A Framework for Online Clustering Based on Evolving Semi-Supervision*

Guilherme Alves[1], Maria Camila N. Barioni[1], Elaine R. Faria[1]

[1]Faculdade de Computação (FACOM/UFU)
Universidade Federal de Uberlândia, MG - Brasil

{guilhermealves,camila.barioni,elaine}@ufu.br

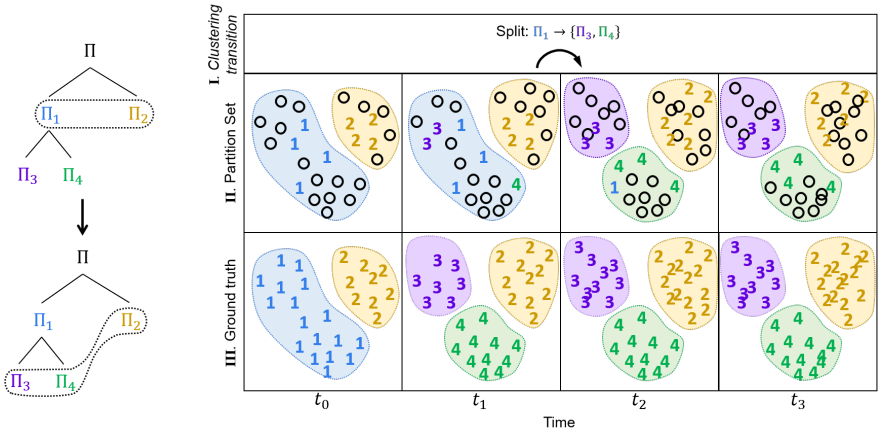
Abstract. *The huge amount of currently available data puts considerable constraints on the task of information retrieval. Automatic methods to organize data, such as clustering, can be used to help with this task allowing timely access. Semi-supervised clustering approaches employ some additional information to guide the clustering performed based on data attributes to a more suitable data partition. However, this extra information may change over time imposing a shift in the manner by which data is organized. In order to help cope with this issue, we propose the framework called* CABESS *(Cluster Adaptation Based on Evolving Semi-Supervision), for online clustering. This framework is able to deal with evolving semi-supervision obtained through user binary feedbacks. To validate our approach, the experiments were run over hierarchical labeled data considering clustering splits over time. The experimental results show the potential of the proposed framework for dealing with evolving semi-supervision. Moreover, they also show that our framework is faster than traditional semi-supervised clustering algorithms using lower standard semi-supervision.*

1. Introduction

The advent of ubiquitous computing is the one of the reasons most responsible for the tremendous increase in data generation. However, notwithstanding the valuable information contained in this huge amount of data, without the support of appropriate approaches the user may end up drowning in the data. Data mining approaches, such as clustering, aim at helping to obtain useful information from large datasets. Clustering approaches have been designed with the purpose of grouping data in order to detect patterns, to summarize information or help in the arrangement thereof [Barioni et al. 2014]. When there is some additional background knowledge available or a subset of labeled data, this additional information may be used to guide the clustering process to a desirable or more suitable data partition. Techniques that employ this additional information, referred to here as constraints, constitute the research area called semi-supervised clustering. Considering that the desired organization for the data may change over time, semi-supervised approaches may be useful for guiding clustering algorithms in the adaptation process.

The motivation for the work described herein is illustrated through the following example. Suppose that a marketing company aims to segment customers into groups, as well as monitor the evolution of these groups over time. In order to find these groups

*This work has been supported by CAPES, CNPq, and FAPEMIG.



(a) Grouping tree. (b) Execution results *vs.* ground truth over time.

Figure 1. An illustrative example. (a) Grouping trees showing the desired data partition highlighted by the dotted region in two different timestamps. (b) The evolution of the clustering structure as new data arrives and the constraint change over time. The numbers represent semi-supervision labels and the black circles represent the data instances without semi-supervision.

there is an available set of characteristics that represent each customer (such as monthly income, age) and additional information concerning the preferences of a small number of customers. Thus, a semi-supervised clustering algorithm is used at first. However, as time goes by, the preferences of the customers may change leading to new clustering structures. That is, at another time the preferences of the customers of a group change and cause the appearance of a new subgroup of costumers. Therefore, constraints, derived from the new preferences of the customers, impose a new clustering structure splitting a previous group into two newer groups. Noteworthy here is that more customers may still appear, but the characteristics do not change, that is, the representation of the customers does not change over time only their preferences.

The previously described scenario is an instance of the problem addressed herein. Figure 1 illustrates the evolution of the clustering structure as the constraints change over time. Figure 1a depicts a grouping tree, describing the different levels of the possible data organization. The desired organization is highlighted by the dotted regions. Thus, one observes that there are two possible types of targeted data organization. On the left-hand side, Figure 1b describes the following sets over time: (I) the cluster transitions detected, (II) the resulted partition set after a semi-supervised clustering, and (III) the ground truth, that is, the optimal partition set desired by a user at each timestamp. By analyzing the clustering results from the timestamp t_0 to t_3, one notes that the user criteria for partitioning the data has changed according to the level of hierarchy desired (Figure 1a). The feedbacks define constraints that guide the clustering process (see numbers as labels in Figure 1b).

Considering that the user constraints may change over time, the following clustering transitions may occur: birth, split or merge. The work described herein is focused on splits and assumes that the evolving semi-supervision can impact the modifications in the clustering structures over time, implying in new splits. Therefore, the main goal of our work is to provide a framework that is able to use and maintain semi-supervision

correctly to enable efficient and effective online clustering processes. The main contributions of the work described herein can be summarized as follows: (1) The introduction of CABESS (Cluster Adaptation Based on Evolving Semi-Supervision) (Section 3), a framework which aims at allowing efficient and effective online clustering using semi-supervision in the form of feedback. (2) The proposal of a strategy that extracts semi-supervision information from feedback given in the form of labels (Section 4.1). (3) An approach to keep the labels consistent over time (Section 4.2).

This paper is organized as follows. Section 2 presents the related works. Section 3 formalizes the problem addressed in this paper and describes our framework CABESS. Section 4 presents Pointwise CABESS, an instantiation of our framework, and its particularities. Section 5 explains the experimental setup and Section 6 presents and discusses the experimental results. The conclusions and the future works are discussed in Section 7.

2. Related Work

The strategies employed in the framework CABESS are related to the research performed on semi-supervised clustering with online learning and tracking clustering. Therefore, Section 2.1 briefly presents the fundamental concepts of batch and online semi-supervised clustering. Following on from this, Section 2.2 describes the related works regarding monitoring evolving clusters.

2.1. Semi-supervised Clustering

The semi-supervised clustering approaches described in the scientific literature provide different approaches in order to allow for the guiding of the clustering process, and the obtainment of meaningful clusters [Barioni et al. 2014]. Generally, traditional semi-supervised clustering algorithms are divided into two categories: similarity-adapting methods that adapt the similarity measure employed in the clustering process in order to satisfy the labels or constraints in the data; and methods that employ labels or constraints provided by the users to modify the clustering assignment step [Basu et al. 2008].

There is also a variety of ways to express and to obtain these constraints. A substantial part of the works described in the scientific literature explores the specification of these restrictions in the form of instance-level constraints [Bilenko et al. 2004], which are composed of two types of constraints: *must-link* (ML) and *cannot-link* (CL). A ML constraint indicates that two data instances must be in the same cluster. A CL constraint implies that two data instances must not be in the same cluster. Other means for expressing constraints are attribute-level constraints [El Moussawi et al. 2016], cluster-level constraints [Dubey et al. 2010], relative constraints [Liu et al. 2011], and labels [Castellano et al. 2013].

Several research works have extended classical clustering methods to be able to deal with additional information. MPCK-Kmeans [Bilenko et al. 2004], for example, is an extension of the widely used K-MEANS algorithm [Jain and Dubes 1988] that incorporates metric learning and semi-supervision in the form of ML and CL. C-DBScan [Ruiz et al. 2007] and SSDBScan [Lelis and Sander 2009] extended the well known DBScan algorithm [Ester et al. 1996]. While the first one deals with pairs of constraints, the latter receives a set of labeled instances as semi-supervision input. These methods require that

the entire dataset and all the semi-supervision information be available at the beginning of the data clustering process.

Examples of online approaches are described in [Castellano et al. 2013] and [Lai et al. 2014]. The first presents an extension of SSFCM for automatic image annotation based on semi-supervised clustering. It assumes that instances belonging to some clusters are available over time and can be clustered as chunks. The authors in [Lai et al. 2014] propose a semi-supervised clustering technique that improves the clustering effectiveness interactively from user feedbacks in the form of *positive*, *negative* and *displacement* feedback. A *positive* feedback means that a data instance was correctly assigned to the current cluster. A *displacement* feedback means that a data instance must not belong to the current cluster and points to the appropriate cluster. Although feedback arrives continually, [Lai et al. 2014] consider that it has all data instances at the beginning of the execution. Furthermore, all of these approaches assume that the user feedback is always coherent among different iterations.

2.2. Tracking and Evolving Clustering

An important research issue that has recently come to be explored is related to the understanding of the clusters behavior over time. The MONIC [Spiliopoulou et al. 2006] and MClusT [Oliveira and Gama 2010] frameworks are some well-known examples that deal with this issue. The first of these two frameworks models and tracks cluster changes in order to understand the nature of the change. The cluster transitions are formalized and an algorithm for detecting transitions is proposed. MClusT builds a bipartite graph for modeling the cluster transitions, where vertexes are clusters and edges represent the relationship between a pair of clusters. Thus, a user may well attain knowledge into what happened with instances of a cluster that split at a particular period. Other approaches have been proposed, such as [Pereira and Moreira 2016], but it is noteworthy that none of these explored the analysis of the evolution of constraints over time.

3. The CABESS Framework

We revisited the semi-supervised clustering model described in [Lai et al. 2014] in order to propose CABESS, a generic framework that aims at using a limited amount of additional information to provide efficient and effective online clustering. It also uses an approach that detects external clustering transitions in order to manage evolving semi-supervision over time. CABESS receives as input a massive sequence of data instances $(x_1, ..., x_n)$, $\mathcal{D} = \{x_i\}_{i=1}^{n}$, and a limited sequence of feedbacks $\mathcal{F} = \{f_i^j\}_{j=1}^{m}$, $|\mathcal{F}| \ll |\mathcal{D}|$. Each instance is described as a d-dimensional feature vector $x_i = [x_i^j]_{j=1}^{d}$ that belongs to a continuous feature space Ω. Each user feedback f_i^j relates to a data instance $x_i \in \mathcal{D}$ and expresses either two types of feedbacks: **positive** and **displacement**.

The clustering process performed by CABESS contains five steps (see Figure 2). Considering that no feedback information is available at first, the clustering process starts summarizing the dataset using a micro-clustering approach based on CF-Vectors (step 1). Among the approaches that can be used in these situations, one finds BIRCH [Zhang et al. 1996] and CluStream [Aggarwal et al. 2003]. A CF-Vector summarizes data of a group of N instances as a triplet $CF = (N, \overrightarrow{LS}, SS)$ where $\overrightarrow{LS}$ is the linear sum of the data instances and SS is the sum of the squared data instances. CF-vectors have important

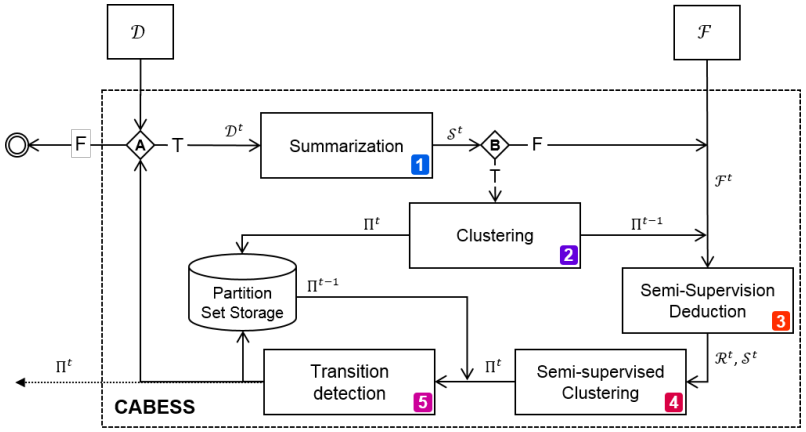


Figure 2. The framework CABESS. Test A verifies if new instances have been generated or the user has been satisfied with the cluster quality (feedback generation). Test B verifies if it is the first clustering performed.

properties that are useful in CABESS. It is easy to compute centroid, radius, and diameter of any group of the data instances from the sufficient statistics of a CF-vector. Moreover, a CF-vector also has properties that allow us to update the group information by using only the sufficient statistics, for example: adding a new data instance, merging two clusters, and subtracting a data instance from a group.

In the second step, CABESS performs a macro-clustering over the resulted summarized representation $\mathcal{S}^t$ using an unsupervised clustering algorithm. This step can employ either DBSCAN or K-MEANS algorithms. Following these steps, CABESS allows the user to provide instance-level feedback regarding a data partition set Π^t, which is used to adjust the clustering process performed at the next timestamp $(t+1)$. In order to do so, the instance-level feedbacks are used to deduce the summarized-level constraints $\mathcal{R}^t$ (step 3). As the deduction process is a contribution derived from this particular work, it is detailed in Sections 4.1 and 4.2.

After the semi-supervision deduction, a semi-supervised clustering algorithm is used to re-organize the data partitions (step 4). Examples of algorithms that are suitable to use in this step are SSDBSCAN and MPCK-MEANS. There arises the need to emphasize that the semi-supervision required by the semi-supervised clustering algorithm must be the same type as the semi-supervision deduced in step 3. Thus, if we choose SSDBSCAN, step 3 must deduce semi-supervision in the form of labels, as SSDBSCAN deals with labels. The last step is responsible for detecting transitions between the current partition set Π^t and the previous partition set Π^{t-1} (step 5). Both MONIC and MCLusT can be chosen for use in this step. The CABESS clustering process finishes when the resulted partition set satisfies the user and when there are no new instances generated in $\mathcal{D}$.

It is worth mentioning here that the clustering algorithms performed in steps 2 and 3 consider the summarized information obtained from the original data. After the clustering process, the user can provide feedback. Thus, a summarized data instance $s_1 \in \mathcal{S}^t$ of cluster $\Pi_i^t \in \Pi^t$, in fact, may represent more than one data instance. Hence, semi-supervision information involving s_1 imposes constraints on all instances represented by s_1.

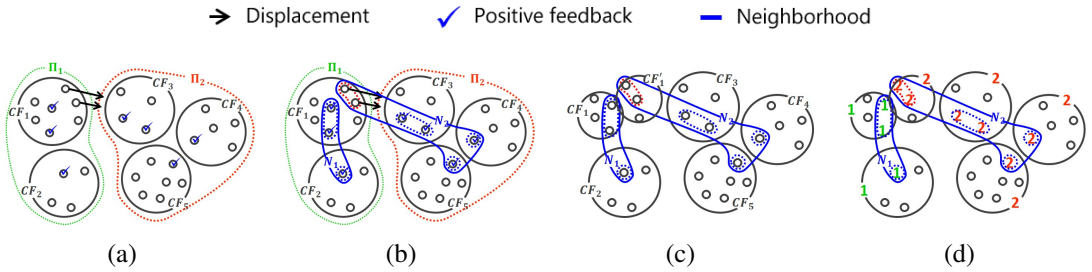


Figure 3. Example of the extraction of the labels from feedback (a,b) and the deduction labels from instance-level to summarized-level (c,d).

4. Pointwise CABESS

In order to illustrate an instantiation of the CABESS framework we will present the Pointwise CABESS. This CABESS instance employs the BIRCH algorithm in order to summarize the data (step 1). The other algorithms used in Pointwise CABESS were the DBSCAN in step 2 to perform the first clustering without any semi-supervision, then SSDBSCAN was adopted to detect clusters with summarized-level labels in step 4, and in the final step, MONIC was used to detect external transitions.

Pointwise CABESS deals with labels as semi-supervision information. In the next section, we explain how Pointwise CABESS extract instance-level labels and how it deduces the summarized-level labels.

4.1. Extracting labels from feedbacks

The summarized-level labels are obtained in two phases, firstly the instance-level labels are extracted from feedback (see Figures 3a, 3b) and then the summarized-level labels are deduced from instance-level labels (see Figures 3c, 3d).

Instance-level labels. The main idea behind the extraction process concerns to the concept of neighborhood. A neighborhood is defined as a set of instances that must be in the same cluster. CABESS computes the neighborhoods as in our main reference [Lai et al. 2014] by following two rules. The first states that all instances of the same cluster that received positive feedback at a previous timestamp are assigned to the equivalent neighborhood. The second states that all instances of different clusters that received displacement feedback at a previous timestamp are assigned to the neighborhood of the initial rule, since the the destination/actual cluster is the same as the instances in the initial rule. Following this, we assign identical labels for instances that are in the same neighborhood. One notes that in Fig. 3b the neighborhood associated with Π_1 will be assigned to label 1 at the end of the process (Fig. 3d)

Summarized-level labels. The deduction of summarized-level labels is performed as a propagation task. CABESS aims to assign labels to Cluster Feature Vectors (CFs) in Fig. 3c. For each summarized instance that contains labeled instances, we assign the instance-level label to the summarized-level label. If one of the summarized instances has labeled instances with different labels, then we need to split it in order to obtain purified summarized instances, i.e., summarized instances that contain only the same label in labeled instances. Noted in Fig. 3d is that CF_1 and CF_2 received label 1, as these summarized instances which are in neighborhood N_1.

Table 1. Datasets employed in the experiments.

Name	# instances	d	# classes	Reference	Type
DB_7	9,050	2	8	[Silva et al. 2015]	
SYN_3	5,000	2	3	streamMOA	Synthetic
SYN_4	10,000	3	5	streamMOA	
FROGS	1,484	8	4	[Colonna et al. 2016]	
IPEA	5,564	5	27	IPEA	Real
$KDD'99_5$	24,692	19	11	UCI	

4.2. Dealing with obsolete labels

Obsolete labels are labels assigned to instances, for which the clusters do not exist anymore. The data instances are still active but they are assigned to other clusters. These labels appear when an external cluster transition happens. For example, let us suppose there is a cluster Π_1^0 at the timestamp $t = 0$ and also that at the next time, Π_1^0 splits into two new clusters Π_{10}^1 and Π_{11}^1. Thus, the label associated with Π_1^0 is not valid at $t = 1$, because it does not help the semi-supervised clustering process.

In order to minimize the problem of obsolete labels, CABESS adopts a detector of transitions. This approach aims at allowing for better neighborhood management. Neighborhoods are responsible for generating new labels and removing obsolete labels. Thus, when a cluster survives both neighborhood and associated labels are preserved. When a split transition is detected, the label associated with the previous cluster is removed and the neighborhood is divided in order to create two or more neighborhoods and new labels are generated accordingly.

5. Experiments

The experiments were conducted using 6 datasets[1], 3 with real data and the other 3 with synthetic data. The details concerning each one are summarized on Table 1. All these datasets share a common feature that allows us to run our experiments, each of its data instances is multi-labeled where labels are in a hierarchical structure.

From the three synthetic datasets, with Gaussian distribution, $\mathbf{SYN}_3$ and $\mathbf{SYN}_4$ were generated using the `RandomRBFGenerator` available at `streamMOA`[2], an interface of MOA[3] (Massive Online Analysis) developed for the R programming language. Let us now consider a third synthetic dataset, denominated as DB7, this is a 2D dataset that contains clusters of different spatial distributions. The grouping trees for the abovementioned synthetic datasets were simulated according to the distance among clusters. Thus, we consider two or more clusters belonging to a large cluster if they are closer than the other clusters.

Considering the real datasets, in **IPEA** dataset[4], each tuple corresponds to one of the 5,564 Brazilian cities among within its Federative unit and Region, and is composed of 5 features with localization information (latitude and longitude) and continuous values related to the following development indexes: IDHM-Longevity, IDHM-Education,

[1]Available at: http://guilhermealves.eti.br/research/data/

[2]Documentation available at: https://cran.r-project.org/web/packages/streamMOA/streamMOA.pdf.

[3]A framework for data stream mining: http://moa.cms.waikato.ac.nz/

[4]Brazilian Institute of Applied Economics Research, IPEA: http://www.ipeadata.gov.br

Table 2. Parameters setting for each dataset.

Algorithm	Parameter	IPEA	KDD'99$_5$	FROGS	DB$_7$	SYN$_3$	SYN$_4$
BIRCH	B and L	500	10k	500	500	50	50
	T	0.07	0.01	0.25	0.01	0.0075	0.0075
DBSCAN	eps	0.075	0.02	0.5	0.075	0.075	0.075

and IDHM-Income. In the **FROGS** dataset, each tuple corresponds to an audio recording of *Anura*, an order of the Amphibian class, and is composed using information concerning specie, genus, and family. **KDD'99**$_5$[5] is a dataset generated by sampling 5% of data instances from the 10% subset of the original KDD Cup 99 dataset, maintaining the proportion of instances in each class. Moreover, this sampled dataset was processed with PCA in order to reduce the dimensionality, as it is the dataset with the highest number of dimensions. We consider two levels of labels found in **KDD'99**$_5$; (1) if the access is an intrusion or not, and (2) if it is an intrusion, what is the type of intrusion.

The main goals of the experiments were to assess the effectiveness and the efficiency of our proposed framework instance, Pointwise CABESS, in comparison with three baseline approaches, two semi-supervised and one unsupervised, considering the four questions presented in Sections 6.1, 6.2, 6.3, and 6.4.

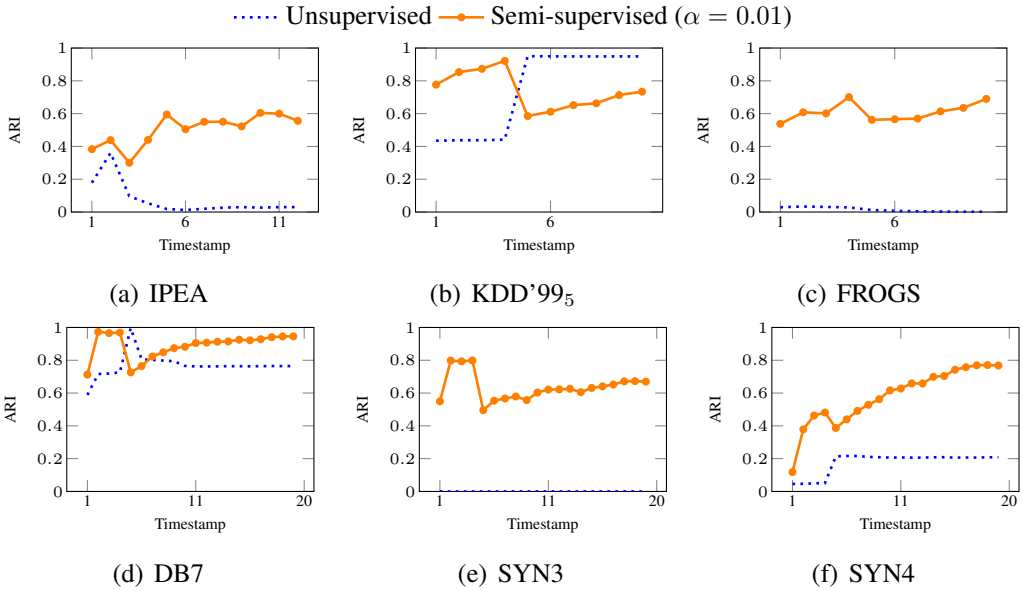
The **Unsupervised Approach** consists in periodically executing a clustering algorithm without any semi-supervision. In our experiments, we adopted DBScan, an efficient well-known unsupervised clustering algorithm. The semi-supervised approaches were run considering two strategies, those being static and window based. The **Static Approach** consists in periodically applying a semi-supervised clustering algorithm. In our experiments, we adopted SSDBScan providing the true labels as semi-supervision. Notice that this approach does not discard any label over time. The **Window-based Approach** is a variation of the previous approach, where instead of executing the clustering algorithm over all the semi-supervision set, we remove old semi-supervision information (labels).

All experiments were implemented within the same platform using the Java programming language. In order to evaluate the effectiveness of the approaches we compared the clustering results against the optimal partition set using the Adjusted Rand Index (ARI) [Hubert and Arabie 1985]. ARI is an external criteria indicated when researchers have a-priori information about the desirable partition set. In our experiments, the desirable partition set is obtained from the true labels in the datasets. For each timestamp t only one label is considered valid for a data instance according to the grouping tree.

Each experiment over Pointwise CABESS was performed based on the Prequential Protocol. Hence, the effectiveness of Pointwise CABESS is evaluated before the user provides current feedback. The other approaches are evaluated periodically without any delay between the new partition set and new feedback.

Considering that our datasets have no online arrival for instances neither for feedback, and our main reference simulates the online aspects of getting user binary feedback, we also simulated the temporal aspects of the data instances and the user interaction phase for obtaining the semi-supervision. Hence, the arrival of the data instances and the user feedback are given according to the uniform distribution. Furthermore, we insert an ex-

[5]UCI KDD archive: http://kdd.ics.uci.edu/



(a) IPEA (b) KDD'99_5 (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

Figure 4. Effectiveness assessment.

ternal cluster transition, according to the grouping tree, at $t = 5$ in order to evaluate the ability of the approaches to adapt to new clusters. In each dataset, one or more clusters are specialized into two or more new clusters and feedback, which are generated according to these new clusters.

Table 2 shows the parameters used in BIRCH and DBSCAN algorithms for each dataset. For the other algorithms, the parameters adopted are the SSDBSCAN in step 4, which used the same value of DBSCAN for $minPts = 2$, and in the final step, MONIC was run using $\tau = 0.7$ and $\tau_{split} = 0.1$. The experiments were run on an Intel Core i7 (3.4 GHz) with 12 GB of RAM, SATA3 HD of 1.31 TB (7,200 rpm) on Windows 7 x64.

6. Results and Discussion

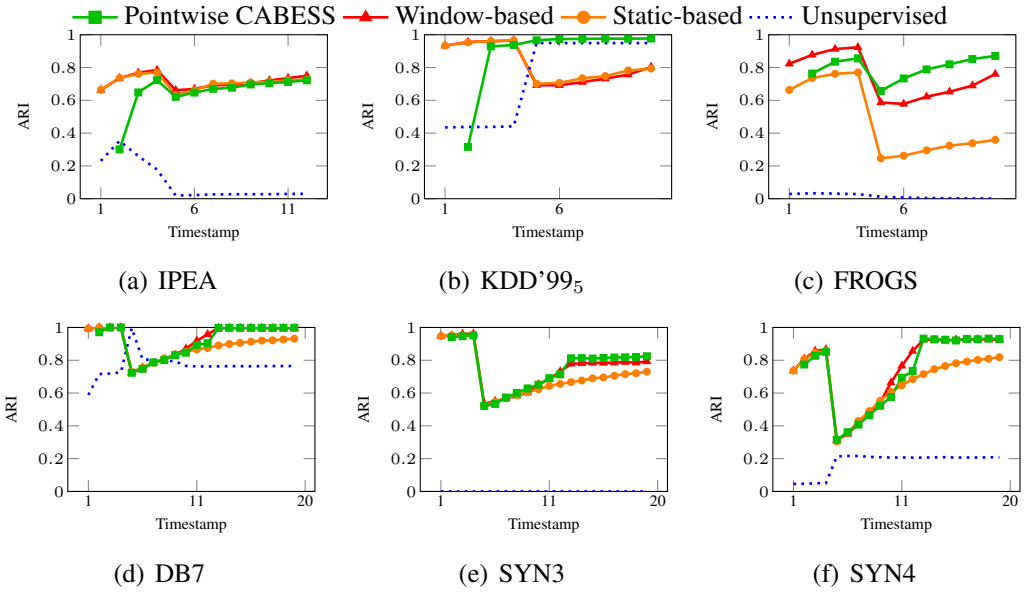
Sections 6.1, 6.2, 6.3 and 6.4 present the discussion of the results obtained throughout the experiments, which took into consideration four questions.

6.1. How accurately does semi-supervision aid on clustering effectiveness when there are external clustering transitions over time? (Q1)

In order to answer Q1, we chose to compare the results obtained running the Unsupervised Approach and the Static Approach on each dataset. This question is important for determining whether a clustering approach can benefit from semi-supervision information. Figure 4 shows ARI considering the semi-supervision rate α set to 1% for the Static Approach. One notes that semi-supervision performs better in most of the datasets. In KDD'99_5 semi-supervision improved the clustering quality before the cluster transition ($t = 5$). Therefore, the results achieved corroborate that low rates of semi-supervision contribute to the obtainment of a more effective clustering in a majority of the scenarios.

6.2. Are there major differences between clustering effectiveness when using semi-supervised clustering approaches based on feedback and labels? (Q2)

One of the main contributions of our approach is its ability of dealing with semi-supervision specified through feedback. The previous question focused on understanding whether online clustering could benefit from semi-supervision. Next, we evaluated



(a) IPEA (b) KDD'99_5 (c) FROGS

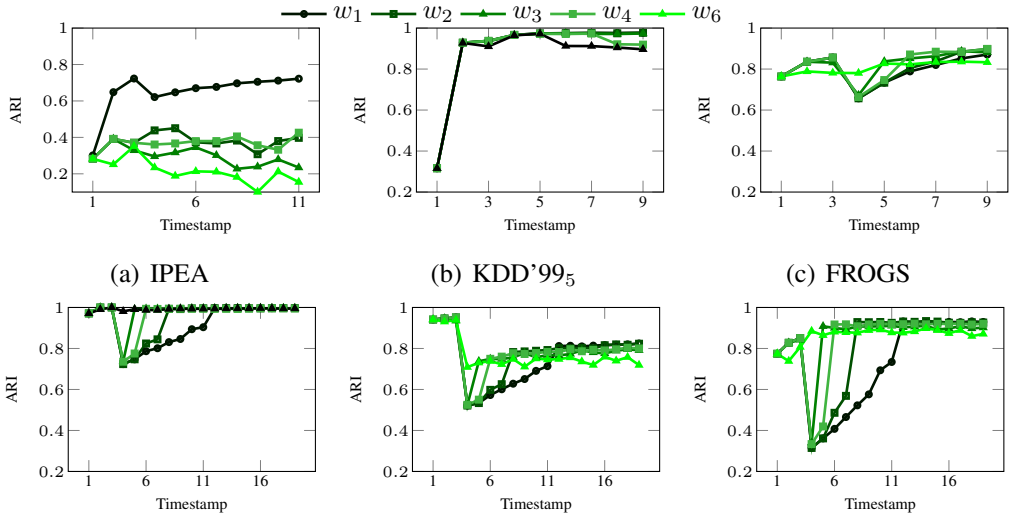
(d) DB7 (e) SYN3 (f) SYN4

Figure 5. Effectiveness comparison between Pointwise CABESS and the baseline approaches ($\alpha = 0.1$ and w_1).

the performance of traditional semi-supervised approaches against our approach. Figure 5 shows ARI for all the baseline approaches considered and the Pointwise CABESS. In this experiments, we adopted $\alpha = 0.1$ and the largest semi-supervision window size w_1. By analyzing the obtained results, one notes that Pointwise CABESS performed better than other approaches above all after the external transition when considering the datasets KDD'99_5 and FROGS. For the other datasets, the effectiveness obtained for the semi-supervised approaches were equivalent. Emphasis is given here to the fact that the semi-supervision information used by Pointwise CABESS is lower standard when compared to the semi-supervision used by the other approaches. Note that our framework receives feedback instead of labels. Thus, it needs to infer and maintain labels correct over time. The experimental results obtained showed that there are no major differences in the clustering quality by using using semi-supervision in the form of labels or inferring labels from feedback, as is the case in our approach.

6.3. How the feedback window size variation affects semi-supervision information and clustering effectiveness? (Q3)

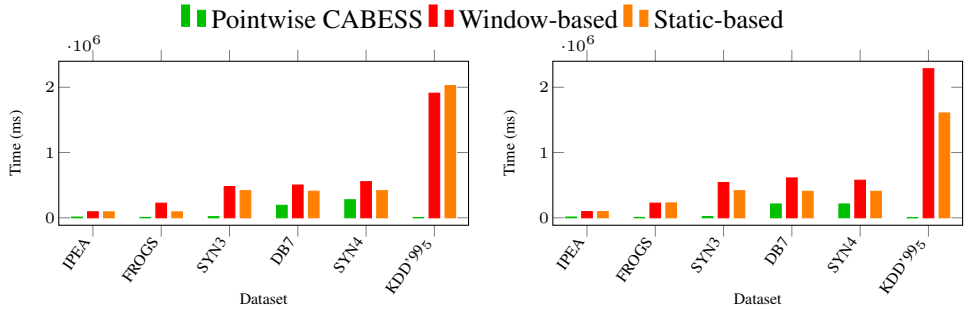
In data stream mining literature there is a common assumption that small windows mitigate the adaptation time of the clustering process [Gama 2010]. In order to answer Q3 we adopted six different semi-supervision window sizes to assess it. Figure 6 shows the Pointwise CABESS ARI for each window setting: $w_6 < w_4 < w_3 < w_2 < w_1$. Some curves are overlapped, e.g, w_6 and w_5 in DB7. Analyses of the obtained results shows that the small window w_6 showed quicker adaptation but in some cases, the quality was unstable. This occurs due to some change of instance over the cluster at each iteration. Note that when we have a large window size w_1, there is a notable slow adaptation but we can see stable results. The use of the detector of transitions is responsible for producing this behavior. Moreover, running Pointwise CABESS over small label windows is the same as the Window-based Approach, due to the fact that a small window size mitigates the benefic effects obtained when using the detector of transitions.



(a) IPEA (b) KDD'99_5 (c) FROGS

(d) DB7 (e) SYN3 (f) SYN4

Figure 6. Effectiveness assessment considering different window sizes.



(a) $\alpha = 0.1$ and w_1 (b) $\alpha = 0.01$ and w_6

Figure 7. Mean runtime to cluster different datasets using high (a) and low (b) semi-supervision rates.

6.4. How efficient is our approach compared to existing semi-supervised approaches? (Q4)

Another contribution of our approach is its ability to summarize data instances and semi-supervision information. Here, we quantified the efficiency of our approach and compared it against the efficiency of the existing semi-supervised algorithms. Figure 7 shows the run times for Pointwise CABESS and the other semi-supervised approaches. Through an analysis of these results, one observes that our framework instance performed faster than the other approaches. The main reason for this behavior is the summarizing algorithm used in its first step. Then, the semi-supervised clustering algorithm considers a smaller number of summarized instances compared to the number of instances considered by the other semi-supervised approaches.

7. Conclusion

The goal of the proposed framework presented herein, CABESS, is to assist online clustering in coping with external transitions. Our experiments and analyses were performed driven by four research questions on three real datasets and three synthetic datasets. The results showed that our approach presents a higher efficiency when compared to other semi-supervised approaches while keeping an equivalent effectiveness.

With CABESS, we can put forward new directions to extend traditional semi-supervised clustering techniques. Our work could be extended to explore other types of semi-supervision information such as instance-level constraints (must-link and cannot-link) that will allow for the employment of other types of semi-supervised clustering algorithms into CABESS. Another interesting direction for future work is to tackle other strategies for detecting transitions and to explore other types of external transitions.

References

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2003). A framework for clustering evolving data streams. In *VLDB*, pages 81–92. VLDB Endowment.

Barioni, M. C. N., Razente, H., Marcelino, A. M. R., Traina, A. J. M., and Traina, C. (2014). Open issues for partitioning clustering methods: An overview. *WIREs Data Min. and Knowl. Disc.*, 4(3):161–177.

Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman and Hall/CRC.

Bilenko, M., Basu, S., and Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *ACM ICML*, page 11, New York, NY, USA.

Castellano, G., Fanelli, A. M., and Torsello, M. A. (2013). Shape Annotation by Incremental Semi-supervised Fuzzy Clustering. In *WILF*, volume 8256 of *LNCS*, pages 193–200. Springer.

Colonna, J. G., Gama, J., and Nakamura, E. F. (2016). Recognizing Family, Genus, and Species of Anuran Using a Hierarchical Classification Approach. pages 198–212. Springer, Cham.

Dubey, A., Bhattacharya, I., and Godbole, S. (2010). A Cluster-Level Semi-supervision Model for Interactive Clustering. pages 409–424.

El Moussawi, A., Cheriat, A., Giacometti, A., Labroche, N., and Soulet, A. (2016). Clustering with Quantitative User Preferences on Attributes. In *IEEE ICTAI*, pages 383–387.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231. AAAI Press.

Gama, J. (2010). *Knowledge discovery from data streams*. Chapman & Hall/CRC.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, USA.

Lai, H. P., Visani, M., Boucher, A., and Ogier, J.-M. (2014). A new interactive semi-supervised clustering model for large image database indexing. *Pattern Recognition Letters*, 37(1):94–106.

Lelis, L. and Sander, J. (2009). Semi-supervised Density-Based Clustering. In *IEEE ICDM*, pages 842–847.

Liu, E. Y., Zhang, Z., and Wang, W. (2011). Clustering with relative constraints. In *ACM SIGKDD*, page 947, New York, NY, USA.

Oliveira, M. D. and Gama, J. (2010). Bipartite graphs for monitoring clusters transitions. In *IDA*, pages 114–124. Springer.

Pereira, G. and Moreira, J. (2016). Monitoring clusters in the telecom industry. In *New Advances in Information Systems and Technologies*, pages 631–640. Springer.

Ruiz, C., Spiliopoulou, M., and Menasalvas, E. (2007). *C-DBSCAN: Density-Based Clustering with Constraints*, volume 4482 of *LNCS*. Springer.

Silva, W. J., Barioni, M. C. N., de Amo, S., and Razente, H. L. (2015). Semi-supervised clustering using multi-assistant-prototypes to represent each cluster. In *SAC*, pages 831–836, New York.

Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., and Schult, R. (2006). MONIC. In *ACM SIGKDD*, page 706, New York, NY, USA. ACM Press.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). BIRCH: An Efficient Data Clustering Method for very Large Databases. *ACM SIGMOD Record*, 25(2):103–114.