Choice of Best Samples for Building Ensembles in Dynamic Environments

Joana Costa[1,2], Catarina Silva[1,2], Mário Antunes[1,3], and Bernardete Ribeiro[2]

[1] School of Technology and Management, Polytechnic Institute of Leiria; Portugal
{joana.costa,catarina,mario.antunes}@ipleiria.pt
[2] Center for Informatics and Systems of the University of Coimbra (CISUC), Department of Informatics Engineering; Portugal;
{joanamc,catarina,bribeiro}@dei.uc.pt
[3] Center for Research in Advanced Computing Systems (CRACS), INESC-TEC, University of Porto; Portugal; {mantunes}@dcc.fc.up.pt

Abstract. Machine learning approaches often focus on optimizing the algorithm rather than assuring that the source data is as rich as possible. However, when it is possible to enhance the input examples to construct models, one should consider it thoroughly. In this work, we propose a technique to define the best set of training examples using dynamic ensembles in text classification scenarios. In dynamic environments, where new data is constantly appearing, old data is usually disregarded, but sometimes some of those disregarded examples may carry substantial information. We propose a method that determines the most relevant examples by analysing their behaviour when defining separating planes or thresholds between classes. Those examples, deemed better than others, are kept for a longer time-window than the rest. Results on a *Twitter* scenario show that keeping those examples enhances the final classification performance.

Keywords: Dynamic Environments, Ensembles, Drift, Text Classification, Social Networks

1 Introduction

Information spread in online scenarios has created a new information sharing paradigm. Nowadays, it is possible to create and disseminate information in numerous formats by publishing it world-wide, making the Web responsible for a deluge of data. Albeit we can undoubtedly benefit from all these data, one major drawback of such overflow is the inability to easily perceive important, significant and accurate information. This challenge arises not only because the amount of data is overwhelming to process, but also because time plays an important role by fast out-dating information.

In dynamic environments drift occurs and deployed models performance is reduced when changes occur between the distribution that generated the data used to define the model and the current drifted scenario. To handle such challenges, some form of model ageing must be put in place.

In [1–3] the proposed approaches try to detect that a drift occurred and react accordingly. This reaction is usually non-trivial, since the new samples can carry more relevant and new information that should probably contribute more to the final model [4]. In fact, in dynamic environments, such as in social networks we will be using as case study, effective learning requires a learning algorithm with the ability to detect context changes without being explicitly informed about them, quickly recovering from the context change and adjusting its hypothesis to the new context [1, 5].

A lot of effort has been place in adapting to such news' samples. However, previous samples can also play an important role, and previously experienced situations should also be considered when old contexts and corresponding concepts reappear [6]. In this work we propose a framework for choosing the best samples for building classifiers in dynamic environments. Using a sliding time-window approach, models are retrained with updated real-time samples. Furthermore, we analyze previous samples' behaviour, e.g. when defining separating planes, to detect stronger examples and keep them in the pool. Hence, these chosen samples are kept for a longer time-window than the rest. Tests carried out on a *Twitter* stream case study support the hypothesis that keeping such chosen examples enhances the final classification performance.

The rest of the paper is organized as follows. Section 2 introduces background concepts and state of the art on dynamic environments and social networks. Section 3 presents the proposed framework for choosing the best samples for building classifiers in dynamic environments. In Section 4 the experimental setup is set, including the case study dataset and the performance metrics, followed by the experimental results and analysis in Section 5. Finally, Section 6 concludes the paper with conclusions and future work.

2 Background

2.1 Social networks

Social networks have settled definitely in the daily routine of Internet users. They have also gained increasing importance and are being widely studied in many fields of research over the last years, such as computer, social, political, business and economical sciences.

There exists a wide set of distinct social networks for different purposes and scope, being *Twitter* and *Facebook* two of the most popular ones. In a purely research perspective, these social networks make accessible through their own public API a deluge of relevant data related with users' daily status, news and events. Data is produced in a non-deterministic way, turning social networks in a dynamic environment in which we may apply learning and detection strategies and algorithms.

The focus of our work is *Twitter* social media platform (`www.twitter.com`), more precisely on applying learning and classification stategies to cope with different types of variations of context (*drift*) through time [1, 7].

2.2 Dynamic environments

Social networks can be seen as a dynamic (also entitled as non-stationary) environment, in which information is produced by users in a timely order. Time plays a crucial role in *Twitter* information processing, as past events can give important insights to understand how previously seen information is relevant to improve learning and classification of future unseen and related events.

In that sense, learning strategies would be able to learn in dynamic environments and apply innovative strategies to dead with a *"recent memory"* of past events, in order to better identify future and unseen ones.

There can be several approaches to tackle dynamic environments [4]: instance selection, instance weighting and ensemble learning. A review of concept drift applied to intrusion detection is presented in [8].

Learn++.NSE and Learn++.CDS [3, 9] are algorithms to deal with drift, namely with imbalanced datasets. An ensemble technique, DWM-WIN, was proposed in [2], to overcome the known limits of dynamic weight majority [10], namely not taking into consideration the timestamp of classifiers or the previous performances.

In this section we presented some examples of the importance of tackling drift in dynamic scenarios like social networks, and particularly in *Twitter*. Multiple applications like spam email filtering, intrusion detection, recommendation systems, event detection, or improve search capabilities are just pointed examples [1].

3 Proposed approach

This section describes the proposed approach to define the best set of training examples using dynamic ensembles in text classification scenarios. We will firstly present our *Twitter* classification problem and then proceed with formalizing the proposed models.

3.1 Case study: Twitter stream

Twitter stream constitutes a paradigmatic example of a text-based scenario where drift phenomena occur commonly. *Twitter* is a micro-blogging service where users post text-based messages up to 140 characters, also known as *tweets*. It is also considered one of the most relevant social networks, along with *Facebook*, as millions of users are connected to each other by a following mechanism that allows them to read each others posts.

Twitter is also responsible for the popularization of the *'hashtag'* concept. An *hashtag* is a single word started by the symbol "#" that is used to classify the message content and to improve search capabilities. Besides improving search capabilities, *hashtags* have been identified as having multiple and relevant potentialities, like promoting the phenomenon described in [11] as *micro-meme*, i.e. an idea, behavior, or style that spreads from person to person within a culture [12]. By tagging a message with a trending topic *hashtag*, a user expands

the audience of the message, compelling more users to express themselves about the subject [13].

Considering the importance of the *hashtag* in *Twitter*, it is relevant to study the possibility of evaluating message contents in order to predict its *hashtag*. If we can classify a message based on a set of *hashtags*, we are able to suggest an *hashtag* for a given *tweet*, bringing a wider audience into discussion [14], spreading an idea [15], get affiliated with a community [16], or bringing together other Internet resources [17].

This case study aims to classify *Twitter* messages. A *Twitter* classification problem can be described as a multi-class problem that can be cast as a time series of *tweets*. It consists of a continuous sequence of instances, in this case, *Twitter* messages, represented as $\mathcal{X} = \{x_1, \ldots, x_t\}$, where x_1 is the first occurring instance and x_t the latest. Each instance occurs at a time, not necessarily in equally spaced time intervals, and is characterized by a set of features, usually words, $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$. Consequently, instance x_i is denoted as the feature vector $\{w_{i1}, w_{i2}, \ldots, w_{i|W|}\}$.

When x_i is a labelled instance it is represented as the pair (x_i, y_i), being $y_i \in \mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$ the class label for instance x_i.

We have used a classification strategy previously introduced in [7], where the *Twitter* message *hashtag* is used to label the content of the message, which means that y_i represents the *hashtag* that labels the *Twitter* message x_i.

Notwithstanding it is a multi-class problem in its essence, it can be decomposed in multiple binary tasks in a one-against-all binary classification strategy. In this case, a classifier h^t is composed by $|Y|$ binary classifiers.

3.2 Learning models

We are focusing on dynamic ensembles in text classification scenarios, where the ensemble must adapt to deal with changes usually dependent on hidden contexts. One of the major challenges is the amount of data, specially when dealing with streams. It is sometimes unfeasable to store all the previously seen data, but it may carry substantial information for future use.

In [18] we have studied the impact of longstanding examples in future classification time-windows. The rationale of the presented idea was to store previously seen examples for a period of time regardless the effect they might have as a solo example. The most relevant action was to keep examples for a period of time instead of discarding them for future use. We were also not dealing with ensembles but single classifiers. Differently from that approach, we are now proposing to choose examples based on the effect they might have individually.

Our baseline model, created for comparison purposes, proposes to store all the information gathered by storing models and combining them as an ensemble. For each collection of documents $\mathcal{T}$, that contain both positive and negative examples and occur in a time-window t, a classifier $\mathcal{C}^t$ is trained and stored. When a new collection of documents in the subsequent time-window occurs, all the previously trained classifiers are loaded, and the system will classify the newly seen examples. The prediction function of the ensemble, composed by the

set of classifiers already created, is a combined function of the outputs of all the considered classifiers. A majority voting strategy where each model participates equally is then put forward. The documents of the previously seen time-windows are not stored in this approach even though the possible learning information is stored along in the classifier trained immediately after it.

We then propose an ensemble learning model, namely reinforced model. The main difference is that we define a collection of documents $\mathcal{R}$ that contain all the classification errors that occur in the time-windows prior to t. The classification errors are considered based on the ensemble classification and not in each model classification output. For each time-window t, a classifier $\mathcal{C}^t$ is trained with the collection of documents $\mathcal{T}$ plus the collection of documents $\mathcal{R}$ and stored. When a new collection of documents in the subsequent time-window occur, all the previously trained classifiers are loaded, and will classify the newly seen examples participating equally to the final decision of the ensemble.

The collection of documents $\mathcal{R}$ might retain the misclassified examples indefinitely or be pruned in a time-based approach. If pruned, the lifetime of an example in $\mathcal{R}$ is dependent of a pre-defined time-window size g, which means that, in time-window t, $\mathcal{R}$ contains the misclassified examples that occur in all time-windows that satisfy the condition t-g> 0.

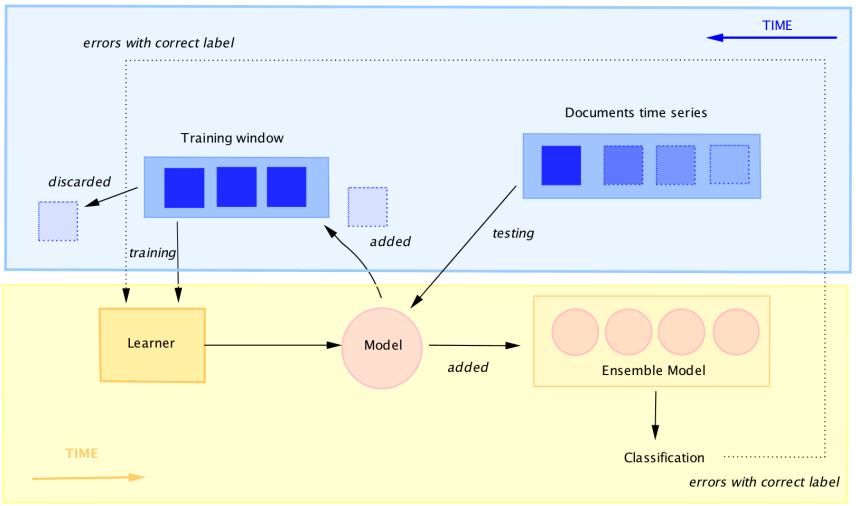

Fig. 1: Proposed models

Figure 1 depicts the proposed models. It is important to understand that we represented time in two different directions because we are working with a time serie and the last seen scenario is the input for the new one. The major difference between both models is using the outcome of the classification as a new incoming in the subsequent training phase.

4 Experimental Setup

In this section we will detail the dataset we propose to test and evaluate our approach. We then characterize the methodology for document representation and proceed dealing with the pre-processing methods. Finally, we conclude by introducing the performance metrics used to evaluate the proposed approach.

4.1 Dataset

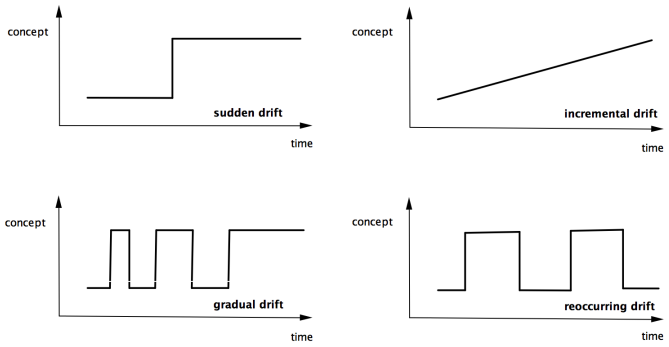

Fig. 2: Different types of drift

The dataset we have defined to evaluate and validate our strategy was carried out by defining 10 different *hashtags* that would represent our drifts (see Fig. 2), based on the assumption that they would denote mutually exclusive concepts, like *#realmadrid* and *#android*. By trying to use mutually exclusive concepts we intend to avoid misleading a classifier, as two different *tweets* could represent the same concept, and that way introducing a new variable to our scenario that could mislead the possible obtained results. In order to achieve a considerable amount of *tweets*, and consequently diversity, we have chosen trending *hashtags* like *#syrisa* and *#airasia*. Table 1 shows the chosen *hashtags* and the corresponding drift they represent. This correspondence was done arbitrarily and do not correspond to any possible occurrence in the real *Twitter* scenario, since as stated above, no information is known about the occurrence of drifts in *Twitter*.

The *Twitter API* (`https://dev.Twitter.com/`) was then used to request public *tweets* that contain the defined *hashtags*. The requests have been cared of between 28 December 2014 and 21 January 2015 and *tweets* were only considered if the user language was defined as English. We have requested more than 75.000 *tweets* concerning the given *hashtags*, even though some of them were discarded, like for instance those *tweets* containing no message content besides the *hashtag*. The *hashtag* was then removed from the message content in order to be exclusively used as the document label. The *tweets* matching this presumptions were

considered labelled and suited for classification purposes, and were used by their appearing order in the public feed.

We have simulated the different types of drift by artificially defining timestamps to the previously gathered *tweets*. Drift Oriented Tool System (DOTS) is a drift oriented framework we have presented in [5]. DOTS was developed to dynamically create datasets with drift and is available for free download at `http://dotspt.sourceforge.net/`. DOTS is used to create the described dataset. It receives the *tweets* requested by the *Twitter API* and reproduces the defined artificial time-stamped time-windows. Time is represented as 100 continuous time windows, in which the frequency of each *hashtag* is altered in order to represent the defined drifts. Each *tweet* is then timestamped so it can belong to one of the time windows we have defined. For instance, Sudden #1 is represented by the appearance of 500 *tweets* with the *hashtag #syrisa* in each time windows from 25 to 32. It does not appear in any other time windows. Differently from textit Sudden #1, *Sudden #2* is represented with only 200 *tweets* with the *hashtag #airasia* in each time windows from 14 to 31. We tried to simulate a more soft occurring drift, but with a more long-standing appearance.

By making both concepts disappear, in time windows, 32 and 31, respectively, we also intended to simulate the opposite way of the [19] proposed sudden drift. Due to space constraints it is unbearable to present a table with the frequency of each *hashtag* in each time window, but it is important to state that *Incremental #2* and *Gradual #2* are represented by the same number of *tweets* in an equal number of time windows, but in a descent way than represented in *Incremental #1* and *Gradual #1.Normal #1*, *Normal #2* and *Normal #3* differ in the number of *tweets* that appear in a constant way in all time windows. Our final dataset contains 34.240 *tweets*.

Drift	**Hashtag**
Sudden #1	#syrisa
Sudden #2	#airasia
Gradual #1	#isis
Gradual #2	#bieber
Incremental #1	#android
Incremental #2	#ferrari
Reoccurring	#realmadrid
Normal #1	#jobs
Normal #2	#sex
Normal #3	#nfl

Table 1: Mapping between type of drift and hashtag.

4.2 Representation and Pre-processing

A *tweet* is represented as one of the most commonly used document representation, which is the vector space model, also known as *Bag of Words*. The collection of features is built as the dictionary of unique terms present in the documents collections. Each *tweet* of the document collection is indexed with the *bag* of the terms occurring in it, i.e., a vector with one element for each term occurring in the whole collection. The weighting scheme used to represent each term is the *term frequency - inverse document frequency*, also know as *tf-idf*.

High dimensional space can cause computational problems in text classification problems where a vector with one element for each occurring term in the whole connection is used to represent a document. Also, overfitting can easily occur which can prevent the classifier to generalize and thus the prediction ability becomes poor. In order to reduce feature space pre-processing methods were applied. These techniques aim at reducing the size of the document representation and prevent the mislead classification as some words, such as articles, prepositions and conjunctions, called *stopwords*, are non-informative words, and occur more frequently than informative ones. An english-based *stopword* dictionary was used, but *Twitter* related words like "*rt*" or "*http*" were also considered as they can be seen as *stopwords* in the *Twitter* context. *Stopword removal* was then applied, preventing those non informative words from misleading the classification.

Stemming method was also applied. This method consists in removing case and inflection information of each word, reducing it to the word stem. Stemming does not alter significantly the information included, but it does avoid feature expansion. Pre-processing methods were applied in DOTS.

4.3 Learning and Evaluation

The evaluation of our approach was done by the previously described dataset and using the Support Vector Machine (SVM) as stated above. This machine learning method was introduced by Vapnik [20], based on his Statistical Learning Theory and Structural Risk Minimization Principle. The idea behind the use of SVM for classification consists on finding the optimal separating hyperplane between the positive and negative examples. Once this hyperplane is found, new examples can be classified simply by determining which side of the hyperplane they are on. SVM constitute currently the best of breed kernel-based technique, exhibiting state-of-the-art performance in text classification problems [21–23]. SVM were used in our experiments to construct the proposed models. Based on [18] a 4 time-window size training window will be used.

In order to evaluate a binary decision task we first define a contingency matrix representing the possible outcomes of the classification, as shown in Table 2.

Several measures have been defined based on this contingency table, such as, error rate ($\frac{b+c}{a+b+c+d}$), recall ($R = \frac{a}{a+c}$), and precision ($P = \frac{a}{a+b}$), as well as combined measures, such as, the van Rijsbergen F_β measure [24], which combines recall and precision in a single score:

	Class Positive	Class Negative
Assigned Positive	a	b
	(True Positives)	(False Positives)
Assigned Negative	c	d
	(False Negatives)	(True Negatives)

Table 2: Contingency table for binary classification.

$$F_\beta = \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R}. \tag{1}$$

F_β is one of the best suited measures for text classification used with $\beta = 1$, i.e. F_1, an harmonic average between precision and recall (2), since it evaluates well unbalanced scenarios that usually occur in text classification settings and particularly in text classification in the *Twitter* environment.

$$F_1 = \frac{2 \times P \times R}{P + R}. \tag{2}$$

Considering the proposed approach and the fact that we are working with a time series and we use a one-against-all strategy, we will have a classifier for each batch of the time series that is composed by $|Y|$ binary classifiers, being $|Y|$ the collection of possible labels. To perceive the performance of the classification for each drift pattern, we will consider all the binary classifiers that were created in all the time series batches. To evaluate the performance obtained across time, we will average the obtained results. Two conventional methods are widely used, specially in multi-label scenarios, namely macro-averaging and micro-averaging. Macro-averaged performance scores are obtained by computing the scores for each learning model in each batch of the time series and then averaging these scores to obtain the global means. Differently, micro-averaged performance scores are computed by summing all the previously introduces contingency matrix values (a,b,c and d), and then use the sum of these values to compute a single micro-averaged performance score that represents the global score.

5 Experimental results

In this section we evaluate the performance obtained on the *Twitter* data set using the two approaches described in Section 3, namely the baseline model approach and the reinforced model approach. In the reinforced model approach we obtained results by storing examples during 4 time-windows, represented as "$Reinforced_4$", and by storing examples *ad eternum*, named "$Reinforced_{forever}$". Table 3 summarises the performance results obtained by classifying the dataset, considering the micro-averaged F_1 measure.

Drift	Baseline	$Reinforced_4$	$Reinforced_{forever}$
Sudden #1	74,80%	75,45%	72,37%
Sudden #2	87,80%	88,06%	86,55%
Gradual #1	52,55%	54,82%	89,03%
Gradual #2	62,21%	63,43%	89,21%
Incremental #1	88,58%	88,99%	94,86%
Incremental #2	77,21%	79,26%	74,28%
Reoccurring	35,33%	36,63%	72,42%
Normal #1	70,89%	71,86%	91,89%
Normal #2	90,49%	91,01%	94,69%
Normal #3	81,52%	83,74%	73,60%
Average of micro-averaged F_1	78,27%	79,33%	83,75%

Table 3: Micro-averaged F_1

Analysing the table we can observe that globally, and considering the average of the micro-averaged F_1, the storage of the priorly misclassified examples improves the overall classification. This is normal and expected as the learning models are trained with more informative examples and this leads to a better performance.

It is particularly important to note that model "$Reinforced_4$", which stores relevant examples for 4 time-windows, presents an improvement in the classification performance of all classes, regardless the type of drift they represent. This demonstrates the importance of the misclassified examples to improve the classification performance of the subsequent time-windows.

Nevertheless, when considering storing those examples for longer periods, specially *ad eternum*, one must notice that this improve is not straightforward. Most classes benefit from storing examples, and we have a significant improve in the average of the micro-averaged F_1, that increases from 78,27% to 83,27%, but some classes, namely *Sudden#1*, *Sudden#2*, *Incremental#2* and *Normal#3* have a worst classification performance. Firstly, both classes that represent a sudden drift have a performance decrease, *Sudden#1* from 74,80% to 72,37% and *Sudden#2* from 87,80% to 86,55%. We are confident that this decrease might be explained by the nature of the drift pattern.

A sudden drift is characterized by an abrupt increase of the frequency of a given class that occur during a period of time, followed by its disappearance. Storing examples that were misclassified, specially the positive ones that appeared firstly and remained misclassified until the classifier identified them as positive, will delude future classifiers, when the drift pattern is no longer represented. Secondly, we have identified a performance decrease in the classification of the class that represent *Incremental#2* drift. Similarly to what is happening with the sudden drift, the positive misclassified examples might be contributing to this decline, as the frequency of examples of this class is vanishing in time. Finally, we have the *Normal#3* drift. Differently from the mentioned above, the frequency of this drift is not diminishing in the time serie. There is nothing in

the drift pattern that allows us to infer what is happening, as it is exclusive to the *Normal#3* drift, and does not appear in the *Normal#1* or *Normal#2*, with the same nature. Although this is a supposition, that must be validated in future work, we do believe that it might be related to the class, that is the *hashtag* we have chosen to represent it. One of the possible problems that might arise from our approach is to store examples that are not representative of the class.

As we cannot guaranteed that a message is well-classified by its *hashtag*, we might be propagating errors by storing examples that were misclassified, but, differently from what we want, that is to store the most informative ones, we might be propagating the ones that do not represent the class at all. This is a problem that might arise in a dataset like ours, because it is impossible to validate that the *Twitter* user that wrote the *tweet*, is using the *hashtag* correctly.

6 Conclusions

In this paper we propose a method to determine the most relevant examples, by analysing their behaviour when defining separating planes or thresholds between classes. Those examples, deemed better than others, are kept for a longer time-window than the rest. The main idea is to boost the classification performance of learning models by providing additional and significant information.

We have used a *Twitter* case study to show that keeping those examples enhances the final classification performance. Since it is not known which types of drift occur in the context of social networks, and particularly in *Twitter*, we have also simulated different types of drift in an artificial dataset to evaluate and validate our strategy.

The results revealed the usefulness of our strategy, as the results improved by 5% in comparing to the baseline approach, considering the average of the micro-averaged F_1.

It is also important to conclude that we have shown that retaining informative examples during the right amount of time can improve the learners' ability to identify a given class, independently from the drift pattern the class is representing. We do believe that it is problem dependent, even thought it is an important insight in dynamic models, as they are particularly difficult learning scenarios. A special attention must be given to classes that tend to disappear, as retaining examples, in this particular case, for long periods can lead to misclassications.

Our future work will include not only a more profound study about the longevity of those examples, i.e., for how long is it relevant to retain those examples. By understanding the suitable longevity of those examples, we can maximize the cost benefit relation between the storage computational complexity and the classification performance increase. Another effort should be done in minimizing the use of those examples. In our approach we retain the relevant examples and present them to all models that compose the ensemble, but in future work we want to understand if we could have a similar income if we retain examples in a model based strategy, instead of an ensemble based one. The question that arises is that an example can be relevant to a model but irrelevant

to another, and thus we can retain the example and provide it just to the model that needs it, instead of all models that compose the ensemble.

7 Acknowledgments

This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project «POCI-01-0145-FEDER-006961», and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

This work was supported by national funds through the Portuguese Foundation for Science and Technology (FCT), and by the European Regional Development Fund (FEDER) through COMPETE 2020 – Operational Program for Competitiveness and Internationalization (POCI).

References

1. J. Costa, C. Silva, M. Antunes, and B. Ribeiro, "Concept drift awareness in Twitter streams," in *Proc. 13th Int. Conference on Machine Learning and Applications*, 2014, pp. 294–299. 2, 3
2. D. Mejri, R. Khanchel, and M. Limam, "An ensemble method for concept drift in nonstationary environment," *Journal of Statistical Computation and Simulation*, vol. 83, no. 6, pp. 1115–1128, 2013. 2, 3
3. G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2283–2301, 2013. 2, 3
4. A. Tsymbal, "The problem of concept drift: definitions and related work," Department of Computer Science, Trinity College Dublin, Tech. Rep., 2004. 2, 3
5. J. Costa, C. Silva, M. Antunes, and B. Ribeiro, "Dots: Drift oriented tool system," in *Processings of the 22nd International Conference on Neural Information Processing (ICONIP)*, 2015, pp. 615–623. 2, 7
6. G. Widmer and M. Kubat, "Effective learning in dynamic environments by explicit context tracking," in *Proc. European Conference on Machine Learning*, 1993, pp. 227–243. 2
7. J. Costa, C. Silva, M. Antunes, and B. Ribeiro, "Defining semantic meta-hashtags for Twitter classification," in *Proc.11th Int. Conference on Adaptive and Natural Computing Algorithms*, 2013, pp. 226–235. 2, 4
8. J. Kim, P. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, "Immune system approaches to intrusion detection - a review," *Natural Computing*, vol. 6, no. 4, pp. 413–466, 2007. 3
9. R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, pp. 1517–1531, 2011. 3
10. J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: a new ensemble method for tracking concept drift," in *Proc. 3rd Int. Conference on Data Mining*, 2003, pp. 123–130. 3

11. J. Huang, K. M. Thornton, and E. N. Efthimiadis, "Conversational tagging in Twitter," in *Proc. 21st ACM conference on Hypertext and hypermedia*, 2010, pp. 173–178. 3
12. (2012, October) Merriam-webster's dictionary. 3
13. M. Zappavigna, "Ambient affiliation: A linguistic perspective on Twitter," *New Media & Society*, vol. 13, no. 5, pp. 788–806, 2011. 4
14. S. Johnson, "How Twitter will change the way we live," *Time Magazine*, vol. 173, pp. 23–32, 2009. 4
15. O. Tsur and A. Rappoport, "What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities," in *Proc. 5th Int. Conference on Web Search and Data Mining*, 2012, pp. 643–652. 4
16. L. Yang, T. Sun, M. Zhang, and Q. Mei, "We know what @you #tag: does the dual role affect hashtag adoption?" in *Proc. 21st Int. Conference on World Wide Web*, 2012, pp. 261–270. 4
17. H.-C. Chang, "A new perspective on Twitter hashtag use: diffusion of innovation theory," in *Proc. 73rd Annual Meeting on Navigating Streams in an Information Ecosystem*, 2010, pp. 85:1–85:4. 4
18. J. Costa, C. Silva, M. Antunes, and B. Ribeiro, "The impact of longstanding messages in micro-blogging classification," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8. 4, 8
19. I. Zliobaite, "Learning under concept drift: an overview," Vilnius University, Faculty of Mathematics and Informatic, Tech. Rep., 2010. 7
20. V. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1999. 8
21. T. Joachims, *Learning Text Classifiers with Support Vector Machines*. Kluwer Academic xpublishers, Dordrecht, NL, 2002. 8
22. S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002. 8
23. J. Costa, C. Silva, M. Antunes, and B. Ribeiro, "On using crowdsourcing and active learning to improve classification performance," in *Proc. 11th Int. Conference on Intelligent Systems Design and Applications*, 2011, pp. 469–474. 8
24. C. van Rijsbergen, *Information Retrieval*. Butterworths Ed., 1979. 8