Where Are We Going? Predicting the Evolution of Individuals

Zaigham Faraz Siddiqui[§], Márcia Oliveira[†], João Gama[†], and Myra Spiliopoulou[§]

[§]University of Magdeburg, Germany
[†] LIAAD - INESC TEC and University of Porto, Portugal
siddiqui@iti.cs.uni-magdeburg.de; marcia@liaad.up.pt; jgama@fep.up.pt; myra@iti.cs.uni-magdeburg.de

Abstract. When searching for patterns on data streams, we come across perennial (dynamic) objects that evolve over time. These objects are encountered repeatedly and each time with different definition and values. Examples are (a) companies registered at stock exchange and reporting their progress at the end of each year, and (b) students whose performance is evaluated at the end of each semester. On such data, domain experts also pose questions on how the individual objects will evolve: would it be beneficial to invest in a given company, given *both* the company's individual performance thus far and the drift experienced in the model? Or, how will a given student perform next year, given the performance variations observed thus far? While there is much research on how models evolve/change over time [Ntoutsi et al., 2011a], little is done to predict the change of individual objects *when the states are not known a priori*. In this work, we propose a framework that learns the clusters to which the objects belong at each moment, uses them as *ad hoc states* in a state-transition graph, and then learns a mixture model of Markov Chains, which predicts the next most likely state/cluster per object. We report on our evaluation on synthetic and real datasets.

Keywords clustering, cluster transition, data streams, evolutionary data mining, label prediction, perennial objects

1 Introduction

Stream mining is a mature research field, encompassing algorithms that learn a model from observed data, and adapt it as the data generating distribution changes. However, many real-world applications are not solely interested in capturing model change but also in understanding how individual *objects* evolve. Consider for example patients with a chronic disease or customers of a company. The treating physician, or the marketing department, is interested in understanding how each individual changes in response to observable measures (e.g., medication and marketing actions, respectively) and due to unobservable factors

(health deterioration caused by the disease and changes in the customer's personal preferences, respectively). In view of the current economic downturn, the ability to accurately predict corporate bankruptcy is also of utmost importance and one of the steps to avoid it. By analysing the information of the registered companies, at the end of each accounting period, it is possible to derive distinct company profiles, and generate probabilistic estimates of their evolution. Thus allowing to predict the likelihood that they will survive or go bankrupt. We propose a framework that captures the evolution of objects over time, derives 'ad hoc' states (as opposed to the *a priori* defined states of a Markov model), and predicts the transitions of objects from one state to another.

Intuitively, the individuals, whose evolution is to be predicted, can be modelled as *relational* objects: a patient is an entity accommodating personal information and is linked to further instances - medical tests, diagnoses and treatments. These instances constitute a stream, while the individuals themselves are *perennial*, in the sense that they are permanently stored and (ir)regularly enriched with stream instances that reference them. Relational stream mining encompasses methods that learn and adapt a model for such objects [1–3]. However, these methods do not predict the next state/cluster of an object.

There are many methods that study how clusters evolve in a concrete application, e.g. in the context of community evolution. However, as elaborated in the extensive literature discussion of [4], works on *modelling and monitoring* the phenomenon of cluster evolution are very sparse and essentially point back to the early framework MONIC [5]. MONIC encompasses a set of 'transitions' that a cluster may experience (split into many clusters, merge with other clusters, survival and disappearance), a set of measures and a cluster comparison mechanism using them to assess whether a cluster observed at some timepoint has survived, disappeared, merged or become split at the next timepoint. The frameworks MEC [4] and FINGERPRINT [6] build upon MONIC to explain evolution: they both model the clusters and their transitions as nodes, respectively directed edges, in an *evolution* graph, which they use to explain how the underlying population evolves. FINGERPRINT [6] summarizes paths on this graph to build a more concise representation of evolution. MEC [4] observes the evolution graph as a state-transition graph and applies Hidden Markov Models to explain the evolution of the underlying dynamic environment, and has been used experimentally to explain the evolution of Portuguese economic activity sectors on the basis of a recent dataset (cf. [4] for details). In this study, we use the concepts of MONIC [5] and extend MEC [4] by deriving a minimal number of Markov Chains that can explain the observed evolution graph.

Hidden Markov Models are also used by Laxman et al. in [7], the goal of which is to predict target event types over a categorical sequence stream of historical events. Laxman et al. first identify significant frequent episodes of event sequences in a time window, and then estimate a mixture of Hidden Markov Models (HMM), each one associated to a distinct episode. The estimated model is then used to predict future occurrences of a given target event type. However,

they work in the context of supervised learning. We rather aim to identify Markov Chains that explain the unlabelled data.

There are also studies concentrating on how individual objects evolve over time. Gaffney and Smith [8] model the evolution of an object as a trajectory and cluster together objects that evolve similarly. Krempl et al. [9] extended [8] into the online algorithm TRACER that discovers and adapts the clusters as new observations of existing objects arrive and new objects appear. However, these algorithms concentrate on monitoring objects that evolve similarly, while our approach, similarly to MEC [4] and FINGERPRINT [6] concentrate on deriving models that explain the transitions of the clusters themselves.

We propose a framework that incrementally (a) derives and (b) adapts the states of perennial objects, and (c) predicts a perennial object's next state, whereupon the states are clusters that are learned and adapted. at each timepoint. We then build a transition graph for the perennials, where a perennial experiences a 'transition' whenever it moves from one cluster to another. We use this transition graph to find the optimal number of Markov Chains that can predict the next state of each object, given the states learned thus far.

The paper is organized as follows. In the next section we formalize the problem of predicting the next state of a perennial object, when the states are themselves learned. Thereafter, we present our framework for learning the states and then learning a state transition model over them. We then report on our experiments in a synthetic data set with complex forms of drift and in a real dataset. The last section concludes our study.

2 Problem Description

Assume a stream of perennial objects that contains an infinite sequence of objects $\mathcal{O} = \langle x_1, \ldots, x_j, \ldots \rangle$ and a discrete sequence of timepoints $\langle t_1, \ldots, t_i, \ldots \rangle$. The objects from $\mathcal{O}$ are encountered at regular intervals with a different set of values, which is consistent with their most recent state. The objects arrive as a stream of objects $\mathcal{X} = \langle x_1^{t_1}, x_2^{t_1}, \ldots, x_1^{t_i}, x_2^{t_i}, \ldots \rangle$ with increasing number of timepoint t_i. For any two occurrences of $x_j^{t_i}$ and $x_j^{t_k}$ of an object x_j, $x_j^{t_i} \neq x_j^{t_k}$, if $t_i \neq t_k$. For multiple instances of x_j, the one with the higher t index represents the most recent instance of x_j, i.e., if $t_i > t_k$, then $x_j^{t_i}$ is its most recent instance.

Being perennial, the objects are dynamic. We investigate how these perennial objects *evolve* over time. Formally, for a given instance $x_j^{t_i}$ of an object x_j at timepoint t_i, what is the most probable instance $\hat{x}_j^{t_{i+1}}$, for x_j, at timepoint t_{i+1}? For the case where x_j has $d = 1$ dimension only, this is a single-value prediction. When $d > 1$ though, the prediction task becomes multi-valued and is not trivial (see left of Figure 1).

We reduce the problem to a single-valued prediction task by utilising the clustering structure over the data. In right of Figure 1, we show a clustering model ζ that is learned over the individual instances of the objects for a given time horizon. Each centroid/cluster in the model represents a possible state which an object can inhabit. For example, if centroid $C3 \in \zeta$ is closest to the

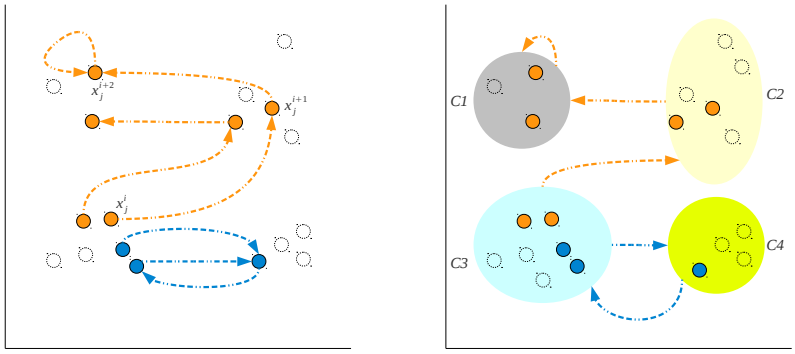

Fig. 1. (left) The original problem reduced to **(right)** a single-value prediction task. Different transition profiles are represented in different colours. (In the left sub-figure we mark the timepoint information for object x_j only. The rest can be interpreted on the basis of transitions.)

instance x_j^i of an object x_j at t_i, then x_j is said to be in state $c_j^i = C3$ at t_i. Therefore our question becomes, for a given object x and its state c^i at t_i, what is the most likely state $\hat{c}^{i+1}$ that x would acquire at t_{i+1}.

Our prediction task is of evolutionary nature and the transition profiles are defined over in terms of how objects change the cluster membership over time. This means that a cluster found at some timepoint serves as an 'ad hoc' learned state, so that an object moving from one cluster to another experiences a *state transition*. In clustering, there can be many different clusters that describe the objects. In the context of our framework, this means that there can be more than one type of evolutionary patterns on the objects, depending on the clusters.

Our method builds up on the MEC framework [4]. As discussed in Section 1, MEC traces evolution through the detection and categorisation of clusters transitions, such as *births*, *deaths*, *splits* and *merges*. The clusters are represented by enumeration (also known as *extensional definition of clusters*). In this kind of representation a cluster is defined by its members, i.e., by the observations that were assigned to it by a given clustering algorithm. It takes as input a set of clusterings, each one generated at a different timepoint. It performs pairwise mappings, between clusters obtained at timepoint t_i and at timepoint t_{i+1}. The *mapping* process explores the concept of conditional probability and is restricted by a user-defined threshold - the *survival threshold* τ, where $\tau \in [0.5, 1]$.

As in MEC, we trace evolution through the detection and categorisation of cluster transitions. However, we do not use the extensional definition of clusters: rather, the evolution of the objects results to new clusters, changing the nature of clusters and the subsequent transitions. We thus observe multiple evolutionary patterns in the data, not a single one as MEC. Hence, we aim to identify the optimal number of Markov Chains that explain the observed evolution.

ID_{old}	Timepoint	Values		ID_{new}	Values
j	i	$\text{tuple}(x_1^{t_1})$	$\rightarrow$	$j \oplus i$	$\text{tuple}(x_j^{t_1})$
j	$i+1$	$\text{tuple}(x_1^{t_{i+1}})$	$\rightarrow$	$j \oplus i+1$	$\text{tuple}(x_j^{t_{i+1}})$
$\ldots$	$\ldots$	$\ldots$	$\rightarrow$	$\ldots$	$\ldots$
j	k	$\text{tuple}(x_1^{t_k})$	$\rightarrow$	$j \oplus k$	$\text{tuple}(x_j^{t_k})$

Fig. 2. Flattened out timepoint information

3 Framework

Our proposed framework for mining and predicting over a stream of perennial objects consists of two components. First component comprises of a clustering algorithm that operates over stream with a *flattened* time horizon[1] and finds clusters over the individual instances of the objects. All clusters that exist within the given time horizon are discovered by the algorithm. Second component takes as input the clustering resul. It learns a mixture of Markov Chains based on how objects change their cluster membership over time. Using this mixture of Markov Chains, we predict what is the next most likely cluster/ state an object is going to be in. We next describe our approach in detail.

3.1 Clustering Objects with Flattened Time-Horizon

First component of our framework operates over all the instances (including past instances) of an object and separates them into groups of similar instances, performing the steps described below.

Time Flattening Before the clustering can be performed, we first flatten out the timepoint information. This means that multiple instances $x_j^{t_i}, x_j^{t_{i+1}}, \ldots$ of an object x_j are initially treated as distinct objects: we create their identifiers by concatenating the timepoint information and the identifier of x_j. Timepoints are not exploited for the individual instances-turned-objects, as we illustrate in Figure 2. The timepoint information is not discarded, though: we later use it when we re-construct the original objects and their evolution as sequence of cluster membership transitions.

Constructing Cluster Transition Sequences Once the instances of the objects are flattened out, the instances are fed to a clustering algorithm[2]. In Figure 3, we depict the flattened instances and the clusters learned over them. It is important to note that the clusters span across the whole time horizon. There might be some clusters that contain instances that belong to only a subset of timepoints, while others contain instances from the complete set of timepoints.

[1] We describe how we flatten the time horizon later in the section.

[2] We have experimented with various clustering algorithms and have found EM clustering [10] to work best with the datasets that are available to us. The parameters for the clustering, i.e., K (#clusters) can be discovered by experimentation.

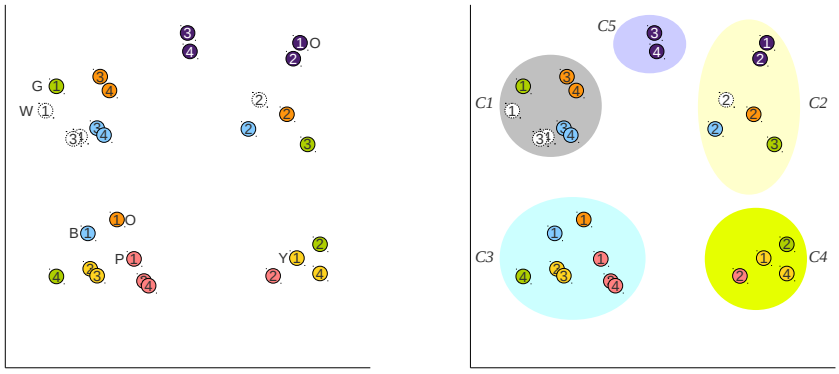

Fig. 3. (left) Flattenned instances, **(right)** Clustering over instances.

Before the transition model can be learned, the results are first converted into sequences. First, the concatenated identifiers are un-winded and the object and timepoint data are linked to the individual instances. These data are then used to construct sequences for each individual object. The sequence represents how the individual instances of an object underwent a change, how the object changed clusters and evolved over time. For a sequence seq_x of object x, the i-th element of the sequence holds the cluster id for the instance x^i at t_i. In the left of the Figure 4 we depict the constructed sequences.

3.2 Learning the Cluster Transition Model for Perennial Objects

From the constructed sequences, the cluster memberships for an object x can be determined for each timepoint. For two consecutive timepoints t_i and t_{i+1}, the object x is said to make a transition from cluster c^i at t_i to c^{i+1} at t_{i+1}. However, there can be more than one type of transition profile exhibited by the objects. Before we can learn a Markov Chain, we separate the sequences for the objects into groups. Each group of sequences serves as a seed and a separate Markov Chain is learned over it.

In order to separate the sequences into groups we used Levenshtein distance [11]. Unlike other measures (i.e., Euclidean distance and cosine similarity) the groups are not prototype-based. First, all the unique sequences are identified and the $benefit()$ is computed for each sequence whether it can serve as a good seed or not. The formula for benefit is given in the Equation 1.

$$benefit(s) = \sum_{s' \in seqs} \left\{ \begin{array}{ll} dist_{max} - d & d < dist_{max} \\ 0 & d >= dist_{max} \end{array} \right\} \tag{1}$$

where, $d = distance(s, s')$ and $seqs$ are the constructed sequences for the objects.

In order to select the seeds for the mixture model, we use following heuristic. We sort the sequences on the $benefit()$ value and compute the average benefit

ID	Sequence	ID	Sequence
B	3211	U	2255
O	3211	X	3211
M	1211	Y	3211
P	3433	Z	4334
Y	4334		
G	1423		
V	2255		

Unique	Contributors	Benefit
3211	3211:4, 1211:1	9
~~1211~~	1211:1, 3211:4	6
4334	4334:2, 3433:1	5
~~3433~~	3433:1, 4334:2	4
2255	2255:2	4
~~1423~~	1423:1	2
#seeds=3	$\frac{b_{avg}}{2}$=3	b_{avg}=6

Fig. 4. (left) Objects reconstructed as sequences of clusters they belong to and **(right)** sequence sorted according to the benefit values. The benefit is computed with $dist_{max} = 2$. Sequences with $benefit() > \frac{b_{avg}}{2} = 3$, serve as potential seeds. Sequences that have already contributed to a Markov Chain, e.g. `1211`, get removed from the potential seeds.

b_{avg}. We mark all sequences with $benefit() > \frac{b_{avg}}{2}$ as potential seeds. Starting with the first sequence s with the highest benefit, we train a Markov Chain over all sequences s' such that $distance(s, s') < dist_{max}$. In other words, if a sequence s' contributed to the benefit value for s, it is incorporated into the Markov Chain for seed s. Once a sequence s' contributes to some Markov Chain of sequence s, it is unmarked as potential seed and we do not learn a separate Markov Chain for it. We keep learning the Markov Chain until there are no more potential seeds left or until we have learned k_{MC} Markov Chains[3]. The process of sequence separation is shown in right of Figure 4 for the running example. At the end of this process, some sequences might have been left, which do not contribute to any Markov Chain (e.g., seq "`1423`"): they are assigned to the Markov Chain with the highest log-likelihood.

4 Evaluation

We have evaluated our framework on a real and a synthetic datasets described below. The objective of the evaluation was to learn the transition profiles with the least number of Markov Chains, while observing least number of examples.

4.1 Data Description

European Companies (EC) Dataset To evaluate our framework in a real world scenario, we extracted publicly available dataset[4], comprising financial and accounting information for a set of European individual companies, for the time horizon that goes from 2003 to 2007. After a pre-processing stage, which involved the removal of missing values, duplicates and outliers, we selected only the subset of companies with information reported for all timepoints under analysis. We obtained a set comprised of 836 distinct companies, each one characterized

[3] k_{MC} parameter explicitly sets the number of Markov Chains in the mixture. However, setting it is not mandatory, it is an *optional* parameter.

[4] `http://pages.stern.nyu.edu/~adamodar/New_Home_Page/data.html`

by 7 continuous variables. We considered this dataset appropriate to test our approach due to two main reasons:

1. Financial and accounting information resulting from companies activity is dynamic since it is collected at regular timepoints, usually on a yearly basis.
2. One of the basic assumptions in Management is the *principle of continuity* that claims that a company's goal should be to develop its activity continuously and with unlimited duration, avoiding the interruption or cessation of its activity. Therefore, it is highly likely that over time any profit-driven organization will go through a series of different stages during its life cycle.

Synthetic Dataset The data generator[5] takes as input the number of clusters and number of transition profiles. It first initialises Gaussian distributions for each cluster. Transition profiles are defined in terms of how objects jump from one Gaussian cluster to another. The transitions are represented as a matrix and are generated in a semi-controlled way: each object adheres to certain transition profile. T initial cluster for each object is chosen randomly; a transition matrix is used to determine subsequent clusters. Individual instances for each object are generated depending on the cluster the object was in at t_i.

4.2 Evaluation Settings

Baseline We compare our strategies for the real dataset against a baseline algorithm we call MEC+; it is built upon MEC [4]: it uses the graph generated by MEC and learns a single Markov Chain, which we use to predict the next state of the perennial objects.

As pointed out in Section 3.1, we have used various clustering algorithms: K-Means, EM with Gaussian Mixtures and DBSCAN[6]. To find out the number of clusters K, we tried silhouette coefficient at distinct timepoints. For the complete flattened-out data, the separation was low. Hence, not only was it difficult to find an appropriate value of K, but the quality of K-Means and DBSCAN was also affected. Since our objective is not to build a "crisp clustering" but to discover regions in which objects move, and since EM can find overlapping clusters (which we verified through experimentation as well), we chose it as the base clustering algorithm and vary K according to values of silhouette at individual timepoints.

To compute the performance of the learned models, we initially used two measures: *predictive accuracy* and *perplexity*. Accuracy tell us whether the predicted states match the actual states. Thus, models yielding higher accuracies are assumed to have better performance. However, accuracy does not take into account neither the indeterministic nature of the system nor the probabilistic nature of the outcome of a model. To incorporate this information in our evaluation we consider perplexity, which is a widely used measure to evaluate the generalization performance of a probabilistic model. Intuitively, we can understand perplexity as the degree of surprise of a trained model to unseen data.

[5] `http://omen.cs.uni-magdeburg.de/itikmd/software/index.html`
[6] For DBSCAN we mostly get a big cluster and the remaining is noise

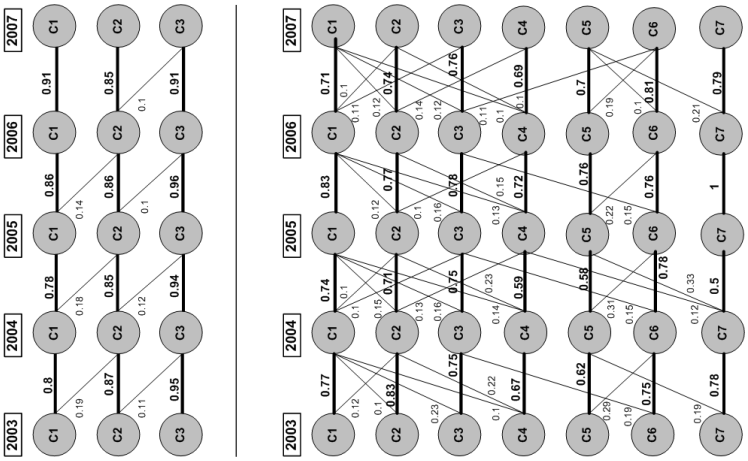

Fig. 5. Bipartite graphs for the European individual companies data, corresponding to time span $[2003, 2007]$. **(Left)** Setting the number of clusters to $k_t = 3$, $\forall_{t\in[2003,2007]}$. **(Right)** Setting the number of clusters to $k_t = 7$, $\forall_{t\in[2003,2007]}$.

Therefore, better probabilistic models tend to offer smaller perplexities on test data. When Markov Chains are used for prediction, perplexity measures how well the model fits the underlying unknown process distribution being represented by independent test samples. We use the following formula to compute perplexity:

$$Perplexity(M) = b^{-\frac{1}{N}\sum_{i=1}^{N} log_b P(x_i|M)} \tag{2}$$

where x_i is a test example drawn from the test sample $x_1, x_2, ..., x_N$, with size N, b is the base of the logarithm, given by the number of states in the distribution, and $P(x_i|M)$ is the probability of the most likely outcome for a test example x_i $(i = 1, ..., N)$, provided by the probabilistic model M.

4.3 Experimental Results

Setting the K is an unsolved problem in general. For the concrete scenario, we experiment with different values of K and study how homogeneity increases with the number of clusters. First we show the clustering results over EC dataset along with the transitions (see Figure 5). For a low value of $K = 3$, the clusters discovered are abstract. For most objects there were not a lot of inter cluster transitions. It is apparent that for such a case, the baseline can approximate the transitions well by using only one Markov Chain. While for $K = 7$, algorithm discovered clusters at a higher resolution. The algorithm was able to discover local transitions which otherwise were obscured at $K = 3$. Because of this we would expect our algorithm to show better performance than baseline at higher values of K. For lower values the performance is expected to be competitive.

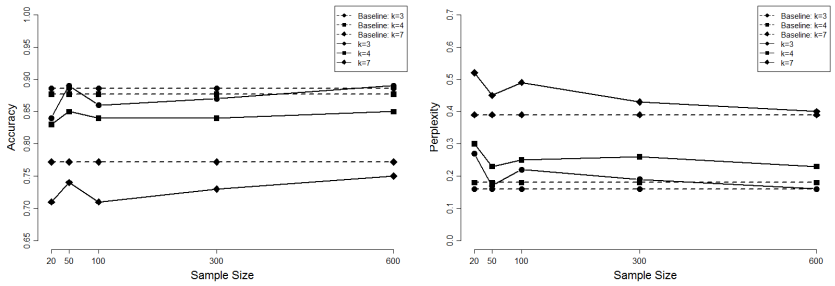

Fig. 6. (left) Accuracy and (right) Perplexity plots for the EC dataset. The mixture model learned for years 2003-2006 with varying sample size for training. The mixture model was used for predicting the clusters for test data for the year 2006. The dotted line represents the baseline MEC for which $k_{MC} = 1$.

In Figure 6 we show the performance for the first set of experiments. The lines depict the different strategies with different parameters, namely the number of clusters $K = 3, 4, 7$ found in the initial clustering (cf. Section 3.1). The strategies were allowed to determine the parameter k_{MC}. Each strategy was compared against MEC+, which was learned at the maximum sample of 600, i.e., number of training examples provided. Varying the sample size has very little effect on the strategies in this set. We see that for each value of K, the baseline ($k_{MC} = 1$) shows better performance than strategies with more than one Markov Chain. This indicates that a single Markov Chain suffices to explain the data.

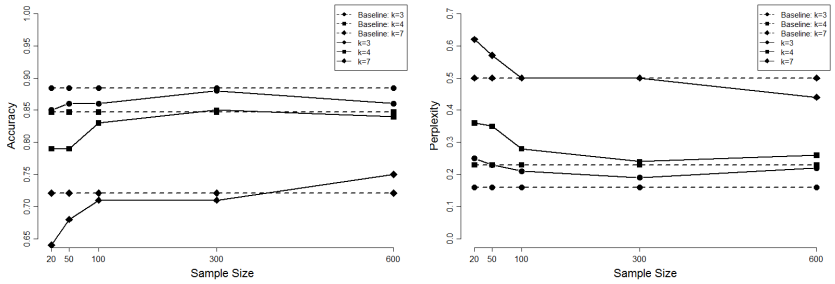

Fig. 7. (left) Accuracy and (right) Perplexity plots for the EC dataset. The mixture model learned for years 2003-2006 with varying sample size for training. The model was used for predicting the clusters for test data for the year 2007, i.e., future.

In Figure 7 we show the results for $K = 3, 4, 7$ clusters. The model was trained for a period from 2003-2006 and the predictions were done for 2007. We see that

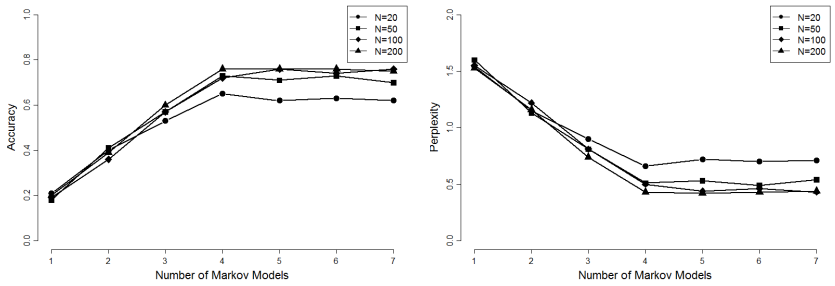

Fig. 8. (left) Accuracy and (right) Perplexity plots for synthetic dataset. The measures are plotted against the number of Markov Chains used in the mixture.

the performance of strategies with a smaller K is better and their baselines also show better performances. This is partially expected, as there are less cluster candidates for a transition, so the probability for each one is higher. For $K = 7$, baseline is inferior to the strategy that use more than one Markov Chain, i.e., $k_{MC} > 1$. This is consistent with the earlier discussion, where clustering with $K = 7$ showed complex transitions.

In Figure 8 we show the results on a synthetic dataset with 4 transition profiles and 300 objects. In this set of experiments our strategies used different sample sizes for training. In this experiment, our baseline is not explicitly visible: it can be found at $k_{MC} = 1$. For low k_{MC}, the mixture model was unable to approximate the different transition profiles; for $k_{MC} = 1$ we record the worst performance. As k_{MC} increases, the accuracy of the prediction improves until the right value for k_{MC} is reached. The strategy with the smallest sample size shows the worst performance, since the small fraction of the data makes it difficult to find the appropriate seeds for learning the Markov Chains. In contrast, if right number of transition profiles is not known in advance, sample sizes hardly have any effect on the performance.

5 Conclusion

In the paper we have presented a framework for predicting the evolution of perennial objects. We have used clustering over the timestamped data to derive *states* of the objects, then performed cluster matching and derived *transitions* as migrations from one cluster to another. On this basis, we developed methods that identify different types of transition profiles and learn a mixture of Markov Chains to predict transitions, whereby the objective is to identify the optimal number of MCs needed to describe the transitions. We have presented the first results from the framework on one real and one synthetic dataset. In the experiments, we have found that the number of clusters K influences the number of MCs needed. So, we intend to investigate methods for fixing K by monitoring

abrupt changes in cluster homogeneity, and also to use clustering methods that do not require a fixed number of clusters as input.

In this study, we have concentrated on learning ad hoc states with clustering. A future step is the extension of the framework for the supervised case. In particular, we are interested in tackling the issue of label change: the label of a perennial object can change [2], similarly to any other variable. Predicting label change would involve a more elaborate definition of state, and replacing Markov Chains with Hidden Markov Models.

Acknowledgments First author was funded by the German Research Foundation project SP 572/11-1 "IMPRINT: Incremental Mining for Perennial Objects". Second author was funded by FCT, under the PhD grant SFRH/BD/81339/2011, by the ERDF through the COMPETE Programme, and by National Funds through FCT within the project FCOMP-01-0124-FEDER-022701. This work is also funded by FCT through the project KDUDS - Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIAEIA/098355/2008).

References

1. Z. F. Siddiqui and M. Spiliopoulou. Combining multiple interrelated streams for incremental clustering. In *21st Int. Conf. on Scientific and Statistical Database Management, SSDBM 2009*, pages 535–552. Springer, 2009.
2. Z. F. Siddiqui and M. Spiliopoulou. Tree induction over perennial objects. In *22nd Int. Conf. on Scientific and Statistical Database Management, SSDBM 2010*, pages 640–657. Springer, 2010.
3. E. Ikonomovska, K. Driessens, S. Dzeroski, and J. Gama. Adaptive windowing for online learning from multiple inter-related data streams. In *Workshop on Learning and Data Mining for Robots (LEMIR'11@ICDM'11)*, pages 697–704, Dec. 2011.
4. M. Oliveira and J. Gama. A framework to monitor clusters evolution applied to economy and finance problems. *Intelligent Data Analysis*, 16(1):93–111, 2012.
5. M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. MONIC – modeling and monitoring cluster transitions. In *12th Int. Conf. on Knowledge Discovery and Data Mining*, pages 706–711. ACM, 2006.
6. I. Ntoutsi, M. Spiliopoulou, and Y. Theodoridis. Summarizing cluster evolution in dynamic environments. In *Int. Conf. on Computational Science and Its Applications, ICCSA 2011*, pages 562–577, 2011.
7. S. Laxman, V. Tankasali, and R. W. White. Stream prediction using a generative model based on frequent episodes in event sequences. In *Procs of the 14th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 453–461. ACM, 2008.
8. S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *5th Int. Conf. on Knowledge Discovery and Data Mining*, pages 63–72, 1999.
9. G. Krempl, Z. F. Siddiqui, and M. Spiliopoulou. Online clustering of high-dimensional trajectories under concept drift. In *ECML PKDD*. Springer, 2011.
10. 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*, 39(1):1–38, 1977. Series B (Methodological).
11. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(1):707–710, 1966.