the algorithm only converges to a local minima, it is very sensitive to the initialization performed, which is another disadvantage of this approach.

DBSCAN [EKSX96] is a density-based algorithm. It needs to be supplied two parameters, which are the minimum number of points, *minPts*, and the radius, ε. The algorithm defines core points as those with a dense neighbourhood, which is considered as such when the number of points in the region is greater than *minPts*. These points are iteratively connected to their neighbours whenever the latter are in the core point's ε-neighbourhood. The ε-neighbourhood depends on the ε parameter, since a point is considered to be in the core point's ε-neighbourhood if it is within the user-defined radius. DBSCAN is presented in algorithm 1.

Algorithm 1 DBSCAN

```
1:  procedure DBSCAN(minPts : neighbourhood_threshold,D : dataset,ε : radius)
2:      Mark all objects as unvisited
3:      repeat
4:          Randomly select an unvisited object x
5:          Mark x as visited
6:          if ε-neighborhood of x has at least minPts objects then
7:              Create a new cluster C and add x to C
8:              Let N be the set of objects in the ε-neighborhood of x
9:              for each point x' in N do
10:                 if x' is unvisited then
11:                     Mark x' as visited
12:                     if ε-neighborhood of x' has at least minPts objects then
13:                         Add those points to N
14:                     end if
15:                 end if
16:                 if x' is not a member of any cluster then
17:                     add x' to C
18:                 end if
19:             end for
20:         else
21:             Mark x as noise
22:         end if
23:     until all points are visited
24: end procedure
```

2.1.1 Stream Clustering

In contrast with static data, continuously arriving data streams bring along some challenges given its continuous and dynamic behaviour. These include the volume of the data, its speed and evolution, the existence of noise and outliers and its eventual high-dimensionality, uncertainty and heterogeneous character. In order to address this challenges, stream clustering algorithms need to meet certain requirements. [Bar02] identifies the following requirements:

- **Compactness of representation**: The clusters must be represented in a compact form, so that the memory resources are not exhausted by the increasing number of points processed.

- **Fast, incremental processing of new data points**: Processing new points has to be an efficient task, which means that it cannot be based on comparisons with all the previously considered points.

- **Clear and fast identification of outliers**: Since noise has a great influence on the clusters, it is essential to have an efficient outlier handling mechanism.

The evolution of the data points over time also plays an important role in stream clustering algorithms. In this sense, these algorithms can be classified according to the kind of window model followed. There are three which are commonly used [ZS02]: landmark window model, sliding window model and damped window model. Figure 2.1 [AWS14] presents an overview of these models.

Window Model	Definition	Advantages	Disadvantages
Landmark window model	Analyze the entire history of data stream	Suitable for one-pass clustering algorithms	All the data are equally important and the amount of data inside the window would quickly grow to unprocessable sizes
Sliding window model	Analyze the most recent data points	Suitable for applications where interest exists only in the most recent information like stock marketing	Ignoring part of streams
Fading (damped) window model	Assign different weights to data points	Suitable for applications where old data has an effect on the mining results, and the effect decreases as time goes on diminishing the effect of the old data	Unbounded time window (the window captures all historical data, and its size keeps growing as time elapses)

Figure 2.1: Window models in clustering data streams [AWS14]

In the context of stream clustering, several algorithms have been proposed and some surveys have been conducted [Mah09, SFB+13, Agg13, WHT13]. In the following subsections, some of these algorithms are described. The major algorithms are explained in more detail and an overview of their derivations is provided.

2.1.1.1 Partitioning Clustering

Partitioning clustering algorithms attempt to find mutually exclusive clusters of spherical shape. The grouping is achieved by using distance-based functions and a mean or medoid to represent clusters centers. This type of clustering methods are considered effective for small to medium-sized data sets [HK06]. STREAM [GMM+03] is one of the most popular partitioning algorithms for streaming data. It is a single-pass algorithm, which is based on the k-median problem. The main steps of the algorithm are as follows:

1. Divide the data stream into chunks of m data points each. The value of m is defined according to memory restrictions.

2. A set of k representatives is picked from each chunk so that each data point is assigned to the nearest representative. The representatives are chosen with the goal of minimizing the sum of squared distances of the assigned data points.

3. After each chunk is processed, the set of k medians is stored along with their weights and the data points are discarded. The weight corresponds to the number of points assigned to the representative. These representatives are considered *level*-1 representatives.

4. When the number of representatives exceeds m, these are clustered by taking into account their weights. The representatives that result from this clustering process are considered *level*-2 representatives.

5. When all the original data points are processed or a clustering result is demanded, the remaining representatives of every level are clustered together.

A divide and conquer algorithm is proposed in [GMMO00] and uses a similar approach also based on the k-median problem. The data is divided into chunks and their size is determined so that they can fit in memory. When the data stream is too large, the algorithm recursively calls itself on a smaller set of weighted centers.

CluStream [AWC+03] is a stream clustering algorithm, whose process is divided into two components: online and offline. The former phase clusters data and summarizes it using microclusters, while the latter performs another clustering using the stored summary statistics. A pyramidal time frame is used for storing microclusters at snapshots in time at different levels of granularity depending upon the recency. A microcluster, for a group of points $X_{i_1} \dots X_{i_n}$, with timestamps $T_{i_1} \dots T_{i_n}$, is defined by the tuple $(\overline{CF2^x}, \overline{CF1^x}, CF2^t, CF1^t, n)$, with [AWC+03]:

- n is the number of data points maintained in the microcluster;
- $\overline{CF1^x} = \sum_{j=1}^{n} X_{i_j}$ is the linear sum of the points;
- $\overline{CF2^x} = \sum_{j=1}^{n} X_{i_j}^2$ is the squared sum of the points;
- $CF1^t = \sum_{j=1}^{n} T_{i_j}$ is the linear sum of the timestamps;
- $CF2^t = \sum_{j=1}^{n} T_{i_j}^2$ is the squared sum of the timestamps;

The microclusters are maintained incrementally, since they have additive and subtractive properties. Besides, they can be merged by simply adding their respective features. In the online phase, when a new point arrives, it is either added to an existing cluster or to a new one. This decision depends on the maximum boundary defined for each cluster. If the point falls within the boundary of a cluster, it is merged to that cluster, otherwise it is put in its own cluster. Since the number of microclusters is to be kept constant, the creation of a new cluster requires one of the existing clusters to be removed or merged into another. This is decided according to certain criteria which takes into account the time recency. The offline phase applies a macroclustering process based on k-means, according to user-specified parameters. These are constituted by the time horizon and by the number of desired macroclusters. The macroclusters correspond to high-level clusters, which are computed using the summarized information of the microclusters obtained in the previous phase.

SWClustering is proposed in [ZCQJ08] and uses a cluster feature vector similar to CluStream's. In this vector, the timestamp of the most recent object is also included and a new data

structure, which is distributed in levels, is defined as a collection of these vectors. StreamKM++ [AMR+12] is a k-means algorithm for data streams which is computed in two steps: merge and reduce. The merge step is performed on a data structure which contains sets of object holders, where the data points are inserted as they arrive. The reduce step is performed to reduce the number of objects that result from the previous step, with the information being summarized in a tree-like structure.

HPStream [AHWY04] is proposed as an improvement of CluStream for the context of clustering high-dimensional data. This is achieved based on a projected clustering approach, which is a technique that determines clusters for specific subsets of dimensions. A fading concept is also utilized with the inclusion of decay-based statistics on the microclusters. Proposed in [YZ06], HCluStream is an improvement of CluStream for clustering over heterogeneous data. It adds support for categorical data by adapting the cluster feature vector to include an histogram of the discrete attributes. Moreover, CluStream's clustering algorithm, k-means, is replaced by k-prototype [Hua97], which supports heterogeneous attributes. Similarly, in [RHM10], HCluWin is presented as an extension of CluWin [CCZ07] for clustering on both numerical and categorical data over sliding windows. With the same motivation as HCluStream, [HW10] proposes MCStream as an improvement of CluStream for heterogeneous data. It solves CluStream's shortcoming based on the idea of dimension-oriented distance. In [HLRH10] SWCUStreams is proposed for clustering data streams with uncertainty by improving CluStream. The uncertainty is considered at the attribute level and it is used to quantify the information on each dimension. In the context of data streams with uncertainty, another proposal is presented in [AY08], which introduces the uncertain clustering feature for summarizing the data. This algorithm is termed as UMicro. With the same motivation, LuMicro is proposed in [ZGZ09], introducing a two-phase stream clustering mechanism, which takes into account the uncertainty of the records. Motivated by the fact that the uncertainty aggravates the sparsity property of high-dimensional data, UPStream is proposed in [Agg09]. The presented algorithm's design allows clustering of uncertain data streams with projected clustering, while also considering the evolution of the stream by a decay factor. In [HLHR10], HU-Clustering is proposed for clustering heterogeneous data streams with uncertainty. With a similar approach as HCluStream with respect to the support of heterogeneous data, HU-Clustering improves LuMicro by including a frequency histogram for the categorical attributes.

2.1.1.2 Hierarchical Clustering

Hierarchical algorithms decompose the data in a hierarchy of clusters. These methods can be either agglomerative or divisive. The first approach begins with small clusters and iteratively merges them until a final cluster is obtained which groups all the data. In contrast, the second approach begins with a single cluster and performs splits iteratively in order to obtain groups with more granularity.

BIRCH [ZRL96] is an incremental clustering algorithm which uses an hierarchical data structure, in the form of a height-balanced tree. Each tree node is defined by a cluster feature vector (CF), defined by the tuple (n, LS, SS), with [ZRL96]:

- n is the number of data points;
- $LS = \sum_{j=1}^{n} X_{i_j}$ is the linear sum of the points;
- $SS = \sum_{j=1}^{n} X_{i_j}^2$ is the squared sum of the points;

for a group of data points $X_{i_1} \ldots X_{i_n}$. These structures allow the computation of the cluster center, radius and diameter:

- $c = \frac{LS}{N}$ is the centroid;
- $r = \sqrt{\left(\frac{SS}{N} - \left(\frac{LS}{N}\right)^2\right)}$ is the radius;
- $d = \sqrt{\left(\frac{2N \cdot SS - 2 \cdot LS^2}{N(N-1)}\right)}$ is the diameter;

These structures have incremental and additive properties, which allow not only an object but also two disjoint vectors to be easily merged.

The algorithm starts by building a tree structure with the CF vectors, loading the data into memory. When a new object arrives, it chooses the nearest non-leaf CF entry node, in terms of Euclidean distance, by traversing the tree from the root to the leaves. If the closest leaf is able to absorb the new entry, given a certain size threshold, the CF vector is updated. If merging is not possible, a new CF entry is created.

COBWEB [Fis96] is an incremental clustering algorithm, which uses a category function to build a tree. Each node of the tree contains a probabilistic description that summarizes the objects maintained by it. When a new point arrives, the algorithm descends the tree from the root along a given path and updates the counts in the nodes traversed. It tries to find the best node to assign the new point using the category utility function.

ClusTree [KABS11] is a clustering algorithm that builds a tree with weighted CF vectors. The algorithm performs merge and split operations automatically to adjust the size of the tree. Furthermore, it is able to adapt itself to different stream speeds.

Doubling [CCFM97] is an incremental hierarchical algorithm based on the k-center optimization, which tries to minimize intra-cluster distance. Guided by a lower bound parameter that defines the optimal diameter for the clustering, the algorithm creates and merges clusters in different phases.

2.1.1.3 Density-based Clustering

Density-based algorithms overcome the issue of partitioning algorithms in the sense that the number of clusters does not need to be defined in advance. Besides, this approach allows arbitrarily-shaped clusters, which the latter lack as they only support convex-shaped cluster structures. Outlier detection is another feature that this category of algorithms support.

A comprehensive review on density-based algorithms is done in [AWS14], exploring nineteen approaches as shown in Figure 2.2 [AWS14]. These are divided into microclustering algorithms and grid-based algorithms. The first category summarizes information in microclusters (similarly

to CluStream [AWC+03]), performing the final clustering on this summarized data (offline step). The second group of algorithms maps data points to grids which are created beforehand and which constitute the basis of the final clustering. A review of the latter has been conducted in [AWSY11]. In the survey [AWS14], the merits and limitations of each algorithm are presented and also their ability to overcome some identified challenges, as illustrated in Figure 2.3 [AWS14].

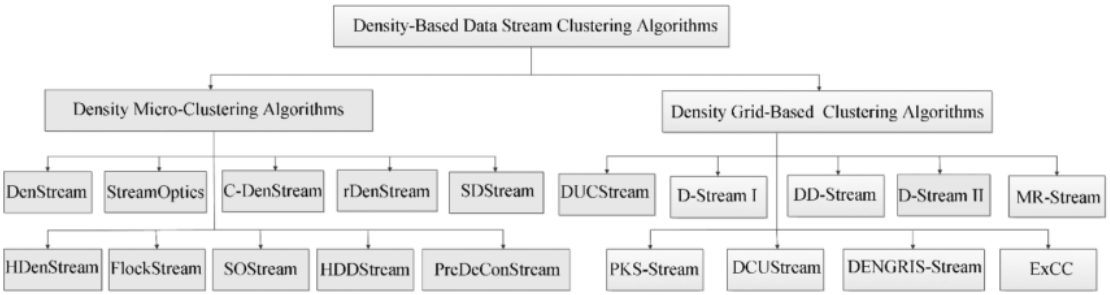


Figure 2.2: Density-based data stream clustering algorithms' categorization [AWS14]

Density-Based Clustering Algorithms	Handling Noisy Data	Handling Evolving Data	Limited Time	Limited Memory	Handling High-Dimensional Data
DenStream	√	√	√	√	-
StreamOptics	√	√	-	-	-
C-DenStream	√	√	-	-	-
rDenStream	√	√	-	-	-
SDStream	√	√	-	√	-
HDenStream	√	√	√	√	-
SOStream	√	√	-	√	-
HDDStream	√	√	-	√	√
PreDeConStream	√	√	-	√	√
FlockStream	√	√	√	-	-
DUCstream	√	-	√	√	-
D-Stream I	√	√	-	-	-
DD-Stream	√	√	-	-	-
D-Stream II	√	√	√	√	-
MR-Stream	√	√	-	-	-
PKS-Stream	√	√	-	-	√
DCUStream	√	√	-	√	-
DENGRIS-Stream	√	√	-	√	-
ExCC	√	√	-	-	-

Figure 2.3: Density-Based Clustering Algorithms and Challenging Issues [AWS14]

An empirical evaluation is also conducted. The following evaluation measures were used: Purity [ZK04], SSQ [HK06], Rand Index [WXC09, Ran71] and NMI [MRS08]. The performance is also compared by taking into account the execution time.

DenStream [CEQZ06] is a two-phase stream clustering algorithm which forms clusters based on dense regions. It defines the concept of a core-micro-cluster, at time t, as $CMC(w,c,r)$ for a group of close points $X_{i_1} \ldots X_{i_n}$ with time stamps $T_{i_1} \ldots T_{i_n}$ [CEQZ06]:

- $w = \sum_{j=1}^{n} f(t - T_{i_j}), w > \mu$ is the weight;
- $c = \frac{\sum_{j=1}^{n} f(t-T_{i_j}) X_{i_j}}{w}$ is the center;

- $r = \frac{\sum_{j=1}^{n} f(t-T_{i_j}) dist(X_{i_j},c)}{w}, r \leq \varepsilon$ is the radius, where $dist(X_{i_j},c)$ denotes the Euclidean distance between the point X_i and the center c;

A damped window model is used by taking into account a fading function given as:

$$f(t) = 2^{-\lambda \cdot t}, \lambda > 0 \tag{2.2}$$

This temporal decay is applied on the microclusters and it gives more weight to newer points.

The concepts of potential core-micro-cluster and outlier microcluster are also defined. They only differ from regular core-micro-clusters in their weight constraint, which affects the merging of points into microclusters. For p-micro-clusters, this constraint is $w \geq \beta \cdot \mu$, and for o-micro-clusters it is $w < \beta \cdot \mu$, with $0 < \beta < 1$.

A p-micro-cluster, at time t, is defined as $PMC(\overline{CF^1}, \overline{CF^2}, w)$ for a group of close points $X_{i_1} \ldots X_{i_n}$ with time stamps $T_{i_1} \ldots T_{i_n}$ [CEQZ06]:

- $w = \sum_{j=1}^{n} f(t-T_{i_j}), w > \beta\mu$ is the weight;
- $\overline{CF^1} = \sum_{j=1}^{n} f(t-T_{i_j}) X_{i_j}$ is the linear sum of the points;
- $\overline{CF^2} = \sum_{j=1}^{n} f(t-T_{i_j}) X_{i_j}^2$ is the squared sum of the points;
- $c = \frac{\overline{CF^1}}{w}$ is the center;
- $r = \sqrt{\frac{\overline{CF^2}}{w} - \left(\frac{\overline{CF^1}}{w}\right)^2}, r \leq \varepsilon$ is the radius;

The microclusters are maintained incrementally by updating $\overline{CF^1}$, $\overline{CF^2}$ and w. If no points are merged on the cluster for time interval δt, $PMC = (f(\delta t)\overline{CF^1}, f(\delta t)\overline{CF^2}, f(\delta t)\dot{w})$. If a point X is merged on the cluster, $PMC = (\overline{CF^1} + p, \overline{CF^2} + p^2, w+1)$.

The pseudo-code of DenStream is presented in algorithm 2. The offline phase generates macro clusters using a variant of DBSCAN. It connects the density regions represented by the microclusters obtained from the online phase.

Several algorithms were proposed as an improvement of DenStream on different aspects. [TRA07] proposed StreamOptics, which extends OPTICS [ABpKS99] and improves DenStream with the goal of allowing the visualization of the cluster structures in data streams. C-DenStream, proposed in [RMS09], improves DenStream by introducing application constraints. These constraints restrict the clustering by establishing data points that must co-exist in a cluster, as well as data points that must be assigned to separate clusters. rDenStream [LxHYfFc09] improves DenStream's outlier handling mechanism for applications where a large amount of these are present. For this purpose, a classifier is applied to the outlier microclusters, allowing discarded data points to be re-learned in an attempt to avoid knowledge points being lost. [RM09] also proposes an improvement over DenStream named SDStream by replacing the damped model of the former with a sliding window model. This is done with the goal of keeping only recent data, while discarding the data points that do not fit in the window's length. HDenStream, proposed in [JH09] improves DenStream by adding support for both numerical and categorical data, since the extended

Algorithm 2 DenStream

procedure DENSTREAM(D,ε,β,μ,λ)
  $T_p = \frac{1}{\lambda}\log(\frac{\beta\mu}{\beta\mu-1})$
  Get the next point X at current time t from data stream D;
  Try to merge X into its nearest p-micro-cluster c_p;
  if $r_p \leq \varepsilon$ **then**
    Merge X into c_p;
  else
    Try to merge X into its nearest o-micro-cluster c_o;
    if $r_o \leq \varepsilon$ **then**
      Merge X into c_o;
      if $w_o > \beta\mu$ **then**
        Remove c_o from outlier-buffer and create a new p-micro-cluster by c_o;
      end if
    else
      Create a new o-micro-cluster by X and insert into the outlier-buffer;
    end if
  end if
  if (t mod T_p) $= 0$ **then**
    for each p-micro-cluster c_p **do**
      if $w_p < \beta\mu$ **then**
        Delete c_p;
      end if
    end for
    for each o-micro-cluster c_o **do**
      $\xi = \frac{2^{-\lambda(t-t_o+T_p)}-1}{2^{-\lambda T_p}-1}$
      if $w_o < \xi$ **then**
        Delete c_o;
      end if
    end for
  end if
  if clustering request arrives **then**
    Generate clusters;
  end if
end procedure

algorithm only supports the former type. The data structure was adapted with the addition of a two-dimensional array that keeps the frequency of the categorical data. This approach is similar to the process that HCluStream used to extend CluStream. MStream, proposed in [WW10], also follows this line of thought, being an algorithm for clustering evolving heterogeneous data streams. Despite being compared to HCluStream and CluStream, it is very similar to HDenStream. This approach also comprises a normalization process on every dimension, which is run at the beginning of the clustering process. DenStream is also improved by SOStream [IDH12], an algorithm developed with the objective of automatically adapting the threshold on density-based clustering. This is achieved by adopting the concept of competitive learning [Koh82]. Both HDDStream [NZP+12] and PreDeConStream [HSGS12] improve DenStream in the context of high-dimensional data by applying the concept of projected clustering. FlockStream [FPS09] proposes an algorithm based on a bio-inspired model with the objective of improving the efficiency of the algorithm by reducing the number of computations performed. LeaDen-Stream [AW13] is an algorithm proposed as an improvement of DenStream for reducing the time complexity of the process. It introduces the concept of mini-micro and micro cluster leaders, which are dependent on the distribution of the data points inside each microcluster. This is motivated by the fact that existing microclustering algorithms ignore the inner distribution, which leads to less accuracy since only the microcluster's centers are sent to the offline phase.

Density grid-based clustering algorithms partition the data space into cells. These cells form grids, which are then clustered according to their density. In this context, several approaches have been proposed, motivated by the ability to merge the advantages of both the density-based algorithms as the ability to detect outliers and provide arbitrarily-shaped clusters, and the grid-based algorithms with the processing time only dependent on the number of grid cells.

In [GLZT05] DUCstream is proposed as a single-pass algorithm for clustering data streams using dense unit detection. The data is processed in chunks and its data points are mapped to dense units, with the clusters being formed based on those which have a larger number of records. The resulting clusters are connected components of a graph composed by the dense units and their relations.

D-Stream is proposed in [CT07], having an online and an offline phase similar to CluStream's, while also handling outliers by means of sporadic grids. These are grids with few objects, which are periodically removed. The online phase is responsible for mapping new data points into the grid and, consequently, updating its characteristic vector [CT07]. This vector keeps information related to the grid such as the update time, grid density and grid status (normal or sporadic). In the offline phase, clusters are adjusted based on the density of the grids according to a certain threshold. Neighboring dense grids are merged to form clusters while the sparse grids are removed from the clusters. The dynamic evolution of the data stream is captured by applying a decay factor to the density of the grids. In [HGRC11], DGTSstream is proposed as an improvement of previous algorithms for better dealing with boundary points, by adopting a method based on similarity. Moreover, a grid-tree structure is used for storing the summary information and a density threshold is adopted based on the average density. DD-Stream [JTY08] is one of the extensions of D-Stream,

which improves the quality of the clusters by extracting boundary points in the grids. This process occurs in the offline phase and the border points are assigned by taking into account the distances from the center of the grids. With the goal of improving the performance of the offline component, [WND+09] proposes an algorithm termed as MR-Stream. This algorithm keeps a tree-like structure for the space partitioning, which allows clustering at multiple resolutions. During the offline phase the clusters are generated by determining the reachable cells at a user-defined distance. DCU-Stream [YLZY12] improves D-Stream by adapting the latter for the context of uncertain data streams, where the data is incomplete or imprecise. This is achieved by considering an uncertain tense weight for each data point that is mapped into the grid. This is assigned by considering the arrival time of the data and also its existence probability. Also in the context of uncertain data, [TCT13] proposes Clu-US. It uses the existence probabilities of the data tuples in the calculation of the distance between adjacent grids, instead of being calculated through the traditional geometric centers. PDG-OCUStream [HCRG11] is another density grid-based approach for clustering uncertain data streams. This algorithm is based on a sliding window model and uses a threshold for the probability density in order to control the cluster quality. In the context of uncertain data streams and motivated by the fact that existing algorithms are sensitive to the user-specified threshold, UG-Stream is proposed in [HZ14]. UG-Stream defines a dynamic threshold, which is computed together with the probability variance of the grid in order to distinguish between dense and sparse grids. With a similar goal as SDStream, DENGRIS-Stream [AW12] is proposed as an improvement of D-Stream for clustering over sliding windows. This approach discards grids whose timestamps are older than the beginning of the window. In [BKC13], an algorithm termed as ExCC is proposed for clustering heterogeneous data. As D-Stream, it also has an online and an offline phase. The numerical data is mapped to the grid, while for the categorical data, granularities are defined based on the unique values in the domain. Unlike D-Stream, a window model is not used, since the pruning is performed by taking the speed of the data stream into account.

In [TC09], an extension of D-Stream is proposed for clustering data streams taking into account the positional information of data in the grid. It uses the correlation between neighboring grids to merge them when this factor exceeds a certain threshold. Motivated by the fact that the sparsity of the grids is aggravated in the context of high-dimensional data, [RCH11] proposes PKS-Stream. This algorithm improves [TC09] and uses PKS-trees for keeping both the non-empty cells and their relations. The removal of the sparse grids occurs in the offline phase when the PKS-tree is adjusted. In [DCHR11], GDH-Stream is also proposed in the context of clustering high-dimensional data. It is based on subspace clustering, which means that the clustering algorithm is performed on a subset of the dimensions, therefore reducing the spatial complexity. The subspace is generated by ranking the dimensions according to their ability to separate projected clusters. This approach is improved by GDRH-Stream as proposed in [HMR12]. This algorithm considers the relative entropy of attributes in order to filter redundant features. They define a weighted attribute relativity measure, which is used to determine the subspace by computing it for the relevant attributes.

2.1.1.4 Model-based Clustering

Modelling techniques try to optimize the data fitness through probabilistic models. The parameters of the model are determined by ensuring a maximum fit of the underlying clusters. EM (Expectation Maximization) [DLR77] is a popular method to determine these parameters by clustering objects based on a membership probability. In the context of data streams, [DLN+09] proposes SWEM as an improvement of EM.

2.1.2 Consensus Clustering

Consensus clustering addresses the problem of reconciling multiple clusters of the same dataset without having access to the underlying features of the data. Different clusters can be obtained by varying selections of attributes or by multiple runs of the same non-deterministic clustering algorithm. The objective is to find an agreement with the multiple clusterings, which highlights their commonalities. A survey of ensemble techniques for clustering has been conducted in [GSIM09] and more recently in [SSVS14] in the context of mixed data clustering. This is an interesting approach for this problem, considering that each dimension can be clustered independently and then merged to reach a final consensus clustering. This process contemplates several approaches [NC07]: Pairwise Similarity, Graph-based, Mutual Information, Mixture Model and Cluster Correspondence.

Pairwise similarity measures similarity between data points based on their shared membership to the ensemble clusters. These measures are applied to a similarity-based algorithm in order to obtain the consensus clustering of the ensemble. Graph-based approaches adapt the ensemble of clusterings to a graph representation, while the consensus is obtained by the application onto a graph-based clustering algorithm. Mutual Information formulates an objective function to be maximized, which is based on the commonalities between the ensemble and the final consensus clustering. The Mixture Model approach is based on the generation of probabilistic models from a finite mixture of distributions. The final solution is obtained by solving the corresponding maximum likelihood problem. Cluster correspondence obtains the consensus clustering by optimization of a linear programming formulation combined with a voting procedure.

Several proposals have been made to adapt traditional consensus clustering approaches for massive datasets and streaming data.

[Eze13] scales an existing consensus clustering algorithm [TJ04], which relies on the Expected Maximization algorithm for mixture models. A strategy for distributing the EM algorithm is developed so as to be run on a cloud, which allows processing large amounts of data. This is done in the context of knowledge mining of large-scale medical data. In this work, multiple clusters are generated by projecting the data to random subspaces.

[Hor07] proposes scalable algorithms for merging cluster ensembles of large data sets and data streams. Different clusters are obtained by clustering disjoint subsets of the data. A global consensus is obtained by partitioning the clusters into consensus chains or groups and computing the weighted mean of the corresponding centroids of each one. The problem is approached in a

graph formulation in which the centroids of each partition are represented as vertices of a graph and the dissimilarity between them as weighted edges. The goal is to partition the r-partite graph, where r is the number of partitions to combine, into k target clusters by grouping similar centroids together. Since the partitioning is an NP-hard problem, two heuristics are used: Bipartite Merger and Metis Merger. The first finds the global consensus clustering by partitioning the ensembles into k equally sized clusters (consensus chains) with one to one centroid mapping between two partitions . The second partitions the ensembles into k centroid groups (consensus groups), but the group sizes are not guaranteed to be the same.

[YC11] proposes a weighted consensus clustering algorithm in a two-stage process. Different representations are extracted from the full temporal dataset, generating partitions which are clustered independently. A weighted consensus function is applied to reconcile these partitions to candidate consensus partitions. The weighting scheme is based on the evaluation of the clusterings of this phase according to three evaluation measures. The resulting candidate consensus clusters are further reconciled by an agreement function to yield a final consensus cluster.

In [ZZTG10] ensemble learning is applied for combining classifiers and clusters for mining data streams. The data stream is partitioned in chunks and a weighting scheme is applied on the ensemble according to the consistency between the base models and the up-to-date model. This is done in order to address the concept drifting problem.

[DAR09] addresses the problem of reconciling multiple clusters from different subspaces by means of a weighting scheme. The weighted clusters are obtained by a locally adaptive algorithm, which are further combined by a consensus function. Two functions are introduced for the weighted clustering ensembles, which approach the problem as a graph partitioning resolution: Weighted Similarity Partition Algorithm and Weighted Bipartite Algorithm. The first algorithm constructs similarities between data points based on the membership probabilities to each weighted cluster. The data points and their respective similarities are then mapped to a graph and a k-way partitioning, where k is the final number of clusters, is computed by minimizing the edge weight-cut. The second approach differs from the first in the sense that the problem is approached as a bipartite graph partitioning problem. In this case, the graph models both data points and clusters, which forms a bipartite graph, where the edges are weighted based on the cluster membership probabilities.

2.1.3 Clustering Evaluation

Clustering evaluation assesses the clustering analysis and the quality of the results generated by the process. This task includes the assessment of clustering tendency, the determination of the number of clusters and measurement of the clustering quality [HK06]. Clustering tendency verifies whether a nonrandom structure exists in the data. This is done to guarantee that the clustering analysis is meaningful for a dataset, since methods for pattern extraction may return misleading clusters. Assessing the clustering tendency can be achieved by using statistical tests for spatial

randomness as the Hopkins Statistic, given as:

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i} \tag{2.3}$$

$$x_i = min_{v \in D}\{dist(p_i, v)\} \tag{2.4}$$

$$y_i = min_{v \in D, v \neq q_i}\{dist(p_i, v)\} \tag{2.5}$$

with p_i data points uniformly sampled from the dataset. $dist(p_i, v)$ represents the distance between a data point and the neighbouring points. For a highly skewed dataset, the value of H is closer to 0.

Determining the number of clusters is a difficult task, since it depends on the shape and scale of the distribution and also on the granularity demanded by the user. An estimate can be obtained from several methods. A rule of thumb is to set the number of clusters to $\sqrt{n/2}$ for a dataset of n points. The elbow method takes into account the fact that the sum of within-cluster variance of each cluster is reduced with the increase of the number of clusters. Even though this increase allows for finer groupings, at some point the reduction on the variance is not significant and does not compensate the performance costs. The optimal number of clusters is, therefore, considered as the turning point. This point can be obtained by plotting the curve of the sum of within-cluster variance against the number of clusters. Another known method for estimating the optimal number of clusters is cross-validation. It divides the dataset into m parts, using m - 1 parts to build a clustering model and the remaining to assess the quality of the previously obtained model.

In order to measure the quality of the clustering several methods can be applied. These methods can be categorized as extrinsic, if a ground truth is available, or intrinsic otherwise. Extrinsic methods evaluate the resulting clusters with respect to the ground truth. Recent studies for this measures are found in [SZ08, WXC09]. Intrinsic methods measure the quality of the clusters by considering their separation. In [Mil81], thirty intrinsic measures are examined. Table 2.1 presents a list of evaluation measures according to their respective category, adapted from [KKJ+10, KKJ+11].

[Mil81] identifies a subset of the thirty internal measures examined by their correlation to the Rand statistic and Jaccard criterion. These six measures are Gamma, C Index, Point-Biserial, Tau, W/B statistics and G(+) index. [SZ08] examines seven external measures for clustering representations on data stream clustering: Purity, Cluster-based entropy, Class-based entropy, Homogeneity, Completeness, V-measure and Variation of Information. CMM (cluster mapping measure) is another evaluation measure, proposed in [KKJ+11] for the context of evolving data streams. Sixteen external measures are studied in [SZ08] for K-means clustering. This number is then narrowed down to thirteen by discarding some equivalent measures. These include the Purity, F-Measure, Mutual Information, Variation of Information, Rand statistic, Jaccard coefficient, Fowlkes and Mallows Index, Hubert's statistics, Minkowski score, classification error and van Dongen crite-

Table 2.1: Clustering Evaluation Measures

Internal Measures	**External Measures**
Gamma	Rand statistic
C Index	Jaccard coefficient
Point-Biserial	Folkes and Mallow Index
Log Likelihood	Hubert Γ statistics
Dunn's Index	Minkowski score
Tau	Purity
Tau A	van Dongen criterion
Tau C	V-measure
Somer's Gamma	Completeness
Ratio of Repetition	Homogeneity
Sum squared distances (SSQ)	Variation of Information
Adjusted Ratio of Clustering	Mutual Information
Fagan's Index	Class-based entropy
Deviation Index	Cluster-based entropy
Z-Score Index	Precision
D Index	Recall
Silhouette Coefficient	F-measure
W/B	Kappa
G(+)	Classification Error

rion. In the context of density-based stream clustering the most common evaluation measures are [AWS14]: SSQ, Purity and Rand Index.

2.2 Distance Measures

Clustering requires the computation of the similarities between objects and so different distance functions are used for this task, depending on the nature of the dimensions. The dissimilarity between two d-dimensional objects x_A and x_B is defined as $dist(x_A,x_B)$.

2.2.1 Numerical Distance

[HK06] refers the following as the most common for numeric data: Euclidean distance, Manhattan distance and the Minkowski distance. The Supremum (or Chebyshev) and the Mahalanobis distances are also mentioned. The Euclidean distance is measured as a straight line and the formula is given as:

$$dist(x_A,x_B) = \sqrt{(x_{A_1} - x_{B_1})^2 + \cdots + (x_{A_d} - x_{B_d})^2} \tag{2.6}$$

If an importance is given to each dimension or attribute as a weight w, the Weighted Euclidean distance can be formulated as:

$$dist(x_A,x_B) = \sqrt{w_1(x_{A_1} - x_{B_1})^2 + \cdots + w_d(x_{A_d} - x_{B_d})^2} \tag{2.7}$$

The Manhattan distance is measured in blocks by summing the vertical and horizontal distances independently. It is defined as:

$$dist(x_A, x_B) = |x_{A_1} - x_{B_1}| + \cdots + |x_{A_d} - x_{B_d}| \tag{2.8}$$

The Minkowski distance generalizes both the Euclidean and the Manhattan distance and it is defined as:

$$dist(x_A, x_B) = \sqrt[h]{|x_{A_1} - x_{B_1}|^h + \cdots + |x_{A_d} - x_{B_d}|^h}, h \geq 1 \tag{2.9}$$

When $h = 1$, the formula corresponds to Manhattan's and when $h = 2$, it corresponds to Euclidean's.

As for the Mahalanobis distance, it takes into account the correlations of the data and includes a covariance matrix S^{-1} being defined as:

$$dist(\overline{x_A}, \overline{x_B}) = \sqrt{(\overline{x_A} - \overline{x_B})S^{-1}(\overline{x_A} - \overline{x_B})^T} \tag{2.10}$$

The supremum distance generalizes the Minkowski distance for $h = \infty$, giving the maximum difference in values between both objects. It is defined as:

$$dist(x_A, x_B) = \lim_{h \to \infty} (\sum_{j=1}^{d} |x_{A_j} - x_{B_j}|^d)^{\frac{1}{d}} = max_j |x_{A_j} - x_{B_j}| \tag{2.11}$$

For the specific case of geographical coordinates, the Haversine formula [MK10] can be used. It measures the great-circle distance between points and so the Earth's shape is considered as a perfect sphere. The formula is given as:

$$dist(x_{A_{Sp}}, x_{B_{Sp}}) = 2R \sin^{-1}\left(\left[\sin^2\left(\frac{x_{A_{lat}} - x_{B_{lat}}}{2}\right) + \cos x_{A_{lat}} \cos x_{B_{lat}} \sin^2\left(\frac{x_{A_{lng}} - x_{B_{lng}}}{2}\right)\right]^{0.5}\right) \tag{2.12}$$

where R is the radius of the Earth and $x_{A_{lat}}, x_{A_{lng}}, x_{B_{lat}}, x_{B_{lng}}$ are the geographical coordinates (latitude,longitude) of both points respectively. The resulting distance is in the same unit as R.

2.2.2 Textual Distance

For measuring the similarity between documents or textual data, [HK06] defines the cosine similarity and the Tanimono distance, which is a variation of the former. In order to compare two documents, they must first be represented as term-frequency vectors. The vector may correspond to the absolute frequency or it may be weighted as $TFIDF$ [MRS08].

The idea of $TFIDF$ is to overcome the fact that the absolute frequency considers all terms as equally important. This improvement is achieved by a weighting technique, which reduces the relevance of common terms. For a tweet t, let α be its textual content and α_i a term in that content.

$TFIDF$ defines the weight of a term as:

$$TFIDF = TF(\alpha_i) \cdot IDF(\alpha_i) \tag{2.13}$$

where $TF(\alpha_i)$ is the term-frequency of the term and IDF the relevance of the term. IDF is given as:

$$IDF(\alpha_i) = \log(\frac{N}{df(\alpha_i)}) \tag{2.14}$$

where N is the size of the text collection and $df(\alpha_i)$ the frequency of the term in the documents. The IDF measure is high for rare terms and low for frequent terms.

[SHK10] proposes an hybrid $TFIDF$ in the context of microblogging summarization. This approach aims to overcome the sensitivity of $TFIDF$ formula to the document length, which poses a problem when generating summaries from multiple documents. Considering a sentence S with n words, the weight assigned to it is given as:

$$W(S) = \frac{\sum_{i=0}^{n} TFIDF(\alpha_i)}{nf(S)} \tag{2.15}$$

where nf is a normalization factor given by the equation:

$$nf(S) = max(minimumThreshold, n) \tag{2.16}$$

Another weighting scheme named $TFPDF$ is proposed in [BI02] with the goal of extracting hot terms which are discussed most often in channels. In this approach, an higher weight is given to a term when its frequency within a channel is also high. Besides, it grows exponentially with the increase of the ratio between the number of documents containing the term and the total number of documents. The formula of $TFPDF$ is given as follows

$$TFPDF = \sum_{c=1}^{C} |F_c(\alpha_i)| \exp\left(\frac{N_c}{df_c(\alpha_i)}\right) \tag{2.17}$$

$$|F_c(\alpha_i)| = \frac{F_c(\alpha_i)}{\sqrt{\sum_{k=1}^{K} F_c(\alpha_k)^2}} \tag{2.18}$$

where C is the number of channels, K the total number of terms in a channel, F_c the frequency of a term in channel c, N_c the number of documents in channel c and df_c the frequency of the term in the documents.

The formula for measuring the similarity using the cosine measure is given as:

$$dist(x_{A_C}, x_{B_C}) = \frac{\beta_A \cdot \beta_B}{\|\beta_A\| \|\beta_B\|} \tag{2.19}$$

where β_A and β_B are two term-frequency vectors. $\|\beta_A\|$ and $\|\beta_B\|$ correspond to the Euclidean norm of the aforementioned vectors. The resulting value varies between 0 and 1. The first is

obtained when both vectors are orthogonal and do not match. A higher value means a greater match factor between the vectors. The Tanimono distance is a variant of this measure for the case of binary-valued attributes. In the referred scenario, the cosine similarity can be interpreted in terms of shared attributes. Its formula is given as:

$$dist(x_{A_C}, x_{B_C}) = \frac{\beta_A \cdot \beta_B}{\beta_A \cdot \beta_A + \beta_B \cdot \beta_B - \beta_A \cdot \beta_B} \tag{2.20}$$

In [RLW12], a variant of Jaccard's similarity with Dice's coefficient is used to compute similarity between documents:

$$dist(x_{A_C}, x_{B_C}) = \frac{|\beta_A \cap \beta_B|}{min(|\beta_A|, |\beta_B|)} \tag{2.21}$$

[RKT11] proposes a variation of Cosine similarity and also of Jaccard similarity in the context of short text clustering. The equations of the variations are as follows, respectively:

$$dist(x_{A_C}, x_{B_C}) = 1 - \frac{\sum_{d=1}^{D} \beta_A^d \cdot \beta_B^d}{\|\beta_A\| \cdot \|\beta_B\|} \tag{2.22}$$

$$dist(x_{A_C}, x_{B_C}) = 1 - \frac{|\beta_A \cap \beta_B|}{|\beta_A \cup \beta_B|} \tag{2.23}$$

2.2.3 Social Distance

A social graph can be inferred from the relationship between users in Twitter. If we consider the users as vertices and the relationships as edges, the social distance is obtained from the distance between the vertices of the graph, which are mapped to tweets' authors. [HK06] defines two distance measures for graphs: Geodesic Distance and SimRank.

Geodesic distance is a simple measure defined as the number of edges which compose the shortest path between the vertices. A shortest path algorithm must be applied, such as Dijkstra's [Dij59]. SimRank is a similarity measure based on random walk and structural context. It considers two vertices as being similar if they have similar neighbours. The concept of *individual in-neighbourhood* of a vertex is introduced, as given by the equation 2.24.

$$I(v) = \{u | (u,v) \in E\} \tag{2.24}$$

for a directed graph $G = (V,E)$, where V is the set of vertices and E the set of edges, such that $E \subseteq VxV$. For two distinct vertices $u, v \in V$, the SimRank distance is given as:

$$dist(u,v) = \begin{cases} 0 & I(u) = 0 \vee I(v) = 0 \\ \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(y)} s(x,y) & I(u) \neq 0 \wedge I(v) \neq 0 \end{cases} \tag{2.25}$$

where C is a constant between 0 and 1. The result is also between 0 and 1.

In [ACF11], a social distance function named Network Similarity is introduced, which is based on the mutual friends graph and the friendship graph. The former graph, $MFG(u,v)$ contains the

mutual friends of the users and their relationships, while the former, $FG(u)$, contains all friends of a user and their relationships. The Network Similarity function is given as:

$$dist(u,v) = \frac{\log(|MFG(u,v)|}{\log(2|FG(u)|)} \tag{2.26}$$

where $|G|$ denotes the number of edges of a graph G.

[Dek06] introduces a social distance function based on link strength, which is weighted based on the periodicity of the communications. The values are assigned from a discrete scale, which varies between 0 (less than once per month) to 1 (communication every day).

[SSB05] compares the performance of six network similarity measures in the context of recommender systems for social networks: $L1Norm$, Cosine similarity, Pointwise Mutual Information (positive correlations), Pointwise Mutual Information (positive and negative correlations), $TFIDF$ and $LogOdds$. These measures are also used for the work in [ACF13], where a user similarity measure is proposed for online social networks by combining both network and profile similarity. Considering two sets of users A and B, the $L1Norm$ is given as:

$$dist(A,B) = \frac{|A \cap B|}{|A| \cdot |B|} \tag{2.27}$$

This measure evaluates to the overlap between the two groups of users, divided by the product of their sizes. It penalizes larger sets more severely than Cosine similarity.

The Pointwise Mutual Information (positive correlations) is given as:

$$dist(A,B) = \frac{|A \cap B|}{|U|} \log\left(\frac{|A \cap B| \cdot |U|}{|A| \cdot |B|}\right) \tag{2.28}$$

where U represents the whole set of users. The Pointwise Mutual Information (positive and negative correlations) is given as:

$$dist(A,B) = \frac{|A \cap B|}{|U|} \log\left(\frac{|A \cap B| \cdot |U|}{|A| \cdot |B|}\right) + \frac{|\overline{A} \cap \overline{B}|}{|U|} \log\left(\frac{|\overline{A} \cap \overline{B}| \cdot |U|}{|\overline{A}| \cdot |\overline{B}|}\right) \tag{2.29}$$

The Pointwise Mutual Information focuses on the correlations between the memberships on each set.

Finally, $LogOdds$ is given as:

$$dist(A,B) = \log\left(\frac{|A \cap B|}{|A \cap \overline{B}|}\right) \tag{2.30}$$

This measure evaluates how membership in one set predicts the membership or absence in another.

2.3 Twitter

This section will discuss Twitter and also research that has been done on this subject.

2.3.1 Description

Twitter [Twi14b] is a microblogging service that allows users to publish short text messages, known as "tweets", with at most 140 characters. It is a social network in the sense that the messages are broadcast to the each author's "followers". Therefore, a relationship is defined by the "follower" or "following" relationship, being that each user is allowed to choose who to follow.

Table 2.2 presents the important concepts associated with a tweet, so as to promote a better understanding of the social interactions.

Table 2.2: Twitter concepts

Concept	Description
Retweet (RT)	Share another user's tweet
Mention (@ + username)	Identify a user in a tweet
Reply (@ + username)	Answer to a previous user's tweet
Hashtag (# + topic name)	Association of a keyword to a tweet
Localization	User's geo-coordinates when sending the tweet

2.3.2 SocialBus

SocialBus [BOM+12], formerly known as TwitterEcho, is a research platform which supports the collection and processing of messages from social networks. It currently supports data extraction from Facebook[1] and Twitter, but it is designed to be easily extensible. The current architecture of SocialBus is presented in Figure 2.4.

The Twitter Consumer retrieves tweets from Twitter using the Twitter Streaming API [Twi14a]. The tweets are sent to a message broker for translation of the data format. The server processes the tweets, extracts metadata, while also being responsible for indexing and tokenization. The processed messages are stored in MongoDB [2]. After persisting the information, it is subjected to batch processing for mining different kinds of knowledge.

2.3.3 Research using Twitter

The interest on Twitter for research purposes has been growing in the last few years. This section will focus on investigations that cluster Twitter data for several purposes, such as topic summarization, event detection and sentiment analysis.

Despite being a rich source of information, the massive amount of tweets makes it difficult for users to plow through them for contents of interest. Twitter topic summarization attempts to solve this issue by summarizing tweets while representing them as short text pieces which cover the most relevant topics. [YZF12] proposes a framework for topic summarization in Twitter which summarizes topics by sub-topics. For this process, a clustering algorithm is used together with a graph-based ranking algorithm which takes into account both the social influence and the

[1] https://www.facebook.com
[2] http://www.mongodb.org/

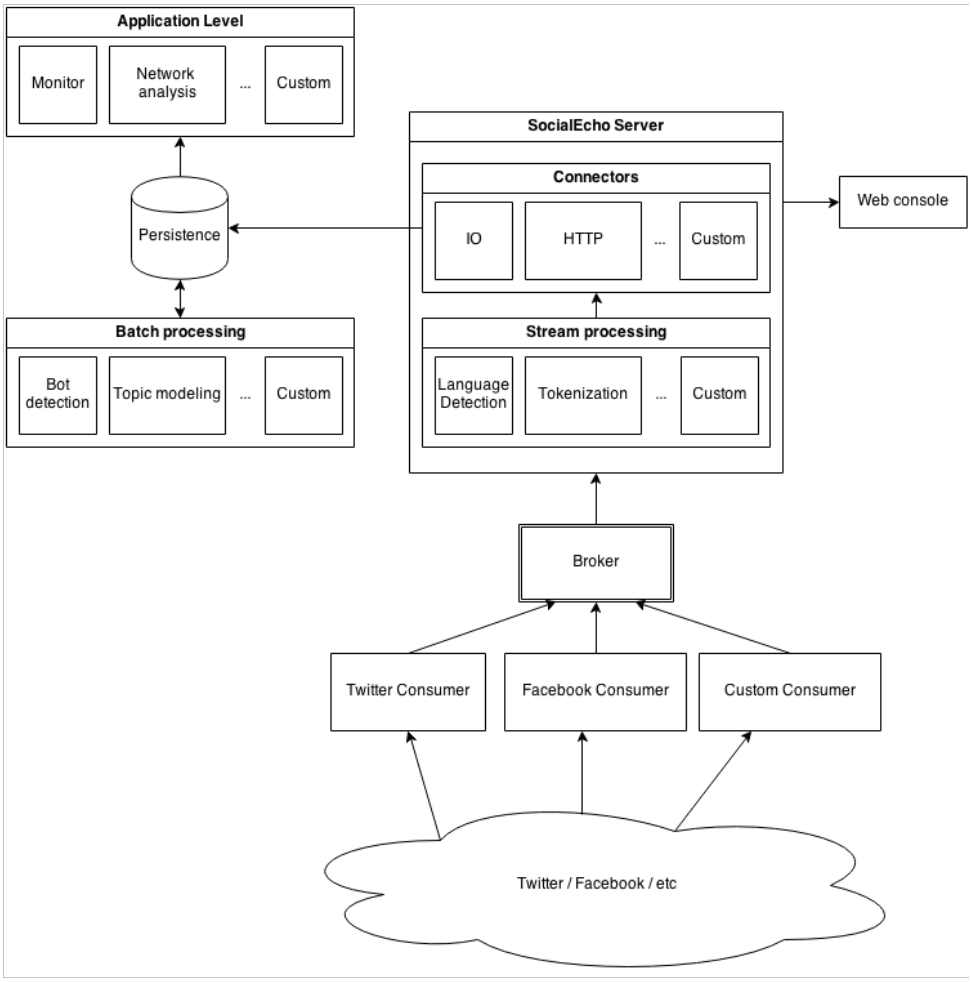


Figure 2.4: SocialBus system architecture [REA15]

content quality of tweets. The content quality is measured by the readability and content richness, while the social influence takes other metrics into account, which include the number of followers, messages and lists, the follower/following ratio and the number of mentions and retweets received by the tweet's author. [SWCC13] proposes a prototype named *Sumblr* for topic summarization over tweet streams. It uses an incremental clustering algorithm and a summarization technique that generates both online and historical summaries. Embedded in the process is also a ranking system, which takes into consideration the temporal, content and social dimensions. For the social dimension an UserRank value is calculated for the tweet's author as in [CLOW11].

[CLOW11] proposes an adaptive indexing scheme for Twitter data so as to allow real-time search on tweets, motivated by the high update and query loads inherent to the microblogging system. It has a ranking system that considers the temporal, content and social dimensions by composing the user's PageRank [PBMW99], the popularity of the topics, the term frequency and the timestamp of the tweet.

Twitter practices have also been studied with focus on retweets. RetweetPatterns [Rod14]

developed a platform for studying the information spreading on Twitter by extracting patterns from the retweets. This is achieved with the aid of GetMove, a tool for social media analysis through moving object pattern mining, in combination with TweeProfiles. [BGL10] examines retweeting as a conversational practice by studying the reasons and styles of what has become a convention in the microblogging service. [Bru12] analyzes hashtag and reply networks in the temporal dimension by extracting data from the Twitter API. It allows the generation of network visualisations, which cover the different phases of the discussions and the formation of clusters based on the extraction of patterns from the interaction of the participants.

Sentiment analysis is another topic of interest on the microblogging platform. The goal of this task is to classify tweets based on the feelings they convey. These can be simply positive or negative or more granular by adding other categories such as neutral and objective. The work in [BF10] describes an approach for opinion mining and sentiment analysis on Twitter data streams using a sliding window model. In this case, the classification target is binary, considering only positive or negative feelings. An extension of MOA [BHKP10] is proposed in [BHP11] for real-time tweet mining with adaptation to changes in the stream. It allows classifying tweets in real-time, which is useful for sentiment analysis. It accesses the Twitter Streaming API and preprocesses the data by extracting content features using an incremental $TFIDF$ weighting scheme. An opinion mining framework for Twitter is proposed in [KBQ14]. It aims to overcome the problems related to the sparsity of the data and identification of sarcasm. The framework processes data from the Twitter Streaming API and uses a classification algorithm with a hybrid scheme.

Unlike sentiment analysis, data analysis in event detection is not confined to tweets mentioning only certain keywords, since the events are not known *a priori*. [Cor12] presents a mechanism for event detection by analyzing hashtags from the Twitter Streaming API. This is achieved by using wavelet signal analysis and the topics are inferred by a Latent Dirichlet Allocation [BNJ12] model. [Lee12] also approaches event detection by performing multidimensional clustering on Twitter data. It detects real-time event topics by extracting spatio-temporal features from the microblogging platform using a density-based online clustering method. The results are presented as a spatial distribution of topics in real-time. In [BNG11], a tool is proposed for real-world event identification by using multidimensional clustering and classification. The former step of the process clusters Twitter data in the temporal, social and content dimensions. The classification step is applied to the resulting clusters, which labels them depending on whether or not they are considered actual events. Event detection is also the research topic of EventRadar [BL12]. It detects local events from Twitter streaming data by clustering and classification. First, the proposed scheme tries to find clusters on Twitter data which contain the same subset of words. This is achieved by applying a density-based clustering algorithm. It then classifies the clusters as potential events or not by taking into account seven day historic data. A visual analytics approach for social media is presented in [CTB+12]. It allows interactive data analysis including exploration of abnormal topics and events. The major topics are extracted from Twitter messages and ranked using Latent Dirichlet Allocation, while the abnormality within topics is identified by seasonal trend decomposition.

Olhó-Passarinho [Mot14] is an extension of TweeProfiles for spatio-temporal analysis of the images contained in tweets. It replaced the tweet's textual content representation of the extended platform with feature vectors of the images. The web application allows the visualization of the clusterings on the three dimensions: temporal, spatial and content. In [EOSX10], a method is presented for identifying lexical variation from raw text based on topic and geographical region. The model is constructed from data collected from the Twitter Streaming API. [RLW12] proposes a tool for discovering and displaying underlying memes in social media. The memes are formed from clusters of common phrases, which appear in multiple documents. These phrases are ranked prior to the clustering process in order to identify the most informative to display to the user. [Li14] proposes methods for mining the online social network data. These methods cover the topics of textual summarization, event detection, storyline generation and classification. The influence of the users is also studied and three important dimensions are identified in this context: Monomorphism vs Polymorphism, High Latency vs Low Latency and Information Inventor vs Information Spreader. For this purpose, a dynamic influence model is presented to calculate the current influence of the users, as well as predict their future influence.

2.4 TweeProfiles2

TweeProfiles2 [Per14] is an extension of TweeProfiles [Cun13] and is a data-mining tool that allows clustering Twitter data streams on real-time, taking into account multiple dimensions. Moreover, it enables the visualization of the results of the referred data mining task in each of the three considered dimensions: spatial, temporal and content. For the analysis of the spatial dimension, the tweet's geographical coordinates are used. Considering that not all the retrieved tweets are geo-located, this aspect poses itself as in import restriction. The temporal dimension is analysed by means of the tweet's timestamp. Since a temporal decay is already applied by the clustering algorithm, this dimension is being implicitly considered, so by this perspective it was decided to cluster on the hour and weekday of the tweet instead. For the content dimension, the tweet's text is used. The fact that the length of a tweet is capped at 140 characters has the drawback of causing the text similarities to be rather small. The developed clustering algorithm, HybridDenStream, was adapted from DenStream to allow the process to take into account not only the numerical dimensions, but all the aforementioned.

2.4.1 System Architecture

Figure 2.5 presents the architecture of the proposed solution for TweeProfiles2. The data stream is pipelined to a back-end server, where the data is preprocessed and then fed to the microclustering algorithm. The resulting microclustering is stored in a MySQL [MyS14] database, which is accessed by the macroclustering algorithm whenever a clustering request arrives. The resulting clusters are then passed to the visualization module, which is responsible for displaying them to allow the analysis of the results [Per14].

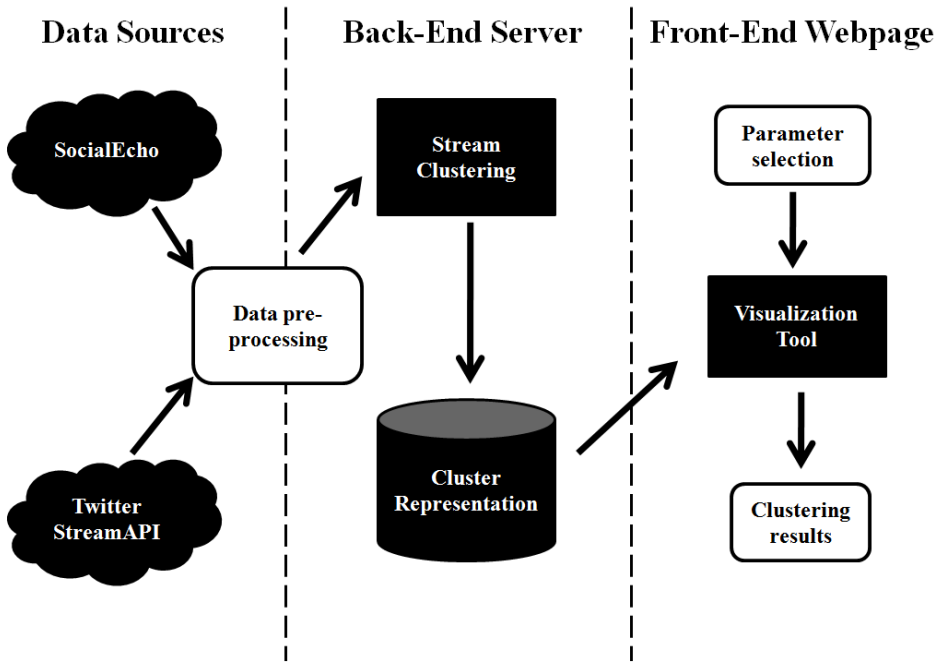


Figure 2.5: TweeProfiles2 system architecture [Per14]

2.4.2 Operation

The HybridDenStream algorithm, which was adapted from DenStream, applies a piece-wise distance function to calculate both the tweet-tweet and tweet-cluster distances. The formulas used for each tweet attribute type is given in Table 2.3 [Per14]. A min-max normalization is applied to the distance values so as to attenuate the differences in the order of magnitude among the several dimensions.

Table 2.3: Distance function by attribute type [Per14]

Tweet Attributes	**Formula**
latitude,longitude	Haversine Formula
hour,weekday	Euclidean Formula
text	Cosine Similarity

The concept of Hybrid MicroCluster is also introduced, which is an improvement over DenStream's microcluster. It enables the summarization of all the information concerning tweets that it needs for the clustering process.

A hybrid-micro-cluster, at time t is defined as $HMC(w, ww, c, r)$ for a group of similar tweets (as defined in equation 2.31) $TW_1 \dots TW_n$ with time stamps $T_1 \dots T_n$, and text vectors $W_1 \dots W_m$ [Per14]:

$$TW(lat, lng, hou, wkd, date, \overline{txt}) \tag{2.31}$$

$$w = \sum_{j=1}^{n} f(t - T_j), w > \mu \text{ is the weight;} \tag{2.32}$$

$$c = \begin{cases} \frac{\sum_{j=1}^{5} f(t-T_i) TW_{i_j}}{w} \\ \frac{\sum_{k=1}^{m} f(t-T_i) W_k}{ww} \end{cases} \text{ is the center;} \tag{2.33}$$

$$r = \frac{\sum_{j=1}^{n} f(t - T_j) dist(X_j, c)}{w}, r \leq \varepsilon \text{ is the radius;} \tag{2.34}$$

where $f(t)$ is the temporal decay function (as defined in equation 2.2 and $dist(X_{i_j}, c)$ denotes the composed distance function between the point X_i and the center c.

The clustering algorithm comprises the following steps [Per14]:

1. When a new point X arrives, the nearest potential-micro-cluster, a, is identified and an attempt is made to add the new point to it;

2. If a (with the new point added) violates the radius constraint ($r < \varepsilon$), X is removed from it, the nearest outlier-micro-cluster, b, is identified and X is added to it;

3. If b respects the radius constraint, the weight of b is checked to see if it is enough to transform it into a PMC ($w > \beta * \mu$). If it is, b is removed from the the OMC buffer and it is added to the PMC buffer;

4. Finally, if b violates the radius constraint, a new OMC, c, is created with only a point X and then it is added to the OMC buffer.

The macroclustering step applies DBSCAN, which accesses the microclusters stored in the database, in order to produce the final clustering results.

2.5 TweeProfiles3

TweeProfiles has been further extended in TweeProfiles3 [Mai15]. It aimed to create a visualization tool for TweeProfiles2 capable of displaying tweet profiles based on multiple dimensions and also to improve the integration of TweeProfiles2 with SocialBus. This integration facilitated gathering real-time data from Twitter with the possibility of imposing restrictions on the retrieved tweets. The visual methods integrated have been supported both in theoretical principles as well as practical by surveying professionals. These visualization improvements also include the implementation of a search module which assists users with textual analysis.

Figure 2.6 presents the interface which allows the analysis of the results over the spatial, temporal and content dimensions.

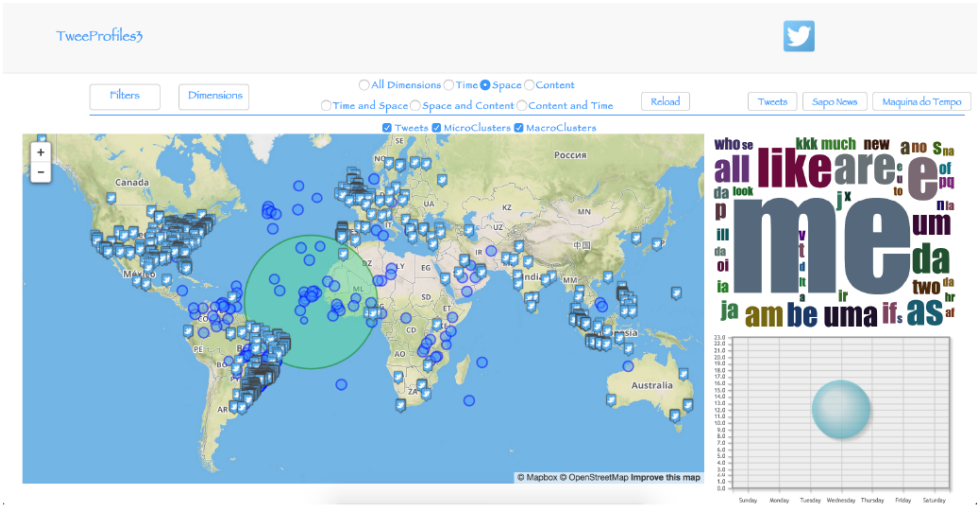


Figure 2.6: TweeProfiles3 visualization interface [Mai15]

2.5.1 System Architecture

Figure 2.7 presents the architecture of TweeProfiles3. The data collection is performed by SocialBus, which handles the extraction and pre-processing methods. This data is saved in MongoDB and serves as input for the microclustering algorithm. The resulting microclusters are passed to the macroclustering algorithm, which stores the final clustering in MySQL. The visualization module then retrieves the clusters from the database and presents them in a user-friendly interface, which allows the user to analyze the results in the considered dimensions.

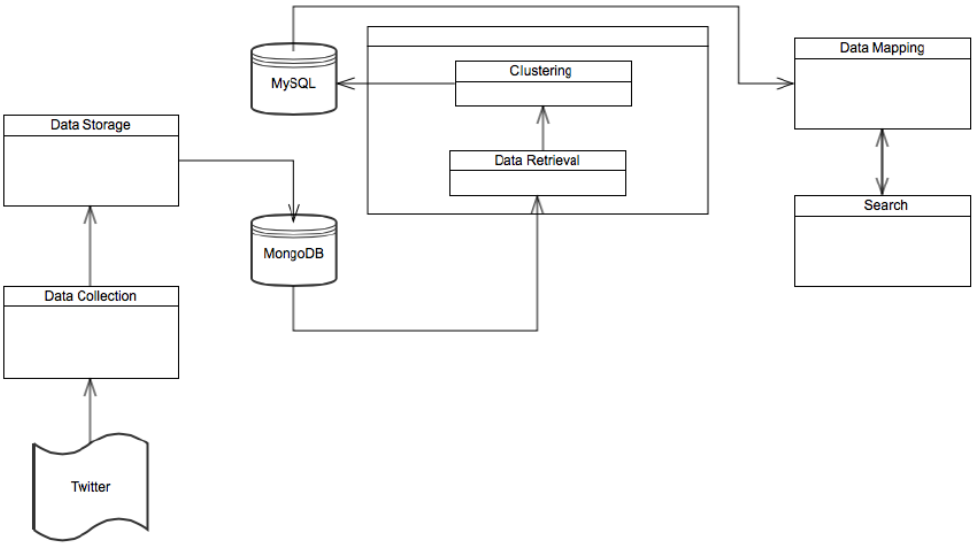


Figure 2.7: TweeProfiles3 system architecture [Mai15]

Chapter 3

TweeProfiles4

This chapter describes the composition and the operations of the developed tool, TweeProfiles4. First, a description of the architecture is provided. Then, the data process and the clustering tasks are explained, followed by the description of the evaluation process. Finally, an overview of the visualization interface is given.

3.1 Introduction

TweeProfiles4 was developed as an extension of TweeProfiles [Mai15], which not only allows clustering data streams in real-time, but also the application of a dynamic weighting scheme regarding each dimension. Furthermore, the previous version of the tool has been extended with the inclusion of a clustering evaluation process.

The clustering is performed over three dimensions: spatial, temporal and content. The spatial dimension takes into account the geographical coordinates of the tweet. The temporal dimension, besides taking into account the hour and weekday of the tweet, also has an implicit influence on the whole clustering process, since a temporal decay is applied. The content dimension refers to the tweet's text, which is limited to 140 characters.

In order to allow the user to dynamically changes his preferences during the clustering process, a consensus clustering approach has been applied, which is explained in greater detail in this chapter. The evaluation process also involved the adaptation of the clustering algorithms.

3.2 System Architecture

TweeProfiles4 is integrated with SocialBus, whose architecture is depicted in Figure 2.4. SocialBus collects the tweets from the Twitter Streaming API, processes them and persists them in MongoDB, which serves as the data source for the clustering algorithm.

The high-level architecture of the whole platform is represented in Figure 3.1.

The online clustering algorithm is responsible for incrementally processing the data by applying several pre-processing operations (see Section 3.3), mining the data and persisting the results in a relational database.

These results are used by both the visualization module and the offline clustering algorithm. The former allows the analysis of the results and is described in detail in Section 3.7. The latter is triggered by user request, which includes his weighting preferences for each dimension. This final operation yields the final clustering results, which are also made accessible for the visualization module through persistent storage in a relational database. The clustering evaluation is also performed during the offline phase, after the computation of the final clusterings.

Both the online and offline clustering algorithms are described in detail in Section 3.5.

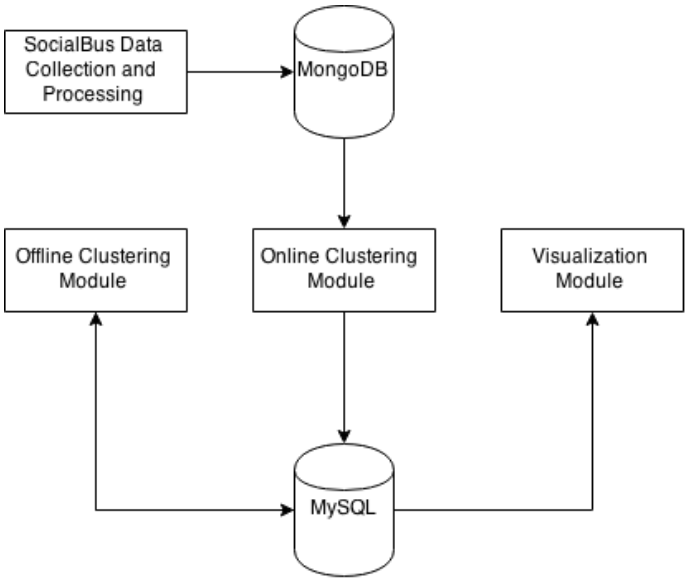


Figure 3.1: TweeProfiles4 system architecture

3.3 Data Processing

The clustering algorithm receives a tweet data stream as input. Each tweet collected from the Twitter API is in the JSON format and contains over 30 fields, some of which contain nested objects. Not all the fields are relevant for the dimensions being studied, so the data is filtered and only the fields presented in Listing 3.1 are kept.

The next pre-processing step involves extracting the hour of the day (0-23) and the day of the week (1-7, mapping to Sunday-Monday) from the date field. The final step is responsible for processing the text of the tweet as enumerated in the following subtasks:

1. Detect the language of the text;
2. Remove any URLs contained in the text;

```
{
    "tweetid":359419233295269888,
    "username":"PipaOliveira_"
    "text":"Uns dormem e outros estudam...",
    "date":"2013-07-22T21:06:37.000Z",
    "lat":"-71.3553287",
    "lon":"-40.15760962"
}
```

Listing 3.1: Example of a tweet after filtering irrelevant fields

3. Remove all the punctuation from the text;

4. Tokenize the text;

3.4 Distance Functions

The distance functions applied to the clustering algorithm were those which had been selected for the previous version of the platform. These are summarized in Table 3.1 (adapted from [Per14]), which also includes the value used for the min-max normalization.

Table 3.1: Distance functions and normalization per dimension

Formula	**Dimension**	**Tweet fields**	**Maximum**
Haversine	spatial	latitude, longitude	20.020 (km)
Euclidean	temporal	hour, weekday	$\sqrt{565} = 23,77$
Cosine Similarity	content	text	1

The normalization of the distance values is required since the order of magnitude varies depending on the formula applied. These similarity measures are used in the computation of both tweet-tweet and tweet-cluster distances.

Defining the distance as a piecewise function allows one to perform a weighted combination of the similarities of each dimension. Considering a dimension D and $dist_D(x_A, x_B)$ as the distance between two entities x_A and x_B, which may be tweets or centroids, and w_D as the relative weight, the weighted distance measure is given as:

$$dist(x_A, x_B) = w_C \cdot dist_C(x_A, x_B) + w_T \cdot dist_T(x_A, x_B) + w_S \cdot dist_S(x_A, x_B) \tag{3.1}$$

with

$$w_C + w_T + w_S = 1 \tag{3.2}$$

The aforementioned weighting scheme is applied both in the macroclustering process performed by DBSCAN and in the evaluation of the final clusterings.

3.5 Clustering

We want to allow the user to dynamically change his preferences regarding the importance of each dimension during the clustering process. Clustering streaming data requires it to be constantly summarized, so it is not possible to obtain meaningful results by starting the clustering process with a set of parameters and then change them at later stages. In order to address this problem, the online clustering process must be agnostic to the weights of each dimension. Moreover, it must provide the necessary representations to be used when a final clustering request arrives, which includes the user's preferences.

Since no assumption can be made on the weights until a final clustering result is demanded, the developed solution relies on the construction of unidimensional microclusters during the online phase. Then, when a clustering request arrives with user-defined weights, weighted multidimensional clusters are constructed from the microclusters computed in the previous phase. Afterwards, the offline phase gives the final clustering from these multidimensional microclusters using DBSCAN. Table 3.2 summarizes the clustering process of each phase, both in the proposed solution and in TweeProfiles2 and TweeProfiles3.

Table 3.2: Clustering process

	Unidimensional Clusters	**Multidimensional Clusters**
Online Phase (Micro)	TweeProfiles4	TweeProfiles2/3
Offline Phase (Macro)		TweeProfiles2/3 TweeProfiles4

Unlike static data clustering, where the data points can be kept in memory, in the streaming paradigm data is summarized and incrementally updated. So, it is not possible to reconstruct multidimensional clusters from unidimensional clusters with full accuracy, since data is inevitably lost. Therefore, the goal is to find a summarization and reconstruction process which gives the most approximate clustering to the one that would be obtained if the weights were being taken into account from the beginning of the online phase.

We will explain the clustering mechanism in greater detail in Section 3.5.1, and the two variants developed for the construction of the muldimensional microclusters in Sections 3.5.2 and 3.5.3.

3.5.1 Clustering Mechanism

The developed algorithm is divided in two phases: online and offline. The online phase is not only responsible for the incremental maintenance of microclusters, but also of a graph which represents the overlap between them. The offline phase is responsible for providing the final clusterings by taking into account the weighting of each dimension defined by the user, the previously obtained microclusters and their relation in terms of degree of overlapping. A diagram of the offline phase is presented in Figure 3.3.

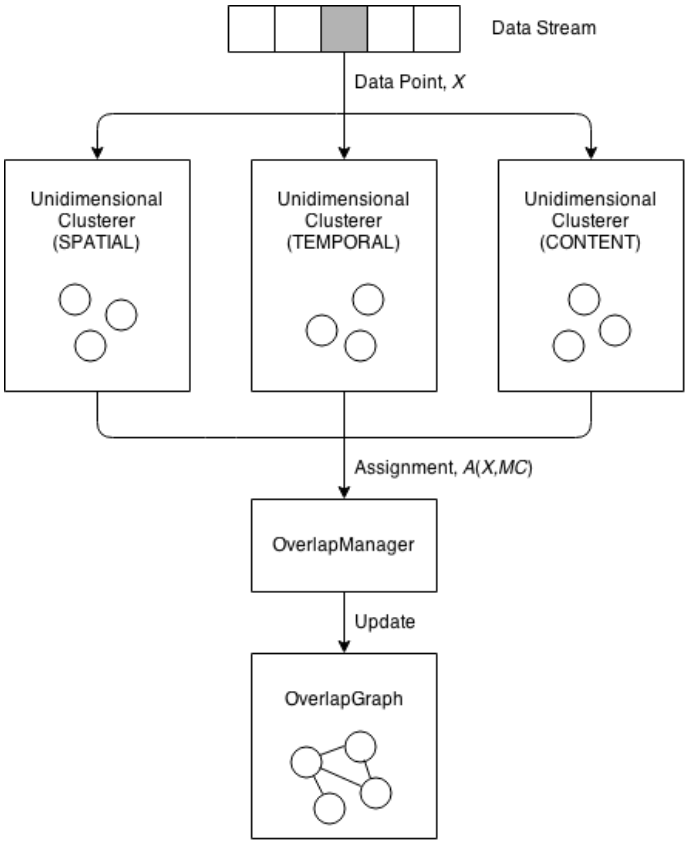


Figure 3.2: Online phase diagram

For the online phase, we instantiate one unidimensional clusterer per dimension being studied. In the context of Twitter data, this means we have a spatial clusterer, a temporal clusterer and a content clusterer. Each of these clusterers corresponds to an independent instance of a microclustering algorithm extended from HybridDenStream (see Section 2.4.2). We consider them unidimensional, since each clusterer only extracts patterns taking into account their respective dimension. The similarity measures applied for each dimension are described in Section 3.4. The pseudo-code of our extension of HybridDenStream for the microclustering algorithm is presented in Algorithm 3. The extension of HybridDenStream also includes changes to the HybridMicroClusters, which had to be modified to keep a sample of the merged tweets, which is used later by the clustering evaluation process (see Section 3.6).

We consider a point-to-cluster assignment $A(X, MC)$, where X is a d-dimensional data point and MC a microcluster, as the relation between a data point and the cluster on which the former was merged by the microclustering algorithm. During the online phase, every time a data point arrives from the stream, it is passed to each one of the clusterers, which will merge it into some microcluster. After merging, each clusterer communicates the corresponding assignment, A, to an *OverlapManager* agent. This agent is responsible for collecting the assignments of each clusterer and updating the *OverlapGraph* accordingly. The aforementioned graph is a undirected

graph, where each vertex, V, is a microcluster and every edge $E(V_1, V_2)$, which connects two vertices $V1$ and $V2$, is weighted according to the overlap between the connected microclusters. The edge weight measures the number of data points in common that each microcluster pair has merged. For every assignment pair $A(X_1, MC_A), A(X_2, MC_B)$, where $X_1 = X_2$, collected by the *OverlapManager*, the edge weight $E_w(MC_A, MC_B)$ is incremented by 1 or set to 1 if no such edge exists on the *OverlapGraph*. The communication between each unidimensional clusterer and the *OverlapManager* is asynchronous and it was implemented with a producer-consumer pattern for performance reasons.

The offline phase is triggered on demand by a request which includes the weighting preference of the user for each dimension. When a clustering request arrives, the microclusters obtained from each clusterer are collected and another graph is derived from the *OverlapGraph*. The new graph is obtained by removing the vertices of the *OverlapGraph* which correspond to outlier-micro-clusters and the edge weights are recalculated as follows:

$$E_w(MC_A, MC_B) = \frac{2 * E_w(MC_A, MC_B)}{MC_{A_{size}} + MC_{B_{size}}} \tag{3.3}$$

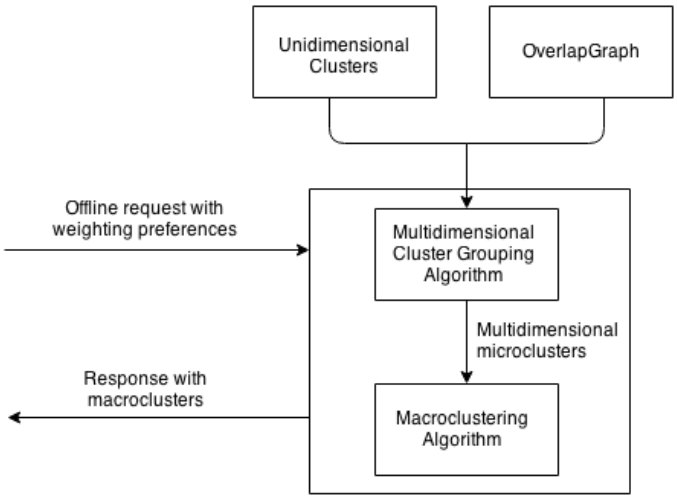


Figure 3.3: Offline phase diagram

The outlier-micro-clusters are only filtered from the *OverlapGraph* at this stage, since these may change to potential-micro-clusters over time and maintaining this consistency during the online phase would introduce a communication overhead, which would degrade the performance.

At this point, the unidimensional microclusters and the final *OverlapGraph* are fed to another algorithm which is responsible for providing the multidimensional microclusters. Two algorithms were developed for this task, to which we refer as Solution A (see Section 3.5.2) and Solution B (see Section 3.5.3).

Algorithm 3 HybridDenStream extension

procedure HYBRIDDENSTREAM EXTENSION(D,ε,β,μ,λ)
  $T_p = \frac{1}{\lambda}\log(\frac{\beta\mu}{\beta\mu-1})$
  Get the next point X at current time t from data stream D;
  Try to merge X into its nearest p-micro-cluster c_p;
  if $r_p \leq \varepsilon$ **then**
    Merge X into c_p;
    Send assignment $A(X, c_p)$ to OverlapManager agent;
  else
    Try to merge X into its nearest o-micro-cluster c_o;
    if $r_o \leq \varepsilon$ **then**
      Merge X into c_o;
      Send assignment $A(X, c_o)$ to OverlapManager agent;
      if $w_o > \beta\mu$ **then**
        Remove c_o from outlier-buffer and create a new p-micro-cluster c_{p_n} by c_o;
      end if
    else
      Create a new o-micro-cluster c_{o_n} by X and insert into the outlier-buffer;
      Send assignment $A(X, c_{o_n})$ to OverlapManager agent;
    end if
  end if
  if (t mod T_p) $= 0$ **then**
    for each p-micro-cluster c_p **do**
      if $w_p < \beta\mu$ **then**
        Delete c_p;
      end if
    end for
    for each o-micro-cluster c_o **do**
      $\xi = \frac{2^{-\lambda(t-t_o+T_p)}-1}{2^{-\lambda T_p}-1}$
      if $w_o < \xi$ **then**
        Delete c_o;
      end if
    end for
  end if
  if clustering request arrives **then**
    Generate clusters;
  end if
end procedure

After the aforementioned algorithm produces the multidimensional microclusters, a final clustering step is applied to generate the macroclusters using DBSCAN. This macroclustering step has been adapted to use a weighted combination of the distance functions as explained in Section 3.4.

3.5.2 Solution A

In this solution, the *OverlapGraph* is applied a k-way partition algorithm [KK98], similar to the approach used in [Hor07], using the METIS [Kar15] package. This outputs arbitrarily-sized partitions, which represent groups of clusters with strong overlapping with each other. The clusters of each partition are then merged to form multidimensional clusters (one per partition). If multiple clusters of the same dimension are present on the same partition, their respective centroid is averaged. The execution of the partitioning algorithm requires the number of partitions to be fixed, which we set to the maximum number of unidimensional microclusters per dimension.

3.5.3 Solution B

The first solution has the drawback that the number of partitions has to be defined *a priori*. Solution B relaxes this constraint and instead tries to group strong overlapping unidimensional microclusters without the referred parameter definition. For this we cluster the *OverlapGraph* using DBSCAN, where the microclusters are the vertices and the distances between each pair is defined by the weight of the edge which connects them. As in Solution A, the multidimensional clusters are formed by merging the clusters of each partition (one per partition), which the algorithm yields. Again, their respective centroid is averaged if the same partition happens to include multiple clusters of the same dimension.

3.6 Evaluation

For the evaluation of the extracted patterns, internal measures were selected and implemented. This decision is supported by the fact that there is no ground truth for the distribution of the Twitter data, therefore external measures are not applicable. The clustering evaluation measures implemented are represented in Table 3.3, according to the objective function which provides the optimal value.

Table 3.3: Evaluation measures and their optimal index value

Maximize	**Minimize**
Silhouette Coefficient	SSQ
Dunn's Index	G(+)
Tau	Davies-Bouldin Index
Gamma	W/B
Point-Biserial	C Index

In the following, we denote
n = number of points,
k = number of clusters,
n_k = number of points in cluster k,
x_i = ith point in cluster C_k,

C_k = set of points belonging to cluster k,
N_t = number of distinct pairs of points:

$$N_t = \frac{n(n-1)}{2}, \tag{3.4}$$

N_w = number of distinct pairs of points belonging to the same cluster:

$$N_w = \sum_{i=1}^{k} \frac{n_i(n_i-1)}{2}, \tag{3.5}$$

N_b = number of distinct pairs of points belonging to the different clusters:

$$N_b = N_t - N_w, \tag{3.6}$$

S_w = sum of the intra-cluster distances:

$$S_w = \sum_{c=1}^{k} \sum_{\substack{i,j \in C_c \\ i<j}} dist(x_i, x_j), \tag{3.7}$$

S_b = sum of the inter-cluster distances:

$$S_b = \sum_{c=1}^{k-1} \sum_{d=c+1}^{k} \sum_{\substack{i \in C_c \\ j \in C_d}} dist(x_i, x_j) \tag{3.8}$$

C Index [HL76] is calculated using equation 3.9:

$$CIndex = \frac{S_w - S_{min}}{S_{max} - S_{min}}, S_{min} \neq S_{max} \tag{3.9}$$

where S_{min} is the sum of the N_w smallest distances between all point pairs and S_{max} is the sum of the N_w largest distances between all point pairs.

Gamma [BH75] is calculated using equation 3.10:

$$Gamma = \frac{s^+ - s^-}{s^+ + s^-} \tag{3.10}$$

where s^+ is the number of times an inter-cluster distance is strictly greater than an intra-cluster distance. s^- is the number of times an inter-cluster distance is strictly less than an intra-cluster distance. The equality cases are not taken into account.

Using the same notation as Gamma, G(+) [Mil81] is calculated using equation 3.11:

$$G(+) = \frac{2s^-}{N_t(N_t-1)} \tag{3.11}$$

Using the same notation as Gamma, Tau [Mil81] is computed using equation 3.12:

$$Tau = \frac{s^+ - s^-}{\sqrt{(N_t(N_t-1)/2-t)(N_t(N_t-1)/2)}} \tag{3.12}$$

where t is the number of comparisons of two point pairs where both represent either intra-cluster comparisons or inter-cluster comparisons.

Silhouette Coefficient [Rou87] is computed using equation 3.13:

$$SilhouetteCoefficient = \frac{\sum_{i=1}^{n} \frac{b(i)-a(i)}{max\{a(i);b(i)\}}}{n} \tag{3.13}$$

where $a(i)$ is the average dissimilarity of the ith point to the others belonging to the same cluster. $b(i)$ is the minimum average dissimilarity of the ith point to the others in different clusters.

Dunn's Index [Dun74] is given in equation 3.14:

$$Dunn = \frac{\min\limits_{1\leq i<j\leq k} dist(C_i,C_j)}{\max\limits_{1\leq c\leq k} \max\limits_{x,y\in C_c} dist(x,y)} \tag{3.14}$$

SSQ [HK06] is computed using equation 3.15:

$$SSQ = \sum_{i=1}^{n}\sum_{c=1}^{k} dist(x_i,C_c) * dist(x_i,C_c) \tag{3.15}$$

Davies-Bouldin Index [DB79] is calculated according to equation 3.16:

$$DB = \frac{1}{k}\sum_{c=1}^{k} \max_{c\neq d}\left(\frac{S_{w_c}+S_{w_d}}{dist(C_c,C_d)}\right) \tag{3.16}$$

where S_{w_c} corresponds to the sum of intra-cluster distances for cluster C_c as defined in equation 3.17:

$$S_{w_c} = \sum_{\substack{i,j\in C_c \\ i<j}} dist(x_i,x_j) \tag{3.17}$$

Point-Biserial [Mil81] is computed using equation 3.18:

$$PB = \frac{(\frac{S_b}{N_b} - \frac{S_w}{N_w})\sqrt{\frac{N_w N_b}{N_t^2}}}{s_d} \tag{3.18}$$

where s_d is the standard deviation of all distances.

W/B [Mil81] is calculated using equation 3.19

$$\underline{W}/\underline{B} = \frac{S_w N_b}{N_w S_b} \tag{3.19}$$

In order to compute the inter-cluster and intra-cluster distances, as required by the evaluation measures, it was necessary to keep a sample of the tweets merged by each microcluster. Regarding the macroclusters, their sample is obtained by merging those of the corresponding microclusters. A sample is kept instead of every tweet, given the memory constraints inherent to the data stream clustering task. For performance reasons, we fixed the maximum size of the sample to 100 tweets. Since there are no dependencies between the computations of each measure, they are all processed in parallel, which significantly speeds up the overall evaluation task.

The distance functions are calculated as described in Section 3.4, which includes the weighting scheme to relativize each dimension.

3.7 Visualization

In the following subsections we describe the relevant improvements and the development that has been conducted in the context of the visualization module of TweeProfiles.

3.7.1 Clustering

The visualization tool has been improved to allow the user to dynamically change the relative weight of each dimension. As presented in Figure 3.4, the user may change his preferences either by dragging the corresponding slider or by setting the weight manually. After the modifications, an offline clustering request is performed and the visualization tool is refreshed automatically with the latest results. This part of the interface replaces the previous version, which only allowed predefined (binary) combinations of the dimensions to be chosen.

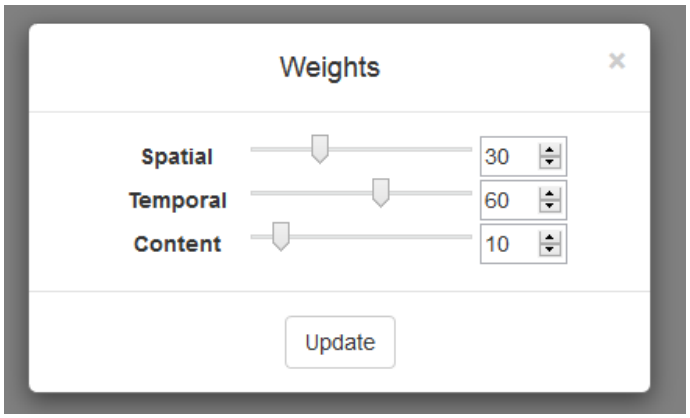


Figure 3.4: Dimension weighting preferences interface

3.7.2 Evaluation

An interface was developed to allow the visualization of the evaluation results regarding the final clusterings. The results are presented in a time series chart (similar to MOA clustering performance interface), which allows dynamically toggling between microcluster evaluation and macro-clustering evaluation. Moreover, it allows the user to dynamically select the evaluation measures to be displayed. The chart is updated at frequent and regular intervals from the database, which is populated by the evaluation module.

An example of the interface with evaluation results from a sample clustering is presented in Figures 3.5 and 3.6. For the visualization of the chart, the Javascript libraries D3.js[1] and C3.js[2] were used.

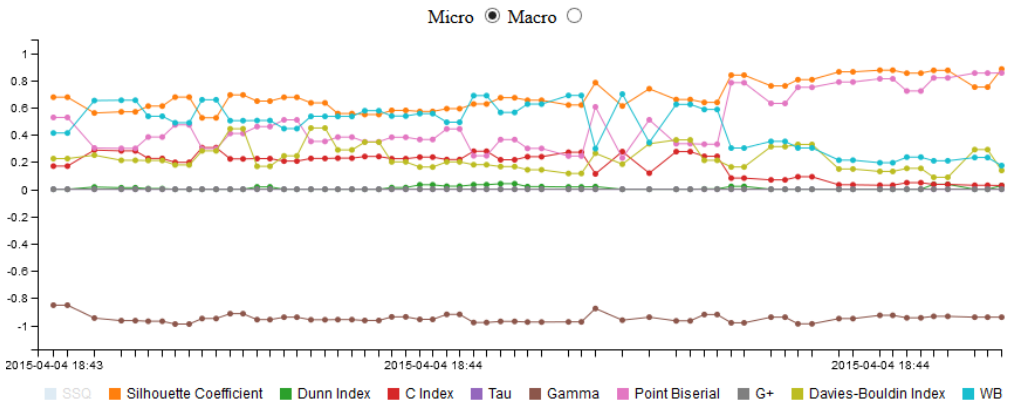


Figure 3.5: Clustering evaluation visualization

[1] http://d3js.org/
[2] http://c3js.org/

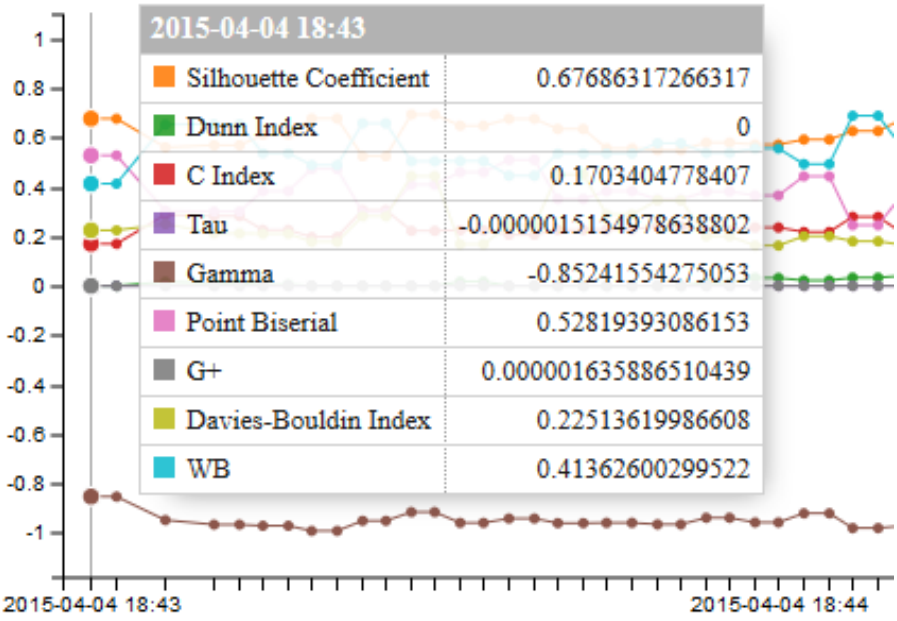


Figure 3.6: Clustering evaluation visualization with detailed information

Chapter 4

Results

In this chapter we present the test dataset, the experimental setup and we discuss the results obtained.

4.1 Exploratory Analysis

In order to test the proposed algorithms, we used a dataset with 112739 geo-located tweets from SocialBus, from June 2012 to February 2013. This is a subset of the dataset that has been used in TweeProfiles [Cun13].

Figure 4.1 shows the distribution of the tweet frequency over the time span of the dataset.

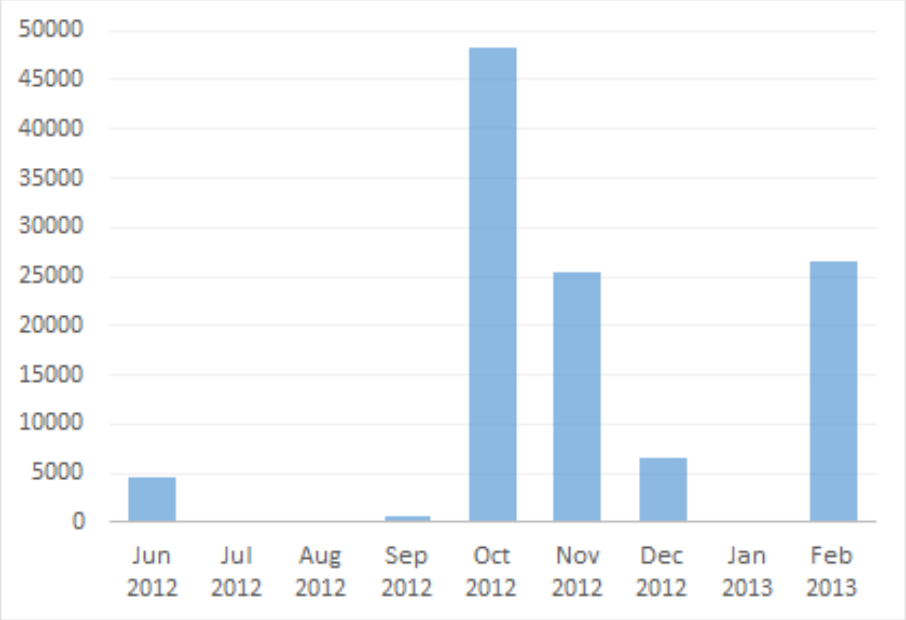


Figure 4.1: Test dataset monthly distribution

The reasons for the observed disparity are related to the filtering applied to only include data from Portuguese users or tweets which were written in Portuguese. Besides, the throughput of the data stream is dependent on the frequency of the posts and not limited by the system. There were also limitations regarding system failures and maintenance during some of the periods.

The geographical distribution of the tweets present in the dataset can be observed in Figure 4.2, with Portugal, Spain and Brazil being the most representative countries.

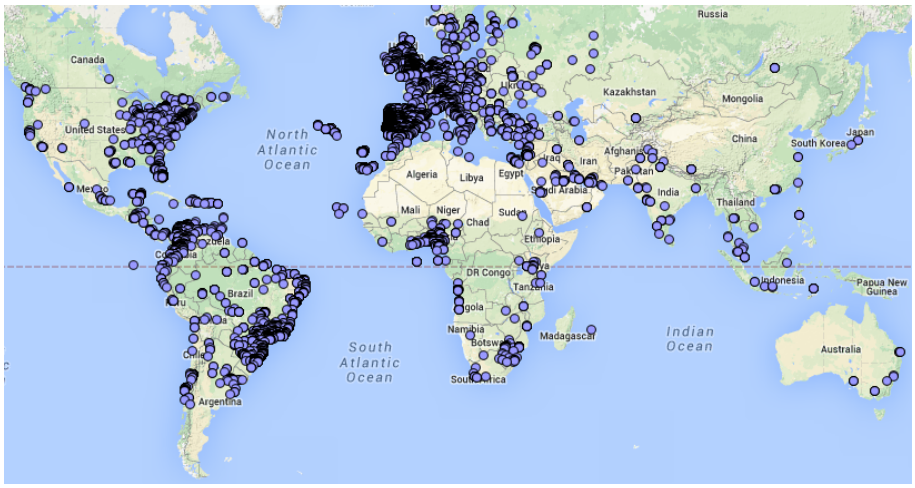


Figure 4.2: Test dataset spatial distribution

Figure 4.3 presents the distribution of the test dataset according to the day of the week. It reveals Friday and Saturday as the most active days in terms of tweet posting, with Tuesday being the least active.

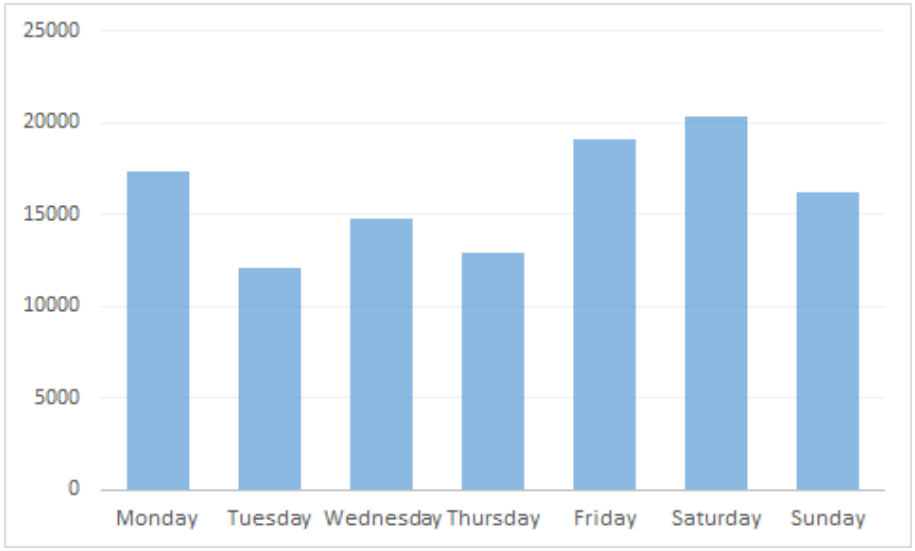


Figure 4.3: Test dataset weekday distribution

Figure 4.3 presents the distribution of the test dataset according to the hour of posting of each tweet. There is less activity until noon, unlike the night hours where the frequency of posts achieves its peak.

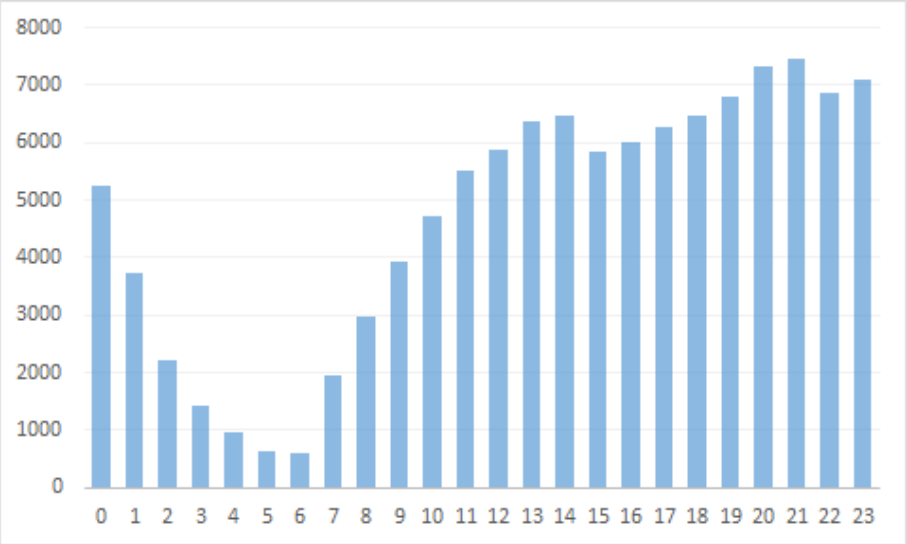


Figure 4.4: Test dataset hourly distribution

We generated a data stream from the static dataset using Streamalizer [Per14], a simple tool to ease the testing of the stream clustering algorithm. It also allows the selection of custom time periods for the tweets' retrieval.

4.2 Experimental Setup

In order to assess the proposed algorithms, a series of tests has been conducted with different conditions. The main goal was to compare our solution with the one proposed and implemented in TweeProfiles2 [Per14].

First, we ran TweeProfiles2's algorithm and both our solutions with a set of defined parameters, as shown in Table 4.1 (adapted from [Per14]), while varying the ε parameters for both the microclustering algorithms and DBSCAN (table 4.2) and the weights of each dimension (table 4.3). For each execution, we gathered the resulting microcluster count and macrocluster count.

After this process, we picked a set of three executions, one per solution, and compared their results from the perspective of each dimension being studied.

Moreover, we have assessed the quality of the results with the application of the implemented evaluation measures.

We have also compared the computational performance of our approach against the solution applied for TweeProfiles2.

Table 4.1: Clustering algorithms fixed parameters

Name	Abbrev	Description	Min Value	Max Value	Defined Value
MinPoints	mp	Defines the minimum number of points to create a microcluster/macrocluster	1	∞	2
InitPoints	ip	Number of points for initialization	0	∞	50
μ	μ	Used in the p-micro-cluster/o-micro-cluster restriction	1	∞	1
Beta	β	Used in the p-micro-cluster/o-micro-cluster restriction	0	1	0.2
Lambda	λ	Used in the time decay function; affects the decay rate of the stream	0	1	0.25
Processing speed	s	Defines the number of data points per time unit	1	∞	100

Table 4.2: Clustering algorithms variable parameters

Name	Abbrev	Phase	Description	Min Value	Max Value	Step Value
Epsilon	eps	Micro	Defines the minimum radius of a microcluster	0.2	1	0.1
Epsilon	eps	Macro	Defines the minimum radius of a ε-neighbourhood	0.2	1	0.1

Table 4.3: Weighting parameters

Execution	Spatial	Temporal	Content
1	0.33	0.33	0.33
2	0.5	0.5	0
3	0.5	0	0.5
4	0	0.5	0.5

4.3 Results

4.3.1 Clustering

In this section we present the results of the performed executions as described in Section 4.2.

4.3.1.1 All Dimensions

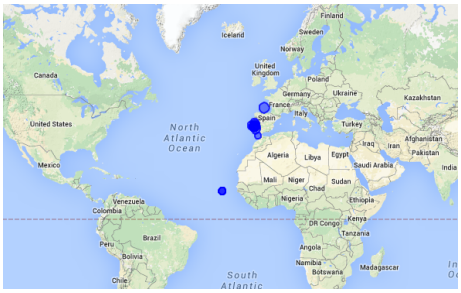
The following results were obtained using the weight parameters of Table 4.3 for Execution 1, which takes into account all dimensions.

Table 4.4 presents the microcluster and macrocluster count for the original solution and both developed solutions.

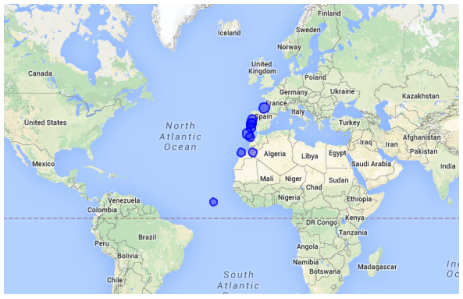
Table 4.4: Clustering results for Execution 1

Solution	Micro	Macro	Macro Outliers
Original	265	17	239
Solution A	125	10	52
Solution B	208	9	126

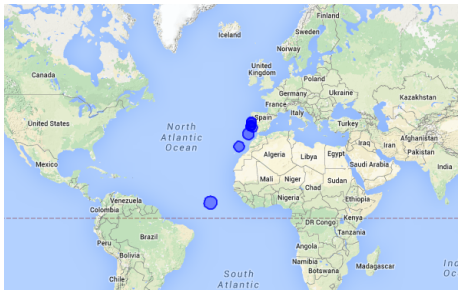
Regarding the spatial dimension, we can observe the macroclusters obtained in Figures 4.5a, 4.5b and 4.5c.



(a) Original



(b) Solution A



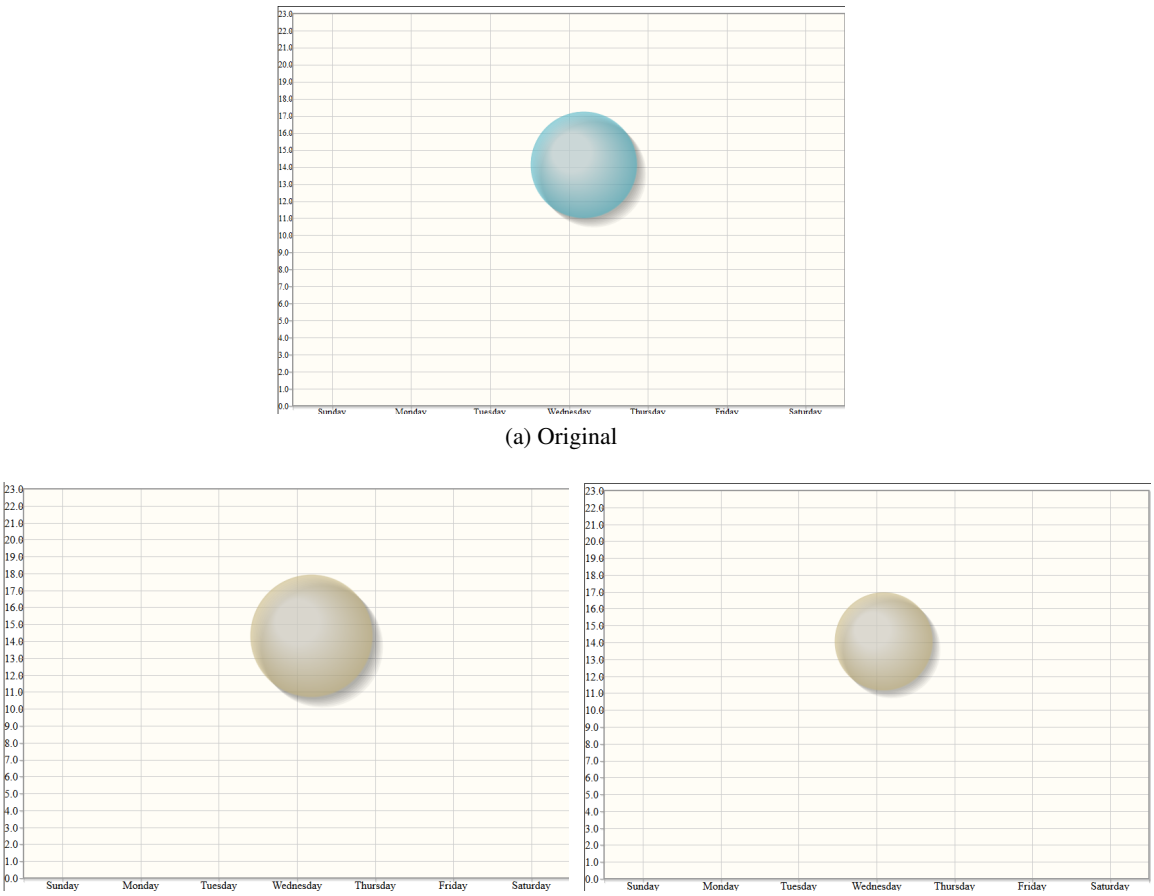
(c) Solution B

Figure 4.5: Spatial view of the macroclusters for Execution 1

The temporal dimension is represented in Figures 4.6a, 4.6b and 4.6c.

The content dimension is represented in Figures 4.7a, 4.7b and 4.7c.

The dissimilarity in terms of the number of microclusters is noticeable, particularly between the original solution and solution A. However the difference is reduced for the macroclusterings. From a spatial perspective, we observe similar results with the exception of some macroclusters positioned closer to Africa in the developed solutions when compared to the original. Regarding the remaining dimensions, we find the results to be very similar.



(a) Original

(b) Solution A

(c) Solution B

Figure 4.6: Temporal view of the macroclusters for Execution 1

(a) Original

(b) Solution A

(c) Solution B

Figure 4.7: Content view of the macroclusters for Execution 1

4.3.1.2 Spatial and Temporal Dimensions

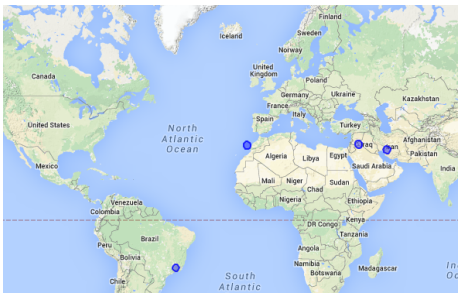
This subsection presents the results which were obtained using the weight parameters of Table 4.3 for Execution 2, which comprises the spatial and temporal dimensions.

Table 4.5 presents the microcluster and macrocluster count for the original solution and both developed solutions.

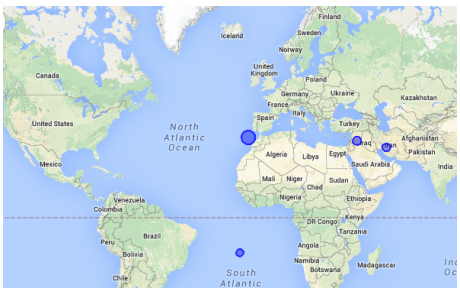
Table 4.5: Clustering results for Execution 2

Solution	Micro	Macro	Macro Outliers
Original	164	4	4
Solution A	139	4	46
Solution B	167	4	0

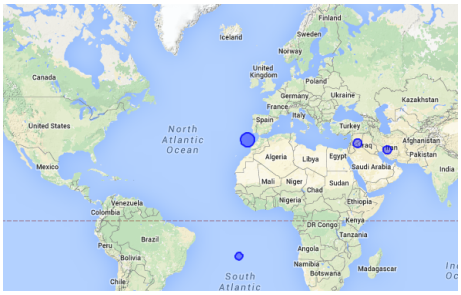
Regarding the spatial dimension, we can observe the macroclusters obtained in Figures 4.8a, 4.8b and 4.8c.



(a) Original



(b) Solution A



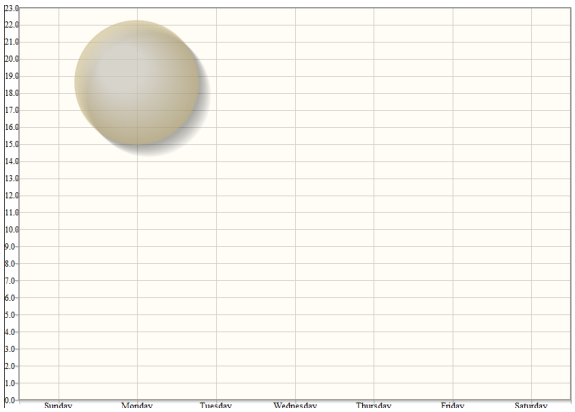
(c) Solution B

Figure 4.8: Spatial view of the macroclusters for Execution 2

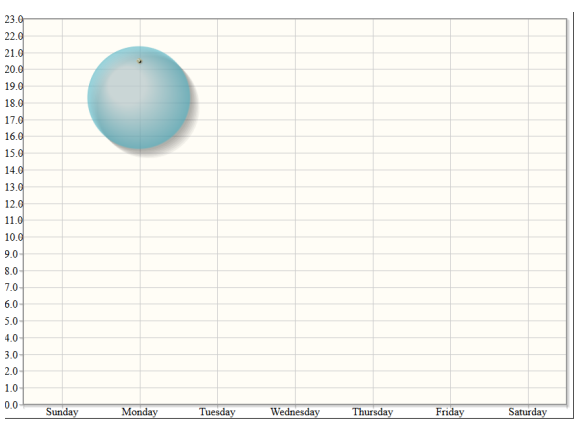
The temporal dimension is represented in Figures 4.9a, 4.9b and 4.9c.

Finally, the content dimension is represented in Figures 4.10a, 4.10b and 4.10c.

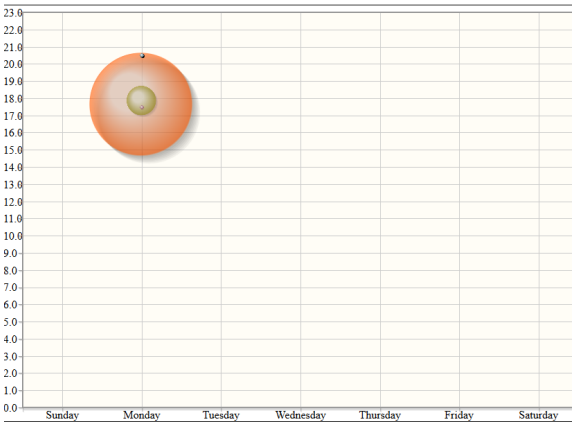
Despite the variations in the number of microclusters of the solutions, the resulting macroclusters are the same in number and very similar in every perspective. The similarities between the developed solutions are the most noticeable. We can observe some differences between these solutions and the original, namely on the spatial view (Figure 4.8a), regarding the macrocluster further south, which is marked above Brazil on the original solution as opposed to a positioning



(a) Original



(b) Solution A



(c) Solution B

Figure 4.9: Temporal view of the macroclusters for Execution 2

more to the east on the others (Figures 4.8b and 4.8c). This macrocluster also affects the content dimension, causing the differences observed between Figure 4.10a and Figures 4.10b and 4.10c. However, the variations in the aforementioned dimension are not very relevant, since it is not being weighted in this execution.

(a) Original

(b) Solution A

(c) Solution B

Figure 4.10: Content view of the macroclusters for Execution 2

4.3.1.3 Spatial and Content Dimensions

In this subsection we present the results which were obtained using the weight parameters of Table 4.3 for Execution 3, which comprises the spatial and content dimensions.

Table 4.6 presents the microcluster and macrocluster count for the original solution and both developed solutions.

Table 4.6: Clustering results for Execution 3

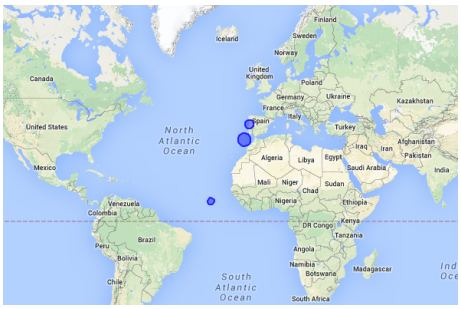
Solution	Micro	Macro	Macro Outliers
Original	106	3	27
Solution A	110	3	14
Solution B	139	2	20

The macroclustering results from the spatial perspective can be observed in Figures 4.11a, 4.11b and 4.11c.

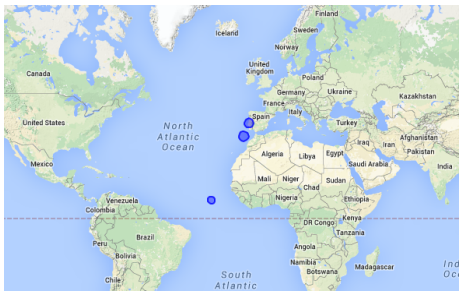
The temporal dimension is represented in Figures 4.12a, 4.12b and 4.12c.

The results from the content perspective are represented in Figures 4.13a, 4.13b and 4.13c.

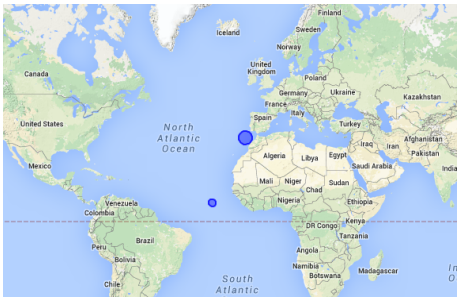
Even though the number of microclusters varies for the solutions, the macroclusterings are very similar, specially between the original solution and solution A. For these solutions, we have



(a) Original



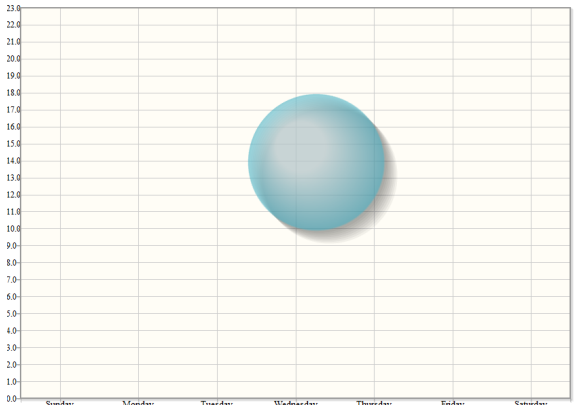
(b) Solution A



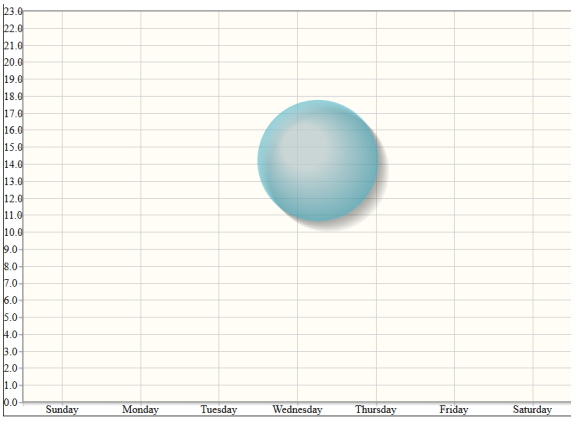
(c) Solution B

Figure 4.11: Spatial view of the macroclusters for Execution 3

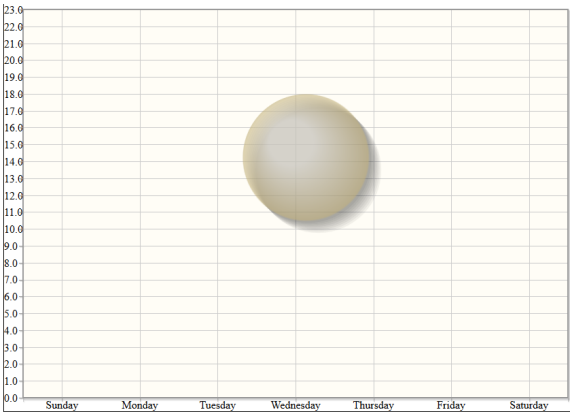
3 macroclusters as a result of this execution and one less for solution B, which is the macrocluster further north (Figures 4.11a, 4.11b and 4.11c). From the temporal perspective, we have the same results on the three solutions. As for the content dimension, we do not observe any significant differences either, even though solution B considers fewer macroclusters.



(a) Original



(b) Solution A



(c) Solution B

Figure 4.12: Temporal view of the macroclusters for Execution 3

(a) Original

(b) Solution A

(c) Solution B

Figure 4.13: Content view of the macroclusters for Execution 3

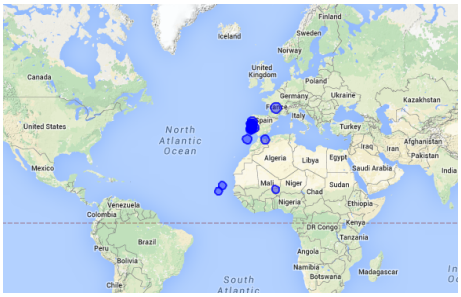
4.3.1.4 Temporal and Content Dimensions

The following results were obtained using the weight parameters of Table 4.3 for Execution 1, which takes into account the temporal and content dimensions. Table 4.7 presents the microcluster and macrocluster count for the original solution and both developed solutions.

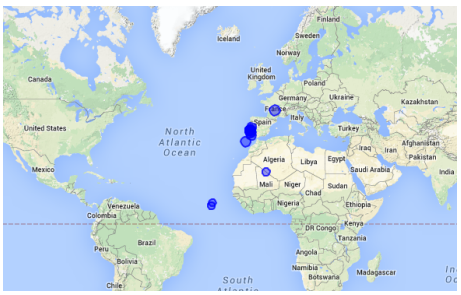
Table 4.7: Clustering results for Execution 4

Solution	Micro	Macro	Macro Outliers
Original	164	19	26
Solution A	153	16	0
Solution B	180	15	0

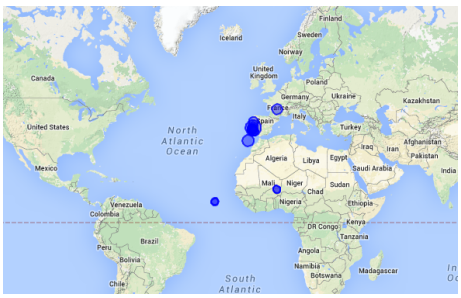
The macroclustering results from the spatial perspective can be observed in Figures 4.14a, 4.14b and 4.14c.



(a) Original



(b) Solution A



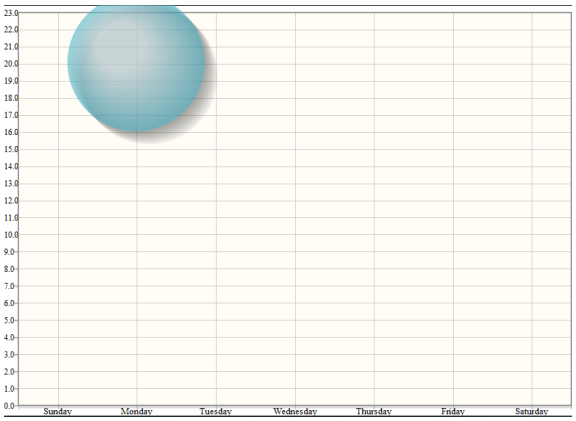
(c) Solution B

Figure 4.14: Spatial view of the macroclusters for Execution 4

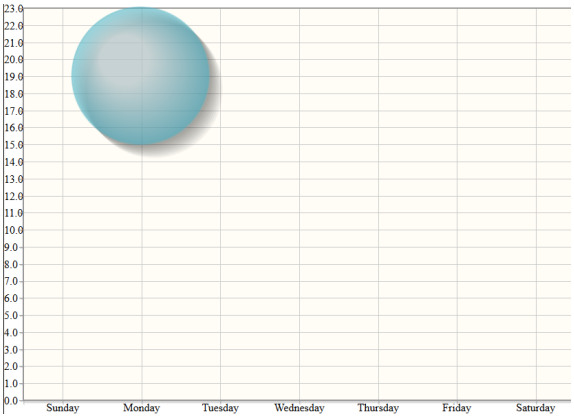
The temporal dimension is represented in Figures 4.15a, 4.15b and 4.15c.

The results from the content perspective are represented in Figures 4.16a, 4.16b and 4.16c.

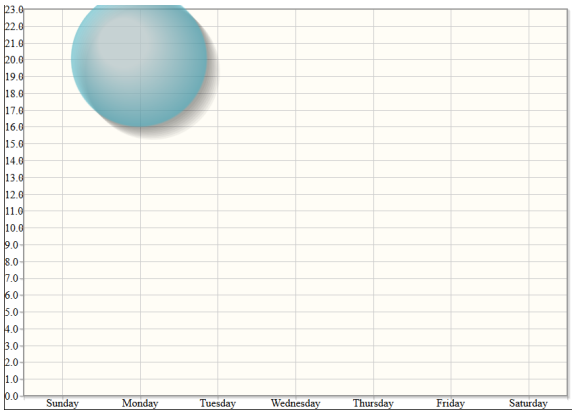
In this execution, we also observe significant variations regarding the number of microclusters, even though the difference related to the number of macroclusters is rather small. From the temporal perspective, the similarities between the results are noticeable, specially between the original solution and solution B (Figures 4.15a and 4.15c). We observe the same degree of similarity between all solutions regarding the content dimension. The spatial dimension is the one from which



(a) Original



(b) Solution A



(c) Solution B

Figure 4.15: Temporal view of the macroclusters for Execution 4

more differences can be observed, which is not very significant since this is the dimension being discarded in the weighting scheme of this execution.

(a) Original

(b) Solution A

(c) Solution B

Figure 4.16: Content view of the macroclusters for Execution 4

4.3.1.5 Overall Discussion

The first execution was the one which yielded more dissimilarities in terms of the number of microclusters, even though there was an approximation regarding the macroclustering. It makes sense that we find more differences in this execution since it considers all the three dimensions, while the others are considered pairwise. This is also the execution which reveals more differences from the spatial perspective of the macroclustering results. The second execution provides the most similar results in terms of the number of macroclusters, which is the same for every solution. However, it is the one which yields more differences regarding the content dimension. From the temporal perspective, we found its results to be very similar across all executions.

Overall, we find more similarities between both developed solutions than between each one and the original, which is not unexpected since they use the same unidimensional microclusters as basis. Solution A gives a better approximation overall, although the increase is minimal compared to Solution B.

We consider our results satisfactory and we believe our solutions have the capability of providing a good approximation to an approach without a dynamic weighting scheme. Although some

differences were observed, it's important to notice that the *MinPoints* parameter value of the microclustering algorithm was set to 2, which promotes more variability in terms of both the number of clusters and their features.

4.3.2 Evaluation

In this section we present the results obtained for one of the solutions in terms of the number of clusters created, the number of outliers and the clustering evaluation according to the implemented internal measures. The executions were performed for Solution A using the same weighting parameters as before (Table 4.2). Table 4.8 presents the number of microclusters and macroclusters obtained for the different weight combinations for the studied dimensions.

Table 4.8: Clustering results per weight combination

Execution	Micro	Macro	Macro Outliers
1	121	42	1
2	139	4	46
3	110	3	14
4	130	19	56

Tables 4.9 and 4.10 present the weighted evaluations of the obtained macroclusters according to the implemented internal measures. The former table corresponds to measures whose optimal value is the greatest of the index, unlike the latter, whose optimal value is the opposite.

Table 4.9: Clustering evaluation per weight combination I

Execution	Silhouette Coefficient	Dunn's Index	Tau	Gamma	Point-Biserial
1	0.44	0.018	-2.64E-6	-1	0.029
2	0.49	0.004	-8.73E-7	-1	0.039
3	0.50	0	-5.56E-7	-1	0.053
4	0.45	0.022	-1.36E-5	-0.99	0.05

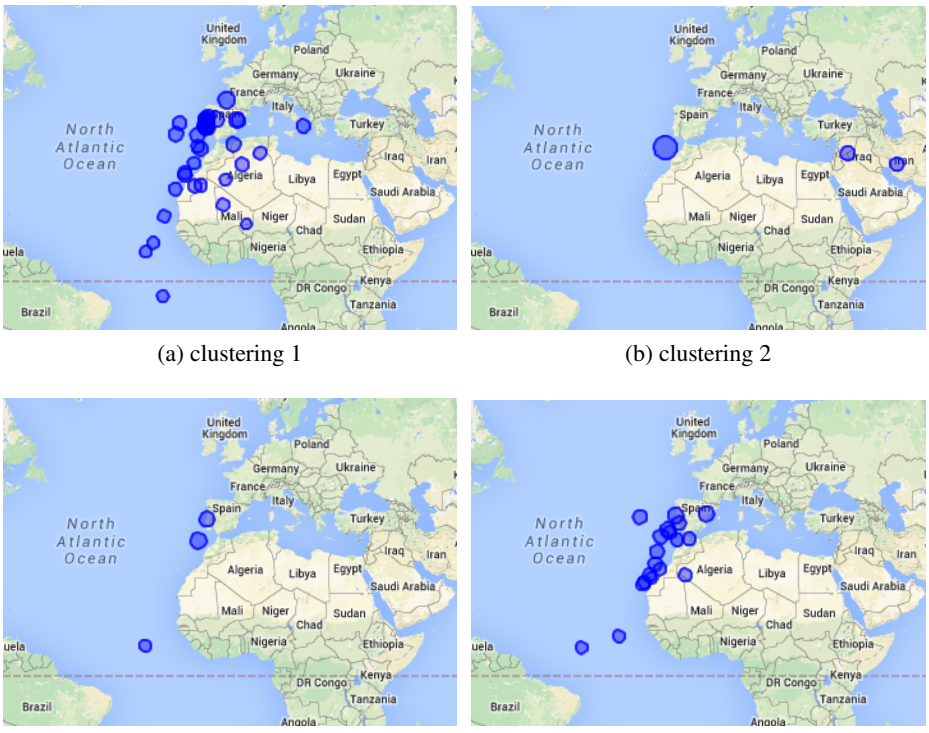
Table 4.10: Clustering evaluation per weight combination II

Execution	SSQ	G(+)	Davies-Bouldin Index	W/B
1	97.21	4.95E-7	1.725	0.965
2	46.65	1.66E-6	1.761	0.808
3	84.88	8.19E-7	1.535	0.946
4	138.68	1.56E-6	4.666	0.918

The second and third clusterings are those which perform better according to the evaluation measures from both groups. In terms of number of macroclusters, the aforementioned clusterings are those which have the least. This relates to their evaluation, since the resulting macroclusters are more concise and suffer an increase in their inter-cluster distance. This reduction is also supported by figures 4.17a,4.17b,4.17c and 4.17d.

We can also observe that including the content dimension results in macroclusters closer to Portugal (figures 4.17a, 4.17c and 4.17d) which makes sense since the content of the tweets is similar. This is not only due to the language of the tweets itself, but also because of the topics being discussed, with the most trending keywords being: *lisboa*, *porto*, *portugal*, *coimbra*, *aveiro*, *sintra*, *oeiras*, *centro*, *escola*, *parque*, *metro* and *universidade*. Including the spatial dimension results in macroclusters more distant from each other (figures 4.17a, 4.17b and 4.17c) in terms of the geolocation of the cluster center.

Regarding the temporal dimension, it is more related with the spatial dimension than the content dimension. That is the reason why the second clustering (Fig. 4.17b) has fewer macroclusters than the last (Fig. 4.17d).



(a) clustering 1

(b) clustering 2

(c) clustering 3

(d) clustering 4

Figure 4.17: Spatial view of the macroclusters

4.3.3 Performance

We compared the performance of the developed solution during the online clustering process against the solution presented in TweeProfiles2. Only one of the proposed solutions was used for testing, since both share the same logic for the online phase.

Table 4.11 shows the performance results measured in terms of CPU Time, User Time and Real Time, otherwise known as Wall Clock Time. These correspond to averaged results of ten

executions performed for both algorithms. For each execution, we defined a window of 30000 tweets, beginning at a random time point, which were fed to each algorithm as input data.

The tests were performed on a PC with Intel Core 2.5GHz CPU and 8GB RAM, running on Windows 8.1.

Table 4.11: Performance results

Measure	**TweeProfiles2**	**TweeProfiles4**
CPU Time	20.88s	101.984s
User Time	20.75s	101.390s
Real Time	21.60s	47.815s

Since our solution is based on a consensus clustering approach, it is inherently more computationally expensive. The expected elapsed time of running the unidimensional clusterers would be about three times the original if implemented in a sequential fashion. However, since their computations are independent, they were implemented in parallel. From a fully parallelizable approach, we would expect the elapsed time to be the same as the original. Nonetheless, the considered phase also comprises the computations performed by the *OverlapManager*, which incurs added complexity.

From table 4.11, we can see that our solution is nearly five times costlier than the original in terms of both CPU Time and User Time, but the ratio for the Real Time is only slightly above two. If implemented sequentially, we would not have this performance gain, since the Real Time would be close to the CPU Time, as we observe in the results retrieved from the original solution.

It should be noted that the relative performance is very sensitive to the input data, since the number of microclusters being formed in one aproach may be much different from the other. For a certain input, the original solution might output 5 multidimensional microclusters, while our solution could yield 1 microcluster for the spatial dimension, 2 for the temporal dimension and 20 for the content dimension. In this case, the content clusterer would be the bottleneck, potentially 4 times costlier than the original solution alone. This is the reason why we chose to perform ten executions with randomized tweet windows.

Chapter 5

Conclusions and Future Work

In this chapter we present a summary of the project, a discussion of the decisions taken and the limitations regarding the clustering process and the visualization. We finish the chapter with some suggestions for future work.

5.1 Summary

The goals of this dissertations were the proposal and development of a multidimensional clustering algorithm with support for a dynamic weighting scheme and the selection and implementation of a clustering evaluation process.

To accomplish these goals, a consensus clustering approach has been applied, which uses an extended version of the HybridDenStream clustering algorithm [Per14]. Two variants of the approach have been implemented, which differ from one another in the process used to generate the multidimensional microclusters. The first solution relies on a graph partitioning algorithm [KK98], which requires the number of partitions to be defined. The second solution relaxes this constraint by obtaining the partitions using DBSCAN [EKSX96].

Regarding the clustering evaluation, several internal evaluation measures have been implemented and adapted to support the multidimensional context of Twitter data. This has been achieved with the introduction of weighted distance functions for the computation of intra-cluster and inter-cluster distances.

The visualization tool has also undergone some changes. The first modification includes an interface to allow the user to dynamically alter his preferences. For this task, three sliders are provided (one for each dimension), which the user can drag independently to set the corresponding weight. The second modification comprises an interface to display the clustering evaluation results. These are presented in a time series chart, which is updated at frequent intervals and allows a dynamic selection of the evaluation measures to be displayed.

Several performance optimizations have also been performed including the filtering process from SocialBus [BOM+12], the data storage on the database, the clustering algorithm and the visualization tool.

This dissertation has been an extension of the TweeProfiles tool and all the goals have been met. The focus was on the improvement of the clustering algorithm in order to add more flexibility for the user to change the weight of each dimension in the clustering process and obtain results on-the-fly. As the data mining project gained maturity, it also became necessary to provide metrics to support the user in the analysis of the quality of the results produced by the clustering algorithm. In this sense, the inclusion of clustering evaluation measures has been another differentiating factor compared to previous versions.

5.2 Discussion

In the developed platform, there are some debatable aspects which are discussed below.

- **Weight Combinations**: The number of weight combinations for the experimental setup was chosen arbitrarily. The main reason for not increasing this number was the time consumed for running the process on the full dataset for each algorithm.
- **Partitioning**: For the graph partitioning algorithm applied in the first solution, we defined the number of partitions as the maximum number of microclusters on each dimension. This decision is purely arbitrary and its impact should be understood in more detail.
- **Evaluation**: For the evaluation of the clustering, it was necessary to keep a sample of tweets for each microcluster and macrocluster. The size of each sample has been capped at a fixed number, which has been chosen taking into account the performance of the evaluation task. This limit is a trade-off between a more accurate evaluation and a more efficient one in terms of computation performance.
- **Database**: A MySQL database was used to store the results of the mining and evaluation processes. This approach was chosen to facilitate the integration with the visualization tool and the adaptation of the clustering algorithm used in previous versions of the project. Even though the data is structured, the relational nature of this database system negatively affects the response time, given the current data access pattern.

5.3 Future Work

There are still aspects that could benefit from improvements, namely:

- **Clustering Algorithm**: The algorithm used for the unidimensional clustering was adapted from HybridDenStream [Per14], which is based on DenStream [CEQZ06]. Several proposals have been made as an improvement of the latter and it would be interesting to study their suitability in the context of Twitter data.

- **Temporal Distance**: For the temporal dimension, an Euclidean distance is used to measure the similarity between weekdays. As a consequence, Saturday is more similar to Sunday than to Friday, according to the used mapping. It would be interesting to study the impact of applying a distance measure which took into account the circular aspect of the days of the week.

- **Social Distance**: The social dimension was not included in TweeProfiles4. This is a problem that has still not been solved in a satisfactory way in the TweeProfiles project. Regarding the social graph, one of the issues is that the Twitter Streaming API does not include enough information to build one consistently. Although this data could be obtained through the Twitter REST API, this endpoint imposes rate limiting restrictions, which makes the process infeasible. Besides, even if the information was fully accessible, it would not be feasible to maintain the social graph in memory, given its complexity. Normalization of the distances would also be an issue, since they are unbounded. The problem with formulating an alternative process for studying the social interactions is that it would have to overcome the aforementioned matters and also be meaningful to TweeProfiles.

- **Performance**: Even though an effort was made to parallelize the clustering algorithm where possible, the workflow distribution might still be improved. For this, one could make use of an asynchronous message passing model, using a toolkit like Akka.[1]

- **Database**: For the storage of the clustering and evaluation results, a document-oriented database could improve the performance in terms of response time. This would reduce the detrimental effect of the large number of JOIN operations which are expensive in a relational database. MongoDB would be a good candidate, since it is already used for the data collection, therefore reducing the development stack.

[1] http://akka.io/

References

[ABpKS99] Mihael Ankerst, Markus M. Breunig, Hans peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. pages 49–60. ACM Press, 1999.

[ACF11] Cuneyt Gurcan Akcora, Barbara Carminati, and Elena Ferrari. Network and profile based measures for user similarities on social networks. In *Proceedings of the 2011 IEEE International Conference on Information Reuse and Integration, IRI 2011*, pages 292–298, 2011.

[ACF13] CuneytGurcan Akcora, Barbara Carminati, and Elena Ferrari. User similarities on social networks. *Social Network Analysis and Mining*, 3(3):475–495, 2013.

[Agg09] Charu C. Aggarwal. On High Dimensional Projected Clustering of Uncertain Data Streams. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1152–1154. IEEE, 2009.

[Agg13] CC Aggarwal. A Survey of Stream Clustering Algorithms. pages 229–252, 2013.

[AHWY04] 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 '04, pages 852–863. VLDB Endowment, 2004.

[AMR+12] 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*, 17, 2012.

[AW12] Amineh Amini and Teh Ying Wah. DENGRIS-Stream: A density-grid based clustering algorithm for evolving data streams over sliding window. In *International Conference on Data Mining and Computer Engineering*, pages 206–211, 2012.

[AW13] Amineh Amini and Teh Ying Wah. LeaDen-Stream: A Leader Density-Based Clustering Algorithm over Evolving Data Stream. *Journal of Computer and Communications*, 01(05):26–31, 2013.

[AWC+03] Charu C. Aggarwal, T. J. Watson, Resch Ctr, Jiawei Han, Jianyong Wang, and Philip S. Yu. *A Framework for Clustering Evolving Data Streams*. 2003.

[AWS14] Amineh Amini, Teh Ying Wah, and Hadi Saboohi. On Density-Based Data Streams Clustering Algorithms: A Survey. *Journal of Computer Science and Technology*, 29(1):116–141, 2014.

REFERENCES

[AWSY11] Amineh Amini, Teh Ying Wah, Mahmoud Reza Saybani, and Saeed Reza Aghabozorgi Sahaf Yazdi. A study of density-grid based clustering algorithms on data streams. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 3, pages 1652–1656. IEEE, 2011.

[AY08] Charu C. Aggarwal and Philip S. Yu. A framework for clustering uncertain data streams. In *Proceedings - International Conference on Data Engineering*, pages 150–159, 2008.

[Bar02] D Barbará. Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3(2):23–27, 2002.

[BF10] Albert Bifet and Eibe Frank. Sentiment knowledge discovery in twitter streaming data. *Discovery Science*, 2010.

[BGL10] Danah Boyd, Scott Golder, and Gilad Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2010.

[BH75] Frank B. Baker and Lawrence J. Hubert. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 70(349):31–38, 1975.

[BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010.

[BHP11] Albert Bifet, Geoffrey Holmes, and Bernhard Pfahringer. MOA-TweetReader: real-time analysis in twitter streaming data. *Discovery Science*, pages 46–60, 2011.

[BI02] Khoo Khyou Bun and M. Ishizuka. Topic extraction from news archive using TF*PDF algorithm. In *Proceedings of the Third International Conference on Web Information Systems Engineering, 2002.*, pages 73–82. IEEE Comput. Sci, 2002.

[BKC13] Vasudha Bhatnagar, Sharanjit Kaur, and Sharma Chakravarthy. Clustering data streams using grid-based synopsis. *Knowledge and Information Systems*, pages 1–26, 2013.

[BL12] Alexander Boettcher and Dongman Lee. EventRadar: A Real-Time Local Event Detection Scheme Using Twitter Stream. In *2012 IEEE International Conference on Green Computing and Communications*, pages 358–367. IEEE, 2012.

[BNG11] Hila Becker, M Naaman, and Luis Gravano. Beyond Trending Topics: Real-World Event Identification on Twitter. *ICWSM*, pages 438–441, 2011.

[BNJ12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2012.

[BOM+12] Matko Bošnjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. TwitterEcho - A Distributed Focused Crawler to Support Open Research with Twitter Data. *Proceedings of the WWW 2012, the 21st International Conference Companion on World Wide Web*, pages 1233–1239, 2012.

[Bru12] Axel Bruns. How Long Is a Tweet? Mapping Dynamic Conversation Networks on Twitter Using Gawk and Gephi. *Information, Communication & Society*, 15:1323–1351, 2012.

[CCFM97] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 626–635. ACM, 1997.

[CCZ07] Jian-Long Chang, Feng Cao, and Ao-Ying Zhou. Clustering evolving data streams over sliding windows. *Ruan Jian Xue Bao(Journal of Software)*, 18(4):905–918, 2007.

[CEQZ06] Feng Cao, Martin Ester, W Qian, and A Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. *SDM*, pages 326–337, 2006.

[CLOW11] Chun Chen, Feng Li, Beng Chin Ooi, and Sai Wu. Ti: An efficient indexing mechanism for real-time search on tweets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 649–660. ACM, 2011.

[Cor12] Mário Cordeiro. Twitter event detection: combining wavelet analysis and topic inference summarization. *Proceedings of Doctoral Symposium on Informatics Engineering*, 2012.

[CT07] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 133–142. ACM, 2007.

[CTB+12] Junghoon Chae, Dennis Thom, Harald Bosch, Yun Jang, Ross Maciejewski, David S. Ebert, and Thomas Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 143–152. IEEE, 2012.

[Cun13] Tiago Daniel Sá Cunha. Tweeprofiles: detection of spatio-temporal patterns on twitter. Master's thesis, Faculty of Engineering, University of Porto, Portugal, 2013.

[DAR09] Carlotta Domeniconi and Muna Al-Razgan. Weighted cluster ensembles. *ACM Transactions on Knowledge Discovery from Data*, 2(4):1–40, 2009.

[DB79] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(2):224–227, 1979.

[DCHR11] Wuzhou Dong, Jingyan Cui, Haitao He, and Jiadong Ren. Clustering over High-Dimensional Data Streams Based on Grid Density and Effective Dimension. *International Journal of Advancements in Computing Technology*, 3(8):154–162, 2011.

[Dek06] Anthony Dekker. Conceptual Distance in Social Network Analysis. *Science And Technology*, 6:1–34, 2006.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1959.

REFERENCES

[DLN+09] XuanHong Dang, Vincent Lee, WeeKeong Ng, Arridhana Ciptadi, and KokLeong Ong. An em-based algorithm for clustering data streams in sliding windows. In Xiaofang Zhou, Haruo Yokota, Ke Deng, and Qing Liu, editors, *Database Systems for Advanced Applications*, volume 5463 of *Lecture Notes in Computer Science*, pages 230–235. Springer Berlin Heidelberg, 2009.

[DLR77] 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*, 39(1):1–38, 1977.

[Dun74] J. C. Dunn†. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[EKSX96] Martin Ester, HP Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 1996.

[EOSX10] Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, 2010.

[Eze13] Chidube Donald Ezeozue. *Large-scale Consensus Clustering and Data Ownership Considerations for Medical Applications*. PhD thesis, 2013.

[Fis96] Doug Fisher. Iterative optimization and simplification of hierarchical clusterings. *J. Artif. Int. Res.*, 4(1):147–179, 1996.

[FPS09] Agostino Forestiero, Clara Pizzuti, and Giandomenico Spezzano. FlockStream: A Bio-Inspired Algorithm for Clustering Evolving Data Streams. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 1–8. IEEE, 2009.

[GLZT05] Jing Gao, Jianzhong Li, Zhaogong Zhang, and Pang-Ning Tan. An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection. In *Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'05, pages 420–425. Springer-Verlag, 2005.

[GMM+03] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[GMMO00] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 2000.

[GSIM09] Reza Ghaemi, Nasir Sulaiman, Hamidah Ibrahim, and Norwati Mustapha. A Survey : Clustering Ensembles Techniques. *Engineering and Technology*, 38:636–645, 2009.

[HCRG11] Haitao He, Lijuan Chen, Jiadong Ren, and Wenyan Guo. Probability Density Grid-based Online Clustering for Uncertain Data Streams. *Advances in Information Sciences and Service Sciences*, 3(8):204–211, 2011.

[HGRC11] Guoyan Huang, Wenyan Guo, Jiadong Ren, and Lijuan Chen. A Clustering Algorithm for Data Stream based on Grid-Tree and Similarity. *International Journal of Advancements in Computing Technology*, 3(9):17–24, 2011.

[HK06] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*, volume 54. 2006.

[HL76] Lawrence J Hubert and Joel R Levin. A general statistical framework for assessing categorical clustering in free recall. *Psychological bulletin*, 83(6):1072–1080, 1976.

[HLHR10] Guo-Yan Huang, Da-Peng Liang, Chang-Zhen Hu, and Jia-Dong Ren. An algorithm for clustering heterogeneous data streams with uncertainty. In *2010 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2059–2064. IEEE, 2010.

[HLRH10] Guoyan Huang, Dapeng Liang, Jiadong Ren, and Changzhen Hu. An algorithm for clustering uncertain data streams over sliding windows. In *Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference on*, pages 173–177. IEEE, 2010.

[HMR12] Guoyan Huang, Liyun Miao, and Jiadong Ren. Subspace Clustering over High-Dimensional Data Stream Based On Grid Density and Attribute Relativity. *International Journal of Advancements in Computing Technology*, 4(17):91–99, 2012.

[Hor07] Prodip Hore. *Scalable frameworks and algorithms for cluster ensembles and clustering data streams*. PhD thesis, University of South Florida, 2007.

[HSGS12] Marwan Hassani, Pascal Spaus, Mohamed Medhat Gaber, and Thomas Seidl. Density-based projected clustering of data streams. In *Scalable Uncertainty Management*, pages 311–324. Springer, 2012.

[Hua97] Zhexue Huang. Clustering large data sets with mixed numeric and categorical values. *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining,(PAKDD)*, pages 21–34, 1997.

[HW10] De-cai HUANG and Tian-hong WU. Density-based clustering algorithm for mixture data sets. *Control and Decision*, 3:020, 2010.

[HZ14] Haitao He and Jintian Zhao. A Density Grid-based Uncertain Data Stream Clustering Algorithm. 9:3619–3626, 2014.

[IDH12] Charlie Isaksson, Margaret Dunham, and Michael Hahsler. Sostream: Self organizing density-based clustering over data stream. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 7376 of *Lecture Notes in Computer Science*, pages 264–278. Springer Berlin Heidelberg, 2012.

[Jai10] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[JH09] Lin Jinxian and Lin Hui. A density-based clustering over evolving heterogeneous data stream. In *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009*, volume 4, pages 275–277, 2009.

[JMF99] A. K. Jain, M. N. Murty, and P J Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JTY08] Chen Jia, Chengyu Tan, and Ai Yong. A grid and density-based clustering algorithm for processing data stream. In *Proceedings - 2nd International Conference on Genetic and Evolutionary Computing, WGEC 2008*, pages 517–521, 2008.

[KABS11] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The clustree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29(2):249–272, 2011.

[Kar15] George Karypis. Family of graph and hypergraph partitioning software, 2015. [Online; accessed 4-January-2015].

[KBQ14] Farhan Hassan Khan, Saba Bashir, and Usman Qamar. TOM: Twitter opinion mining framework using hybrid classification scheme. *Decision Support Systems*, 57:245–257, 2014.

[KK98] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[KKJ+10] Philipp Kranen, Hardy Kremer, Timm Jansen, Thomas Seidl, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Clustering performance on evolving data streams: Assessing algorithms and evaluation measures within MOA. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1400–1403, 2010.

[KKJ+11] Hardy Kremer, Philipp Kranen, Timm Jansen, Thomas Seidl, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. An effective evaluation measure for clustering on evolving data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, pages 868–876. ACM Press, 2011.

[Koh82] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[Lee12] Chung-Hong Lee. Mining spatio-temporal information on microblogging streams using a density-based online clustering method. *Expert Systems with Applications*, 39(10):9623–9641, 2012.

[Li14] Jingxuan Li. *Mining the Online Social Network Data: Influence, Summarization, and Organization*. PhD thesis, 2014.

[LxHYfFc09] Liu Li-xiong, Huang Hai, Guo Yun-fei, and Chen Fu-cai. rDenStream, A Clustering Algorithm over an Evolving Data Stream. *2009 International Conference on Information Engineering and Computer Science*, 2009.

[Mah09] Alireza Rezaei Mahdiraji. Clustering data stream : A survey of algorithms. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 13(2):39–44, 2009.

[Mai15] André Maia. Tweeprofiles3: visualization of spatio-temporal patterns on twitter. Master's thesis, Faculty of Engineering, University of Porto, Portugal, 2015.

REFERENCES

[Mil81] Glenn W. Milligan. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46:187–199, 1981.

[MK10] M. T. Markou and P. Kassomenos. Cluster analysis of five years of back trajectories arriving in Athens, Greece. *Atmospheric Research*, 98:438–457, 2010.

[Mot14] Ivo Mota. Olho-passarinho: uma extensao do tweeprofiles para fotografias. Master's thesis, Faculty of Engineering, University of Porto, Portugal, 2014.

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[MyS14] MySQL. Mysql :: The world's most popular open source database. http://www.mysql.com/, 2014.

[NC07] Nam Nguyen and Rich Caruana. Consensus Clusterings. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 607–612. IEEE, 2007.

[NZP+12] Irene Ntoutsi, Arthur Zimek, Themis Palpanas, Peer Kröger, and Hans-Peter Kriegel. Density-based Projected Clustering over High Dimensional Data Streams. In *Proceedings of the Twelfth {SIAM} International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012.*, pages 987–998. {SIAM} / Omnipress, 2012.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[Per14] João Pereira. Tweeprofiles2: real-time detection of spatio-temporal patterns in twitter. Master's thesis, Faculty of Engineering, University of Porto, Portugal, 2014.

[Ran71] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336), 1971.

[RCH11] Jiadong Ren, Binlei Cai, and Changzhen Hu. Clustering over Data Streams Based on Grid Density and Index Tree. *Journal of Convergence Information Technology*, 6(1):83–93, 2011.

[REA15] REACTION. Welcome to socialbus — socialbus 2.0 documentation, 2015. [Online; accessed 10-June-2015].

[RHM10] Jiadong Ren, Changzhen Hu, and Ruiqing Ma. HCluWin: An algorithm for clustering heterogeneous data streams over sliding windows. *International Journal of Innovative Computing, Information and Control*, 6:2171–2179, 2010.

[RKT11] Aniket Rangrej, Sayali Kulkarni, and Ashish V. Tendulkar. Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th international conference companion on World wide web - WWW '11*, pages 111–112. ACM Press, 2011.

[RLW12] Hohyon Ryu, Matthew Lease, and Nicholas Woodward. Finding and exploring memes in social media. *Proceedings of the 23rd ACM conference on Hypertext and social media - HT '12*, page 295, 2012.

[RM09] Jiadong Ren and Ruiqing Ma. Density-based data streams clustering over sliding windows. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pages 248–252. IEEE, 2009.

[RMS09] Carlos Ruiz, Ernestina Menasalvas, and Myra Spiliopoulou. C-denstream: Using domain knowledge on a data stream. In João Gama, VítorSantos Costa, AlípioMário Jorge, and PavelB. Brazdil, editors, *Discovery Science*, volume 5808 of *Lecture Notes in Computer Science*, pages 287–301. Springer Berlin Heidelberg, 2009.

[Rod14] Tomy Rodrigues. Retweetpatterns: detection of spatio-temporal patterns of retweets. Master's thesis, Faculty of Engineering, University of Porto, Portugal, 2014.

[Rou87] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[SFB+13] Jonathan A Silva, Elaine R Faria, Rodrigo C Barros, Eduardo R Hruschka, André C P L F de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Computing Surveys*, 46(1):1–31, 2013.

[SHK10] Beaux Sharifi, Mark-Anthony Hutton, and Jugal K. Kalita. Experiments in Microblog Summarization. In *2010 IEEE Second International Conference on Social Computing*, pages 49–56. IEEE, 2010.

[SSB05] Ellen Spertus, Mehran Sahami, and Orkut Buyukkokten. Evaluating similarity measures: a large-scale study in the Orkut social network. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 678–684, 2005.

[SSVS14] S. Sarumathi, N. Shanthi, S. Vidhya, and M. Sharmila. A Comprehensive Review on Different Mixed Data Clustering Ensemble Methods. *International Journal of Mathematical, Computational, Physical and Quantum Engineering*, 8(8):1117–1127, 2014.

[SWCC13] Lidan Shou, Zhenhua Wang, Ke Chen, and Gang Chen. Sumblr: continuous summarization of evolving tweet streams. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*, page 533, 2013.

[SZ08] Mingzhou Song and Lin Zhang. Comparison of cluster representations from partial second- to full fourth-order cross moments for data stream clustering. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 560–569, 2008.

[TC09] Li Tu and Yixin Chen. Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3):12, 2009.

[TCT13] Li Tu, Peng Cui, and Keming Tang. A Density Grid-Based Clustering Algorithm for Uncertain Data Streams. In *2013 10th Web Information System and Application Conference*, pages 347–350. IEEE, 2013.

[TJ04] a Topchy and Ak Jain. A mixture model for clustering ensembles. *Proc. SIAM Int. Conf. Data Mining*, pages 379–390, 2004.

[TRA07] DimitrisK. Tasoulis, Gordon Ross, and NiallM. Adams. Visualising the cluster structure of data streams. In Michael R. Berthold, John Shawe-Taylor, and Nada Lavrač, editors, *Advances in Intelligent Data Analysis VII*, volume 4723 of *Lecture Notes in Computer Science*, pages 81–92. Springer Berlin Heidelberg, 2007.

[Twi14a] Twitter. Get statuses/sample | twitter developers, 2014. [Online; accessed 20-December-2014].

[Twi14b] Twitter. Welcome to twitter - login or sign up. https://www.twitter.com/, 2014.

[Twi15] Twitter. About twitter - https://about.twitter.com/company, 2015. [Online; accessed 10-June-2015].

[WHT13] Kapil Wankhade, Tasneem Hasan, and Ravindra Thool. A survey: Approaches for handling evolving data streams. In *Proceedings - 2013 International Conference on Communication Systems and Network Technologies, CSNT 2013*, pages 621–625, 2013.

[WND+09] Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3):14, 2009.

[WW10] Renxia Wan and Lixin Wang. Clustering over evolving data stream with mixed attributes. *Journal of Computational Information Systems (June 2010)*, 5:1555–1562, 2010.

[WXC09] Junjie Wu, Hui Xiong, and Jian Chen. Adapting the right measures for K-means clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*. ACM Press, 2009.

[YC11] Yun Yang and Ke Chen. Temporal data clustering via weighted clustering ensemble with different representations. *IEEE Transactions on Knowledge and Data Engineering*, 23:307–320, 2011.

[YLZY12] Yue Yang, Zhuo Liu, Jian-pei Zhang, and Jing Yang. Dynamic density-based clustering algorithm over uncertain data streams. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2664–2670. IEEE, 2012.

[YZ06] Chunyu Yang and Jie Zhou. HClustream: A Novel Approach for Clustering Evolving Heterogeneous Data Stream. In *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 682–688. IEEE, 2006.

[YZF12] Duan Yajuan, Chen Zhum, and W Ei Furu. Twitter topic summarization by ranking tweets using social influence and content quality. *Proceedings of COLING 2012: Technical Papers*, 4(96):763–780, 2012.

[ZCQJ08] Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, 15:181–214, 2008.

[ZGZ09] Chen Zhang, Ming Gao, and Aoying Zhou. Tracking High Quality Clusters over Uncertain Data Streams. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1641–1648. IEEE, 2009.

[ZK04] Ying Zhao and George Karypis. Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. *Machine Learning*, 55(3):311–331, 2004.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.

[ZS02] Yunyue Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment, 2002.

[ZZTG10] Peng Zhang, Xingquan Zhu, Jianlong Tan, and Li Guo. Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams. *2010 IEEE International Conference on Data Mining*, pages 1175–1180, 2010.