Tracking the Evolution of Financial Time Series Clusters

Davide Azzalini
Politecnico di Milano
davide.azzalini@polimi.it

Fabio Azzalini
Politecnico di Milano
fabio.azzalini@polimi.it

Mirjana Mazuran
INRIA
mirjana.mazuran@inria.fr

Letizia Tanca
Politecnico di Milano
letizia.tanca@polimi.it

ABSTRACT

Nowadays, a huge amount of applications exist that natively adopt a data-streaming model to represent highly dynamic phenomena. A challenging application is constituted by data from the stock market, where the stock prices are naturally modeled as data streams that fluctuate very much and remain meaningful only for short amounts of time. In this paper we present a technique to track *evolving clusters of financial time series*, with the aim of constructing reliable models for this highly dynamic application. In our technique the clustering over a set of time series is iterated over time through sliding windows and, at each iteration, the differences between the current clustering and the previous one are studied to determine those changes that are "significant" with respect to the application. For example, in the financial domain, if a company that has belonged to the same cluster for a certain amount of time moves to another cluster, this may be a signal of a significant change in its economical or financial situation.

1. INTRODUCTION

A data stream is an ordered sequence of data elements that are made available over time, and is potentially unlimited. Nowadays, many applications exist that natively adopt this data model, e.g., all the data produced by sensor readings of any kind, in so many application domains. Very interesting data stream applications can be found in finance, where the life of a given company evolves with time and, at each specific time instant, is characterized by different streams of data reporting its stock price, exchange volumes, balance data, and so on. Companies may even appear and disappear from the market, or change their financial behavior: in fact, the speed and dynamics of the financial market make modeling it a challenging task.

Several techniques for analyzing streaming data have been studied; many of them rely on the use of a sliding window, thus they allow to build a model of the data over a sequence of consecutive time instants, i.e. a window, that moves over time in order to keep into account the newest data. Several clustering methods have been investigated in the financial context to cluster companies that have the same behavior, e.g., in terms of stock-prices. However, especially when stock prices are concerned, clusterings performed on different windows – even though these might overlap greatly – tend to change a lot, due to the high dynamicity of the market and to the scarce permanence of relationships between companies over time. In such a scenario the constructed model tends to fluctuate and remains meaningful only for short amounts of time.

To build reliable and stable models we present PETRA, a *Progressive clustEring TRAcker* where the clustering analysis is iterated over time in order to constantly provide an up-to-date and consistent model. PETRA also focuses on identifying "significant" changes in the clusterings constructed over time. The "significance" of a change is defined by means of parameters. This provides a good deal of flexibility as it also allows the system to be easily adapted to different application domains through an appropriate parameter tuning. When applying PETRA to stock prices, we can capture the intrinsic dynamics of the stocks' (and companies') relationships, through the recognition of points in time where their co-movements change, that is, we identify changes in the behavior of a company relatively to the behavior of the other companies in the same cluster. A company changing cluster is a signal to present to investors as well as a building block for more accurate prediction tasks. For example, well-known companies communicate more than small ones, however, small companies are often more interesting for investors since they can bring a higher profit. After clustering we can use the rich information available for the well-known companies in a cluster to obtain some guidelines also for the small ones inside the same cluster.

PETRA is part of a broader project: Mercurio [3], whose aim is to support financial investors in their decision-making process. In particular, PETRA helps investors understand how companies influence each other in order to guide them towards a deeper understanding of the market structure. The system has been designed and optimized for the financial application domain; however, it can be easily applied to any set of time series with rapidly evolving behaviors, provided that the clustering algorithm, the similarity metrics and some other parameters are appropriately tuned.

1.1 Objective and contributions

Our objective is to dynamically track the evolution of a set of entities $E = (e_1, e_2, ..., e_j, ..., e_N)$ by iteratively clustering their data streams over a sliding window $W_i = [t_{i-w}, t_i]$ of

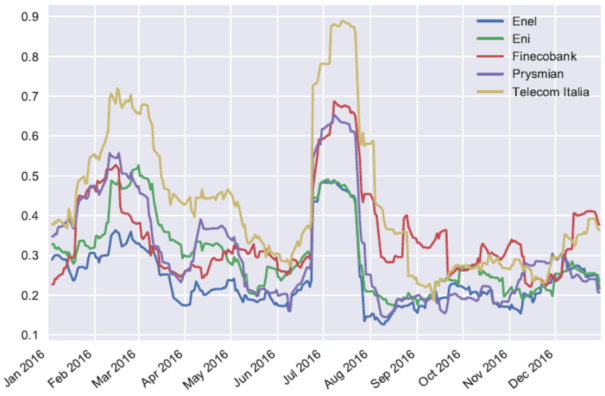

Figure 1: A cluster of similarly-behaving companies

width w (see Figure 3). Given a window W_i, the companies in it are clustered according to their stock price; Figure 1 depicts one such cluster and shows how related companies appear inside the cluster. At each time t_i, the sliding window shifts one time-point ahead, and its content is clustered in K clusters $\mathcal{C}^i = (C_1^i, C_2^i, ..., C_K^i)$. Then, $\mathcal{C}^i$ is compared with $\mathcal{C}^{i-1}$ by associating each new cluster with its "previous version". At the end of the procedure PETRA identifies those entities that have made a "persistent" change of cluster[1]. In particular, our method is aimed at clustering time series that: (i) represent entities belonging to the same domain (e.g. stock-market prices of different companies) and are enough to be grouped into clusters; (ii) are sampled at regular intervals, with the same sampling frequency; (iii) contain a number of time points sufficient to allow a sliding window to move along a significant number of times[2].

Summarizing, our main contributions are:

- the identification of an approach to generate clusters that are well-balanced w.r.t. the problem under consideration;
- the design of a method to trace clusters' behaviors and spot entities' changes of membership;
- the definition of a set of rules needed to identify relevant entity switches.

The rest of the paper is organized as follows: in Section 2 we discuss previous works, Section 3 provides the details about our approach while in Section 4 we show the experimental results and, in Section 5 we draw the conclusions.

2. STATE OF THE ART

Most previous works on clustering stock-market firms showed how custom-made clusters outperform traditional industrial grouping criteria [6, 2]. Our work differs from them in that we move from a static to a progressive clustering approach, which allows us to overcome THE big problem found when dealing with financial information: stock markets move fast, financial insights expire and get old very quickly. Our progressive clustering provides fresh suggestions daily.

Cluster Tracking in evolving data streams is a topic only recently covered by the scientific community. A lot has been done on visually tracking groups of objects [9], e.g., groups of moving people [15] or facial features [12], and on wireless channel modeling to track multi-path components [11]. Our work is different because we are not interested in tracking the shift in position of a cluster at consecutive time instants; rather, we focus on tracking the membership of each object.

General-purpose techniques [5, 4] exist that face the cluster-tracking problem in terms of outliers, i.e. the clustering is considered valid until the number of outliers crosses a predefined threshold, then re-clustering is performed. In PETRA instead, we re-cluster at each iteration since we are not just interested in having a consistent model but also in detecting as soon as possible the entities that change cluster, and promptly report it.

A system similar to PETRA is MONIC [16] that models and monitors cluster transitions. However, PETRA tracks the movements of the companies between clusters in order to generate signals about each company, while MONIC is interested in the transition of the entities between clusters with the scope of drawing conclusion on the lifetime and stability of the clusters. Moreover, MONIC detects the transitions by keeping track of the entity trajectories on spatiotemporal coordinates, while PETRA detects movements by analyzing the contents of the clusters.

3. THE PETRA METHOD

We first discuss our design choices and some domain-dependent parameters that need to be set in PETRA, then we concentrate on the clustering procedure, and finally we show how we track the clusters along time. The description follows the steps of Algorithm 1.

```
K ← desired number of clusters;
i ← 0;
repeat
    CORR = computeCorrelationMatrix(W_i)
    D = computeDistanceMatrix(CORR)
    C^i=clusterData(D, K)
    forall change ∈ trackClusters(C^i, C^{i-1}) do
        if change is relevant then
            trigger warning;
        end
    end
    i ← i + 1;
until new data keeps coming;
```

Algorithm 1: The *PETRA* procedure

3.1 Design choices and parametrizations

Time series clustering algorithms rely on similarity measures for quantifying how homogeneous two observations are. PETRA uses *similarity in time*, which is based on comparing different time series at the same time points, as it is the most suitable one to identify sudden changes in the streams' behaviors.

The similarity measures that can be adopted for clustering time series [1] range from simple ones such as Euclidean distance to more complex ones like Dynamic Time Warping

[1]Note that some changes might be less interesting because being temporary, in the sense that one stream might return to a previous cluster after some time.

[2]Note that the term *significant* is strongly domain-dependent and should be judged by the designer.

(DTW). DTW is extremely powerful, as it finds a relationship between two time series S_1 and S_2 not only when S_2 is obtained by shifting S_1 in time, but also when S_2 is a contraction or an expansion of S_1; e.g. DTW associates the last week of a company with the last month's behavior of another company. However, this feature is not needed in the financial case, where we need to compare companies in the same time interval and at the same time point, thus, simpler approaches, such as the Euclidean metrics and metrics based on correlation, are sufficiently expressive.

As for the clustering method, PETRA employs hierarchical clustering, since it does not require to set the number of clusters a-priori, and at the same time is well suited for easy visualization. Using a Euclidean distance and hierarchical clustering allows us to use Ward's method for linkage [13], which, as we will see, provides a big advantage in the tracking phase. At each iteration, the Pearson Correlation $corr_{ij}$ [10] between each pair of time series i and j is computed and a global similarity matrix $CORR$ is obtained. Then, to obtain a suitable input for a clustering algorithm, a transformation from correlation coefficients to distances is needed. According to Gower and Legendre[8], if $CORR$ is a positive semidefinite similarity matrix, the dissimilarity matrix D where $d_{ij} = \sqrt{(1 - corr_{ij})}$ is a matrix of euclidean distances. Finally, the width w of the sliding window, and the number of desired clusters K, are left as parameters in PETRA since the best choice strictly depends upon the application goal; in Section 4 we present a possible solution to perform an educated guess; we chose a window of 105 days and a number of clusters equal to 4.

3.2 Clustering

At each new iteration, the distance matrix D is computed for the portions of time series inside the window. Then, the hierarchical clustering using Ward's linkage method is performed, a hierarchy of clusters – represented by means of a dendrogram like in Figure 2 – is obtained, and the dendrogram is cut in order to obtain K clusters. Ward's method promotes the merges of small clusters, which is crucial to the tracking phase since well-balanced clusters are easier to track. However, depending on the level of the cut, we obtain different numbers of clusters, and a constant-height cut is not suitable in this case: cutting a dendrogram to obtain a fixed number of clusters irrespectively of its general structure is likely to result in highly unstable clusters. Langfelder at al. [14] solved the problem of sub-optimal performances on complicated dendrograms presenting the Dynamic Tree Cut (DTC) algorithm, which provides a dynamic branch-cutting method for detecting clusters in a dendrogram, depending on their shape. Like in the standard dendrogram creation procedure, clusters are merged in a bottom-up fashion, but this merging depends on a set of parametric criteria such as the minimum size of each cluster and the minimum gap between the joining heights. By using DTC we can obtain much more balanced clusters, both in their size and composition, avoiding the downsides of the standard approach and facilitating the tracking phase. Figure 2 shows an example in which, although both cuts identify three clusters, those obtained with the fixed-height cut are highly imbalanced (i.e. there are two clusters of just one element), while DTC cut delivers very balanced clusters.

3.3 Tracking Clusters

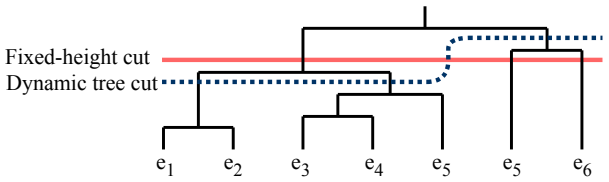

Figure 2: Fixed-height versus Dynamic Tree Cut

The goal of the tracking phase is to observe how clusters evolve and how the clustered streams change their membership. To track cluster dynamics, we have to recognize the same cluster at different times without relying on any external reference. We address this problem by using the Jaccard similarity index [10] and a Linear Programming (LP) formulation. Given two clusterings $\mathcal{C}^i$ and $\mathcal{C}^{i-1}$ obtained from the series over two consecutive windows W_i and W_{i-1}, we compute the Jaccard score of each pair of clusters in the two clusterings. The pairs with highest Jaccard score are the most similar clusters, likely representing the same cluster at consecutive time points. Of course, once a cluster is ascribed to one predecessor it cannot be ascribed to any other one: each cluster needs to be linked to one and only one predecessor. This can easily be formalized in LP. Given two clusterings $\mathcal{C}^i = (C_1^i, C_2^i, ..., C_h^i, ..., C_K^i)$ and $\mathcal{C}^{i-1} = (C_1^{i-1}, C_2^{i-1}, ..., C_j^{i-1}, ..., C_K^{i-1})$, we define the square matrix M of dimension K as the matrix of the Jaccard scores of all the possible pairs (C_h^i, C_j^{i-1}) of clusters. Then we define the square matrix X as a decision variable matrix of dimension K, such that x_{hj} is 1 if $Jac(C_h^i, C_j^{i-1})$ in M is chosen by the optimization process, 0 otherwise. The LP model aims to select from M the K values having highest sum, with the strong constraint that each selected value can share neither the row nor the column with the other selected values. This is equivalent to selecting the K cluster pairs with the highest Jaccard scores, without selecting a cluster more than once. The objective function is defined as $maximize\ M \circ X$ (where $\circ$ represents the element-wise matrix product), subject to the constraints: $\sum_j X_{hj} = 1\ \forall h \in [1, K]$, $\sum_h X_{hj} = 1\ \forall j \in [1, K]$, $X_{hj} \in \mathbb{N}\ \forall h, j \in [1, K]$.

3.4 Discerning relevant changes in clustering

At each new iteration PETRA moves its analysis forward of one time point, performs a new clustering and links each new cluster to its "previous version". By doing so, we detect when an entity moves to another cluster. Three types of change are possible:

- *exit*: a company exits the Reference Cluster, moving into a different cluster;
- *re-entry*: an entity re-enters the Reference Cluster;
- *change outside the reference cluster*: an entity moves between two "non-Reference Clusters".

Note that the Reference Clusters strictly depend on the domain under assessment; in our use case the reference clusters turn out to be the industrial sectors. Reference clusters are re-computed every time we move the time window, and are assigned the industrial sector of the majority of the companies inside it. Thus, a cluster that contains mostly banks will be assigned the label "Bank", one that contains mostly

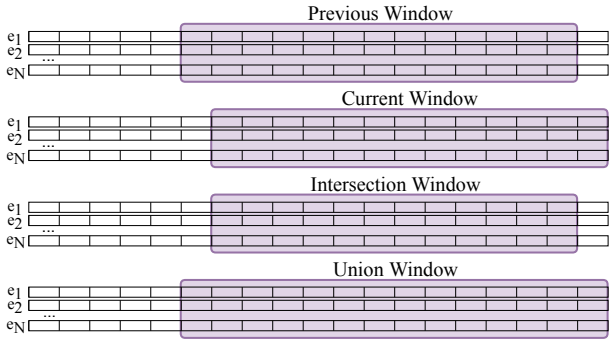

Figure 3: Intersection and Union windows

automotive companies will be assigned "Automotive", and so on. For example, the industrial sector of Unicredit, a famous Italian bank, is "Bank", thus its reference cluster is the one that contains mostly banks. After an iteration, if most of the companies inside Unicredit's cluster are banks, we say that Unicredit is in its reference cluster, and outside its reference cluster otherwise. This allow us to give more importance to those cluster changes that involve the Reference Cluster.

Note that, when using a sliding window, the change in the new configuration might be caused by the new point added or by the one that has been removed. Our goal is to detect changes that are caused by current events, thus, PETRA triggers warnings only when a change is caused by the new point. To do so we use the *intersection window* $W_i^{\cap} = [t_{i-w}, t_{i-1}] = W_i \cap W_{i-1}$, and the *union window* $W_i^{\cup} = [t_{i-w-1}, t_i] = W_i \cup W_{i-1}$ as shown in Figure 3. By computing the clustering also on the intersection window and comparing it with that of W_i we can say, in case they are different, that the new time point t_i has actually influenced the clustering results. Moreover, by computing the clustering also on $W_i^{\cup}$ and comparing it with that of W_i, if no change has happened, we can also say that the removal of the last point t_{i-w-1} has not contributed to the change, meaning that it is entirely due to the new point.

4. EXPERIMENTAL RESULTS

To test and validate PETRA in a real application, we perform a market simulation consisting in the purchase and sale of companies' stocks. This is because, since cluster switches are a latent feature, there is no general ground truth to test against and prove their actual relevance directly. However, as far as the financial domain is concerned, we can use the market gain as an indicator for the goodness of the clustering performed by PETRA.

We consider the stock market prices of the 40 most highly capitalized companies in the Italian Stock Exchange [3] in the period between 2008 and 2017. There are 2 values for each trading day: the opening price (*open*) and the closing price (*close*); we cluster based on the *close-open* indicator. Among the 40 companies we randomly select some of them as *leaders*. Leaders are companies whose behaviour we know in advance (i.e. when their price trend changes): for each of them we identify some companies that are "close to the leader" (*friends*) for which we don't know the future behavior. The simulation consists in buying and selling stocks of the friends depending on the behavior of the leaders. If the clustering is correct, the friends' behavior is similar to the leader's one, hence the simulation will buy and sell the stocks at the right time, resulting overall in a profit.

For the validation, since no ground truth for the trend changes of a company's stock price is available, and since the state-of-the-art techniques for segmenting time series revealed to be not suitable as they do not lend themselves to a unique parametrization suitable for all the companies, we designed an ad-hoc procedure for identifying the major changes in the price trend of a *leader* a-posteriori. The procedure divides the price timeline of a company into intervals of *up-trend*, *down-trend* and *congestion*[4] according to the position of the company's price w.r.t. three moving averages over the price [3].

For each friend, its closeness to the leader is evaluated using PETRA, thus, we first need to appropriately tune the window length and the number of clusters.

The width of the sliding window is a crucial parameter: a too narrow window would result in an excessive sensitivity providing poor generalization potential, while with a too wide one we risk to miss important events. To identify the best width we analyze each possible width value, that is: (i) for each value n between 2 (the smallest possible window) and 1260 (half of the 2008-2017 period) we divide the companies' time series into segments of width n; (ii) for each n, for each time segment T of length n, and for each pair of companies, we compute the correlation between their segments, and arrange them all into a matrix $A_{T,n}$; (iii) we compute the difference between each correlation matrix and the one at the previous time slot; (iv) we compute their variance and then (v) we average all variances together obtaining a single score for each window width n. We chose the value associated with the smallest score corresponding to 105 days. We estimated the number of clusters through knee/elbow analysis, which suggested an optimal number of 4 clusters. After parameter tuning, the simulation picks 3 leaders and then runs PETRA which, at each iteration, adopts a conservative policy:

- If a leader enters an up-trend phase, pick the 4 closest companies in the same cluster as friends and buy them.
- If a leader enters a down-trend phase, sell all its friends.
- If a friend changes cluster, sell it.

To constitute valuable evidence we run the simulation ten thousand times over different portfolios of leaders and then averaged the profits. Comparing the results to a stock market index is the common practice when it comes to market simulation, thus, we use the FTSE MIB[3] index as a baseline; Figure 4 reports the aggregated results for each year by showing the average profits obtained with PETRA and the FTSE MIB profits for the same year. Note how the profits yielded by our procedure are always positive: even in years like 2008, when the market underwent a severe crisis, our procedure managed to generate profits. Although there are certain years in which FTSE MIB outperforms our approach, our results remains still very comparable, especially when considered on the long term.

[3]https://en.wikipedia.org/wiki/FTSE_MIB

[4]A period in which the price undergoes oscillations without a clear direction.

According to [7], the traditional approach when validating Stock Market Clustering is to check if the resulting clusters are consistent w.r.t. the business sectors. Being the industrial sectors static entities, we decided not to use this approach for the validation since the aim of our procedure is to dynamically spot anomalies and changes. Nevertheless our clusters have proved to be consistent also under this point of view. The four clusters identified by our procedure coincide with the sectors: *banking, energy, consumer products* and *industrial products.*

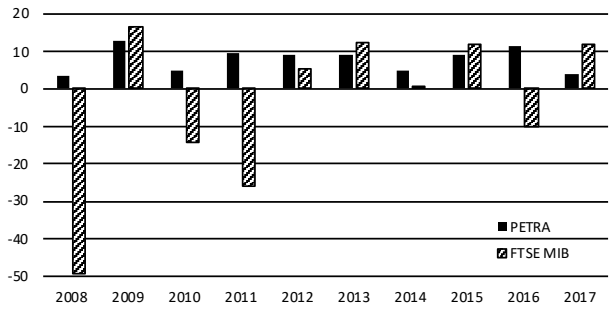

Figure 4: Average profit (in %) for each year between 2008 and 2017

5. CONCLUSIONS

We presented a technique to cluster time series where the clustering is iterated over time and, at each iteration, the differences between the current clustering and the previous one are studied in order determine significant changes. We illustrated the technique on an example in the financial domain, where companies are clustered using the time series of their stocks, and cluster changes are considered as signals of changes of the company's economical or financial situation. By observing the way clusters evolve in time, we see data streams (and the corresponding companies) moving from one cluster to the other one: this results in an efficient way of spotting anomalies and out-of-the-ordinary events that could potentially carry a high predictive power.

Acknowledgments

Mirjana Mazuran is supported by the H2020 research program under grant agreement 800192. We are grateful to U. P. De Vos and to L. Raimondi for taking part in the development of the algorithms and the analysis of the results.

6. REFERENCES

[1] S. Aghabozorgi, A. S. Shirkhorshidi, and Y. W. Teh. Time-series clustering - A decade review. *Information Systems*, 53:16–38, 2015.

[2] S. Aghabozorgi and Y. W. Teh. Stock market co-movement assessment using a three-phase clustering method. *Expert Syst. Appl.*, 41(4), Mar. 2014.

[3] D. Azzalini, F. Azzalini, D. Greco, M. Mazuran, and L. Tanca. Event recognition strategies applied in the mercurio project. In *MIDAS*, pages 29–30, 2017.

[4] D. Barbard and P. Chen. Tracking clusters in evolving data sets. *Data science and Engineering*, 2(3):210–223, 2001.

[5] B. Daniel. Requirements for clustering data streams. *ACM sIGKDD Explorations Newsletter*, 3(2):23–27, 2002.

[6] E. J. Elton and M. J. Gruber. Improved forecasting through the design of homogeneous groups. *The Journal of Business*, 44(4):432–450, 1971.

[7] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market (extended abstract): Which measure is best? pages 487–496, 2000.

[8] J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. 3:5–48, 02 1986.

[9] K. Granstrom, M. Baum, and S. Reuter. Extended object tracking: Introduction, overview and applications. *Journal of Advances in Information Fusion*, 12(2):139–174, 2016.

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

[11] C. Huang, R. He, Z. Zhong, B. Ai, and Z. Zhong. Comparison of automatic tracking and clustering algorithms for time-variant multipath components. In *Globecom Workshops*, pages 1–6, 2017.

[12] N. Islam, M. Seera, and C. K. Loo. A robust incremental clustering-based facial feature tracking. *Applied Soft Computing*, 53:34–44, 2017.

[13] J. H. W. Jr. Hierarchical grouping to optimize an objective function. 58:236–244, 03 1963.

[14] P. Langfelder, B. Zhang, and S. Horvath. Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. *Bioinformatics*, 24(5), 2008.

[15] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer vision and image understanding*, 80(1):42–56, 2000.

[16] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. Monic: modeling and monitoring cluster transitions. In *ECML/PKDD*, pages 706–711. ACM, 2006.