Tracking Recurrent Concepts Using Context in Memory-constrained Devices

João Bártolo Gomes*, Ernestina Menasalvas* and Pedro A. C. Sousa †
*Facultad de Informática, Universidad Politécnica de Madrid, Spain
joao.bartolo.gomes@alumnos.upm.es, emenasalvas@fi.upm.es
† Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Lisboa, Portugal
pas@fct.unl.pt

***Abstract*—The dissemination of ubiquitous devices with data analysis capabilities motivates the need for resource-aware approaches able to learn in reoccurring concept scenarios with memory constraints. The majority of the existing approaches exploit recurrence by keeping in memory previously learned models, thus avoiding relearning a previously seen concept when it reappears. In real situations where memory is limited it is not possible to keep every learned model in memory, and some decision criteria to discard such models must be defined. In this work, we propose a memory-aware method that associates context information with stored decision models. We establish several metrics to define the utility of such models. Those metrics are used in a function that decides which model to discard in situations of memory scarcity, enabling memory-awareness into the learning process. The preliminary results demonstrate the feasibility of the proposed approach for data stream classification problems where concepts reappear and memory constraints exist.**

***Keywords*-Ubiquitous Knowledge Discovery, Data Stream Mining, Concept Drift, Recurring Concepts, Context-awareness, Resource-awareness**

I. INTRODUCTION

Learning from data streams in ubiquitous devices where the data distributions and target concepts may change over time is a challenging problem, known as concept drift [13]. In real world classification problems it is common for previously seen concepts to reappear [14]. This represents a particular type of concept drift [13], known as concept recurrence [1], [5], [10], [14], [15].

Prediction models usually change over time, for example in product recommendations where customer interests change due to fashion, economy or other *hidden context* [8], [14]. Several methods have been proposed to detect and adapt to concept drift [6], [13]. The usual approach is to use a forgetting mechanism and learn a new decision model when drift is detected [6]. A possible solution to exploit recurrence, is to store previously learned models that represent observed concepts, thus avoiding relearning a previously learned concept when it reappears [1], [5], [15]. The main drawback of this approach is associated with its memory consumption cost. Depending on the capabilities of the device where the learning algorithm is executed, this can be a critical factor. Thus resource-awareness [2], [3] should be considered in memory constrained scenarios. Examples of ubiquitous applications that use resource-constrained devices include intelligent vehicles, personal digital assistants (PDA), wireless sensor networks (WSN) or ambient intelligence systems to name a few.

Resource-awareness means monitoring the availability of the required resources and adapt accordingly. This has been addressed in [3] where a generic framework that uses the mining algorithm granularity settings in order to change the resource consumption patterns according to availability of different resources. For example the parameter that controls the number of clusters is defined as a function of the available memory.

To address the problem of concept recurrence in resource-constrained devices, we propose a method that adapts its behaviour according to available memory. It extends existing drift detection methods [6] by associating context information with learned decision models, under the assumption that recurring concepts are related with context [1]. Such context information is used to improve the adaptation of the learning process to drift. The method used to decide which models to discard according to memory constraints constitutes the new contribution of this paper. This method uses pre-defined criteria to assess model utility in order to discard low utility models, allowing the storage of new ones in situations of memory scarcity.

This paper is organized as follows. Section II describes the problem of concept recurrence with memory constraints, with its assumptions and requirements, followed in Section III by the proposed resource-aware solution. In Section IV the preliminary experimental results obtained are presented and discussed. Finally we provide some concluding remarks and outline future research work in Section V.

II. RECURRING CONCEPTS LEARNING PROCESS

Let us assume a system learning from a stream of records where concepts can change over time, based on an incremental version of the Naive Bayes algorithm [4]. Note that any other incremental classification algorithm can be used instead without loss of generality. The performance of the learning algorithm is monitored using a drift detection mechanism [6] that triggers an event when drift is detected.

 ISBN: 978-1-61208-100-7

To make this system resource-aware and able to handle recurring concepts, the following requirements must be met:

- **i)** adapt the classification model to concept drift.
- **ii)** recognize and use past models from previously seen concepts when these reappear.
- **iii)** use contextual information to improve adaptation to drift.
- **iv)** adapt memory consumption according to availability of resources.

We assume that:

- the target concepts are related to available context. This is the main motivation behind storing models together with context.
- the memory available in the device running the learning process can be obtained anytime through a known interface.

Drift Detection method - When the records distribution is stationary the classifier error-rate decreases as the number of training records grows. This assumption is shared by most approaches dealing with drift [13], as it is a natural property in real world problems where periods of stable concepts are observed followed by change to a new period of stability (with a different target concept). Proposed in [6] a method for drift detection uses this assumption to find drift events. This method stores the probability of misclassifying p_i = (F/i) and the standard deviation s_i = $\sqrt{pi(1-pi)/i}$, where i is the number of trials and F is the number of false predictions. These values are updated incrementally. Two levels are defined, a warning level and a drift level. Each of them is reached according to pre-defined conditions based on p_i and s_i and its minimum values $p_{i_{min}}$ and $s_{i_{min}}$. Note that it is possible to observe an increase in the error-rate reaching the warning level, followed by a decrease. This is considered a false alarm.

The continuous learning process consists of the following steps, as presented in [1]:

1) process the incoming records from the data stream using an incremental learning algorithm (base learner) to obtain a decision model capable of representing the underlying concept.
2) the drift detection method monitors the error-rate of the learning algorithm.
3) when the error-rate goes up the drift detection mechanism indicates,
   - **warning level**: store the incoming records into a warning window and prepare a new learner that processes incoming records while the warning level is signaled.
   - **drift level**: store the current model and its associated context into a model repository; use the model repository to find a past model with a context similar to the occurring context that performs well with the new data records (i.e., represents the current concept, this was presented by the authors in[1]). Reuse the model from the repository as base learner to continue the learning process as in point 1. If no model is found use the learner that was initiated during warning level period as base learner.
   - **false alarm (normal level after warning)**: the warning window is cleared and the learner used during the warning period is discarded. The learning process continues as in point 1, which is also the normal level in terms of drift detection.

III. Proposed memory-aware approach

Adaptation to concept recurrence depends on storing past models. In memory constrained situations, the challenge consists in assessing model utility and adapt the learning process to memory availability. Having a decision function will enable to decide which model to discard in such situations, freeing memory for a new model.

A. Preliminaries

1) Context: Context information depends on the data mining problem and is accessed through a known interface. We assume context modeling to be based on the Context Spaces model [11] where a context is represented as an object in a multidimensional Euclidean space. A context state C_i is defined as a tuple of N attribute-values,

$$C_i = (a_1^i, ..., a_n^i)$$

a_n^i represents the value of attribute n for context state C_i. A context space defines the regions of acceptable values for these attributes. In this work, we use numerical context attributes and the Euclidean distance as the measure to compare context states formulated as:

$$|C_i - C_j| = \sqrt{\sum_{K=1}^{N} dist(a_k^i - a_k^j)^2}$$

a_k^i represents the k^{th} attribute-value in a context state i. For numerical attributes distance is defined as:

$$dist(a_k^i, a_k^j) = \frac{(a_k^i - a_k^j)^2}{s^2}$$

where s is the estimated standard deviation for a_k. For nominal attributes distance is defined as:

$$dist(a_k^i, a_k^j) = \begin{cases} 0 & if \quad a_k^i = a_k^j \\ 1 & otherwise \end{cases}$$

We define comparison of contexts using the following formulation:

$$similarContext(C_i, C_j) = \begin{cases} true & if \quad |C_i - C_j| \leq \epsilon \\ false & if \quad |C_i - C_j| > \epsilon \end{cases}$$

ϵ is a pre-defined threshold (using zero as threshold means that the contexts are equal).

2) Model Storage: In the Naive Bayes learning algorithm, P_C is the table storing $P(C)$ which represents the observed frequency for each class C, and P_A is the vector that stores P(A_n|C) that represents the frequency table of each feature A_n given class C. This can be expressed as:

$$P_C = P(C) \quad and \quad P_A =< P(A_1|C), ..., P(A_n|C) >$$

In our proposed approach, storing a decision model M using this learning algorithm also requires storing:

- The most frequent context state observed during model M learning period as $freqC$=(a_1, ..., a_n), where each attribute value of $freqC$ is the most frequent value that each attribute takes in that learning period.
- The accuracy Acc_k of M is the accuracy value obtained during the learning period $_k$, with $numCRecords_k$ being the number of correctly classified records by M and $numRecords_k$ being the total number of records processed during $_k$. The accuracy value is updated if M is reused. This is formulated as:

$$Acc_k = \frac{numCRecords_k}{numRecords_k}$$

- T is the timestamp that records the time when the model M was stored.

Consequently each decision model M stored in the model repository is defined as the tuple:

$$M = \{P_C, P_A, freqC, Acc_k, T\}$$

3) Model utility metrics: The definition of model utility metrics is needed so that the mechanism can decide which models to keep or discard in situations of memory scarcity.

We propose the following metrics to define a selection function that chooses which model to remove from the repository,

- Model Accuracy - This metric represents the accuracy of the decision model for the period it was used.
- Mean Square Error - This metric measures the error of the decision model for a set of records (we use the records available in the warning window). The error prediction of model M_i, using the window W_n of n records in the form of (x, c), where c is the true class label for that record. The error of M_i on record (x, c) is 1 - $f_c^i(x)$, where $f_c^i(x)$ is the probability given by M_i that x is an instance of class c. The MSI_i metric can be expressed as:

$$MSE_i = \frac{1}{|W_n|} \sum_{(x,c) \in W_n} (1 - f_c^i(x))^2$$

- Context Similarity - We exploit the context stored within the decision models by using the similarity between contexts as a metric. This is used to maximize context heterogeneity between the models in the repository.
- Timestamp - The timestamp that is stored along with models can be used as a metric of model utility in cases where all the other metrics give the same value. Thus this metric can be used to break ties where the oldest model receives lowest utility according to the situation.

In situations of memory scarcity we make use of a function that selects from the model repository, using the proposed metrics, the model to be discarded. This function searches the model repository for the models with the lowest context distance, in order to maximize the context heterogeneity in the repository. From this subset the ones with highest mean square error are selected. If more than one model remains after this step, the model with lowest accuracy is returned. If still more than one model exists (i.e., the MSE and Accuracy metrics have the same value) the model with lowest timestamp is selected. In algorithm 1 the pseudo-code of this function is presented.

Algorithm 1 Model Selection Function: Selects the model to discard

Require: ModelRepository $MR \rightarrow M$:(P_C,P_A,C,Acc,T)

1: For each M_i, $M_j \in MR$ with $_i \neq _j$, compute $distance(C_i, C_j)$;
2: Let $Context_{min} \subset MR$, where $min(distance(C_i, C_j))$;
3: Let $Error_{max} \subset Context_{min}$, where $max(MSI_i)$;
4: Let $CandidateSet \subset Error_{max}$, where $min(Acc_i)$;
5: **return** $M_i \in CandidateSet$, where $min(T_i)$;

B. Proposed Extension of the Drift Detection Mechanism

We propose as an adaptation strategy to discard a stored model when the memory limit is reached and it is not possible to store further models. This simple strategy enables us to handle the problem of concept recurrence in scenarios with different memory constraints, which limits how many models can be kept for reuse. We are able to test and measure the accuracy loss in such scenarios.

The **sufficient memory** condition is defined as:

$$\begin{cases} if \quad (usedMemory + storageCost \leq MemoryLimit) \\ then \quad true \\ otherwise \quad false \end{cases}$$

The used memory is accessed through an interface with the device as assumed in Section II, $storageCost$ depends on the mining schema of the data records. Since all the stored models share the same data schema this value is a constant. This is a consequence of using the Naive Bayes algorithm as the size of the frequency tables that are stored (see Section III-A2) is determined by the mining schema. The $MemoryLimit$ value depends on the ubiquitous device memory given to run the learning process algorithm.

The proposed learning process is extended in the **drift level** case with the condition:

- if (not **sufficient memory**): discard the decision model returned by function (algorithm 1);

IV. Experimental results

In order to test the proposed learning process, an implementation was developed in Java, using the MOA [9] environment as a test-bed. The available evaluation features have been used and the *SingleClassifierDrift* class that implements the drift detection method [6] has been extended into our proposed approach. We used artificial and real world datasets to evaluate the proposed method.

A. Datasets

1) Artificial Dataset: As artificial dataset, the SEA Concepts [12] with MOA [9] as the stream generator was used. SEA Concepts is a *benchmark* data stream that uses different functions to simulate concept drift, allowing control over the target concepts and its recurrence in our experiment. The SEA Concepts dataset has two classes {class0, class1} and three features with values between 0 and 10 but only the first two features are relevant. The target concept function classifies a record as class1 if $f_1 + f_2 \leq \theta$ and otherwise as class0, f_1 and f_2 are the two relevant features and θ is the threshold value between the two classes. Four target concept functions are defined with threshold values 8, 9, 7 and 9.5 as proposed in the original paper [12].

2) Real World Dataset: As real world dataset we used the Electricity Market Dataset [7]. The data was collected from the Australian New South Wales Electricity Market, where the electricity prices are not stationary and are affected by the market supply and demand. The market demand is influenced by context such as season, weather, time of the day and central business district population density. The supply is influenced primarily by the number of on-line generators. An influencing factor for the price evolution of the electricity market is time. During the time period described in the dataset the electricity market was expanded with the inclusion of adjacent areas (Victoria state), which lead to more elaborated management of the supply as oversupply in one area could be sold interstate. The ELEC2 dataset contains 45312 records obtained from 7 May 1996 to 5 December 1998, with one record for each half hour (i.e., there are 48 instances for each time period of one day). Each record has 5 attributes, the day of week, the time period, the NSW demand, the Victoria demand, the scheduled electricity transfer between states and the class label. The class label identifies the change of the price related to a moving average of the last 24 hours. The class level only reflects deviations of the price on a one day average and removes the impact of longer term price trends. As shown in [7] the dataset exhibits substantial seasonality and is influenced by changes in context. This motivates its use as a real world dataset in our experiments.

B. Context and recurrent concepts definition

As context for the SEA dataset we used a numerical context feature space with two features a_1 and a_2 with values between 1 and 4. It was generated independently as a context stream where the context attribute a_1 is equal to the target concept function number, and a_2 value equals the target concept function 9 in 10 times, which introduces noise in the context stream. We generated 250000 records and changed the underlying concept every 15000 records. The test was executed with a 10% noise value as in the original paper [12], this means the class value of the training record is wrong in 10% of the records, testing how sensitive is the approach to noise.

For the Electricity Market dataset we have considered the classification problem to predict the changes in prices relative to the next half hour, using as predictive attributes, the time period, the NSW demand, the Victoria demand and the scheduled electricity transfer. As context we used the day of week attribute, as in [7] experiments using it lead to 10 different contextual clusters. We expect that the association of this context with the stored models achieves good accuracy results. However, one drawback of a real world dataset is that we do not know for sure what the actual *hidden context* is and when such changes occur, which makes it more difficult to evaluate the obtained results. This dataset was also used in [6] to test the drift detection method in real world problems, achieving good performance results.

C. Experiments

1) Reference experiment: For both datasets the approach proposed in this paper is compared in terms of accuracy with the *SingleClassifierDrift* implemented in MOA[9]. This represents a scenario without memory constrains to be used as reference for the two boundary cases (i.e., case where it is possible to store all the required models vs no models stored). The *SingleClassifierDrift* approach also uses the Naive Bayes algorithm and detects drift using the drift detection method [6]. When it occurs, the system learns a new model by forgetting the old one (i.e., it represents the case where the memory available to store additional models is zero). In the real world dataset we also compare results with an incremental Naive Bayes algorithm [4] (without any mechanism to adapt to drift), again to be used as reference.

2) Experiment with memory constraints: We compared the memory-aware approach in situations with memory constraints, with 7KBytes and 5KBytes and 3KBbytes of available memory. This allowed to test different scenarios, ranging from ones where it is possible to store enough models to represent different target functions and others where due to the strong memory constraints only a reduced number of models can be stored. Note that in the SEA of concepts dataset, this means being forced to store less models than existing target functions. This allowed us to observe and measure how the accuracy declines as the

 ISBN: 978-1-61208-100-7

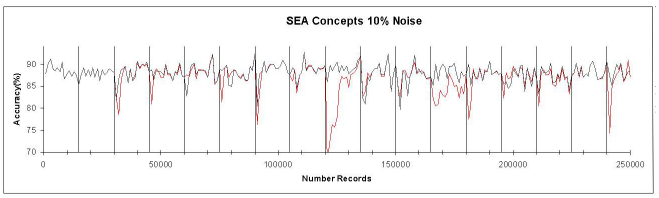

Figure 1. Comparison of accuracy with Proposed approach(black) vs SingleClassifierDrift(red) using the SEA concepts dataset. Black lines show when drift occurs

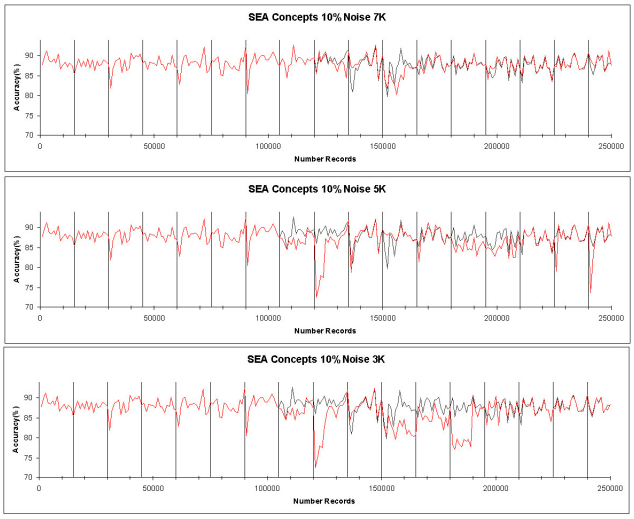

Figure 2. Comparison of accuracy with of the proposed approach with NoConstraints(black) vs Memory-constrained(red) using the SEA concepts dataset. Black lines show when drift occurs.

memory available is reduced and fewer models can be kept. Also it is important to understand the impact of discarding models and the utility of the stored models in the adaptation to recurrence in such memory constrained scenarios.

D. Results with SEA of Concepts Dataset

As can be seen in Figure 1 our approach leads to better accuracy than *SingleClassifierDrift*. In general our approach adapted to drift faster and the models selected using context integration were able to represent the target concepts. This is not observed in the *SingleClassifierDrift* approach that always has to relearn the underlying concept from scratch after drift is detected. It is also noticeable that our approach achieves a more stable accuracy over time, as it recovers much faster from drift than the approach without stored models. The proposed approach obtained 2046 more correct predictions. The integration of context enables to exploit the associations between recurrent concepts and context as a way to track concept recurrence. In situations where this association exists it is possible to achieve better results.

In Figure 2, the memory-aware approach is compared in scenarios with different available memory values. As expected, when memory is reduced, the accuracy is reduced.

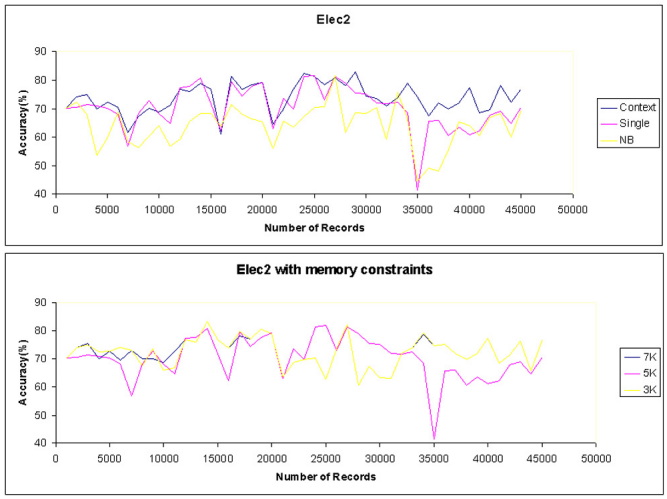

Figure 3. Above(No memory contraints): Comparison of accuracy between the proposed approach(Context), single classifier drift (Single) and incremental NaiveBayes(NB). Bellow: Comparison of accuracy using the proposed approach with different memory values.

In the test scenario with 7Kbytes it is possible to store 7 models, which allow us to keep more than one model for each concept. As a result the performance was very close (with only more 278 misclassified records) to the scenario without memory constraints where 10 models are stored. In the scenario with 5Kbytes, the reduction in accuracy is more significant, with more than 1656 misclassified records and the accuracy curve starts to resemble the *SingleClassifierDrift* seen in 1 especially around records 120000 and 165000 (where the concept with θ=8 is the target concept). Finally in the scenario with 3Kbytes only 3 models can be kept in memory and as a result the performance is further reduced, with more 2881 misclassified records.

E. Results with Electricity Market Dataset

As can be seen in Figure 3, the proposed approach obtained better accuracy results (i.e., 73,4%), which represents a gain of 3,7% and 11% when compared with *SingleClassifierDrift* and incremental NaiveBayes respectively. We can also observe that the proposed approach achieves a more stable accuracy and recovers faster from changes. This can be seen clearly around record 35000.

In relation to the experiments with memory constraints the results show that the overall accuracy is similar between the experiments, in the order of 71% correctly classifier records. For all the tested scenarios the proposed approach still obtains better overall results than the memoryless approach (i.e., *SingleClassifierDrift*) but the accuracy over specific periods depends on the model that is reused and which ones were previously discarded in situations of memory scarcity. This is a direct result of the proposed decision function, and again such difference can be seen around record 35000,

 ISBN: 978-1-61208-100-7

where the tests that kept the adequate model (i.e., 7K and 3K) are able to show improved adaptation.

V. Conclusions and Future Work

In this work, we have proposed a memory-aware approach for the problem of data stream classification that integrates context information with learned models, improving adaptation to drift and exploiting concept recurrence.

Several metrics to define model utility for the challenge of memory adaptation were presented. These were developed in order for the proposed adaptation strategy to suffer minimal loss in accuracy and adaptation to drift when compared to approaches where resources are unbounded and more models can be kept in memory.

We have also tested our approach with the artificial benchmark dataset SEA Concepts and the real world dataset Electricity Market. The experimental results show the advantages of the proposed approach for situations with memory constraints. This is a consequence of minimizing the accuracy loss. As the available memory is reduced while keeping the more relevant models, these are available when the concepts they represent reoccur. We should also note that this is a general approach and better adaptations are expected when using domain sensitive metrics and context. Despite the promising results, we are not exactly sure when drift occurs in the Electricity Market dataset and what really affects change. This limits the depth to which we can evaluate such results. However, such drawbacks are a consequence of learning from real world data.

As future work we plan to test the current approach in a real ubiquitous device with real world problem, and further develop the idea presented in this paper using domain sensitive criteria and context along with more sophisticated adaptation strategies.

Acknowledgments

This research is partially financed by project TIN2008-05924 of Spanish Ministry of Science and Innovation. The work of J.P. Bártolo Gomes is supported by a Phd Grant of the Portuguese Foundation for Science and Technology (FCT).

References

[1] J.P. Bartolo Gomes, E. Menasalvas, and P. Sousa. Tracking Recurrent Concepts Using Context. In *Rough Sets and Current Trends in Computing, Proceedings of the Seventh International Conference RSCTC2010*, pages 168–177. Springer, 2010.

[2] M.M. Gaber, S. Krishnaswamy, and A. Zaslavsky. Ubiquitous data stream mining. In *Current Research and Future Directions Workshop Proceedings held in conjunction with PAKDD*. Citeseer, 2004.

[3] M.M. Gaber and P.S. Yu. A holistic approach for resource-aware adaptive data stream mining. *New Generation Computing*, 25(1):95–115, 2006.

[4] J. Gama and M.M. Gaber. *Learning from data streams: processing techniques in sensor networks*. Springer-Verlag New York Inc, 2007.

[5] J. Gama and P. Kosina. Tracking Recurring Concepts with Meta-learners. In *Progress in Artificial Intelligence: 14th Portuguese Conference on Artificial Intelligence, Epia 2009, Aveiro, Portugal, October 12-15, 2009, Proceedings*, page 423. Springer, 2009.

[6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. *Lecture Notes in Computer Science*, pages 286–295, 2004.

[7] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales, 1999.

[8] M.B. Harries, C. Sammut, and K. Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.

[9] G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis, 2007 - http://sourceforge.net/projects/moa-datastream/.

[10] I. Katakis, G. Tsoumakas, and I. Vlahavas. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, pages 1–21.

[11] A. Padovitz, SW Loke, and A. Zaslavsky. Towards a theory of context spaces. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 38–42, 2004.

[12] W.N. Street and Y.S. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM New York, NY, USA, 2001.

[13] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 2004.

[14] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[15] Y. Yang, X. Wu, and X. Zhu. Combining proactive and reactive predictions for data streams. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, page 715. ACM, 2005.

 ISBN: 978-1-61208-100-7