DOTS: Drift Oriented Tool System

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] Department of Informatics Engineering, Center for Informatics and Systems of the University of Coimbra (CISUC); Portugal
{joanamc,catarina,bribeiro}@dei.uc.pt
[3] Center for Research in Advanced Computing Systems, INESC-TEC, University of Porto; Portugal
mantunes@dcc.fc.up.pt

Abstract. Drift is a given in most machine learning applications. The idea that models must accommodate for changes, and thus be dynamic, is ubiquitous. Current challenges include temporal data streams, drift and non-stationary scenarios, often with text data, whether in social networks or in business systems. There are multiple drift patterns types: concepts that appear and disappear suddenly, recurrently, or even gradually or incrementally. Researchers strive to propose and test algorithms and techniques to deal with drift in text classification, but it is difficult to find adequate benchmarks in such dynamic environments.
In this paper we present DOTS, Drift Oriented Tool System, a framework that allows for the definition and generation of text-based datasets where drift characteristics can be thoroughly defined, implemented and tested. The usefulness of DOTS is presented using a Twitter stream case study. DOTS is used to define datasets and test the effectiveness of using different document representation in a Twitter scenario. Results show the potential of DOTS in machine learning research.

Keywords: Drift, Text classification, Learning algorithms, Software Tool

1 Introduction

The usual challenge for machine learning approaches is to build models that can perform well in classifying new data in production settings. Research efforts are usually put in proposing, implementing and testing innovative algorithms and techniques to deal with such challenges.

One major risk is that once a model is deployed, its performance can be significantly reduced when there is a drift in the distribution generating the new data when compared with the distribution that generated the data used to define the model. In machine learning this can be called as *model ageing*, in an analogy with *software ageing* [1]. Not all models age well and there are several recent approaches [2–4] that try to deal with such challenges, namely by proposing dynamic techniques that try to detect, or deal, with drifts in different scenarios.

In the presence of drift, learning is not an easy task and requires special approaches, different from those commonly used, as the arriving instances can not be treated as equally important contributors to the final model [5]. In non-stationary environments, e.g. social networks as the Twitter stream, effective learning requires a learning algorithm with the ability to detect context changes without being explicitly informed about them, quickly recover from the context change and adjust its hypothesis to the new context [2]. It should also make use of previously experienced situations when old contexts and corresponding concepts reappear [6]. Additionally, drifts can have different patterns and thus must be treated differently. Four types of drift can be identified, as can be seen in Fig. 1: sudden, gradual, incremental and reoccurring [7].

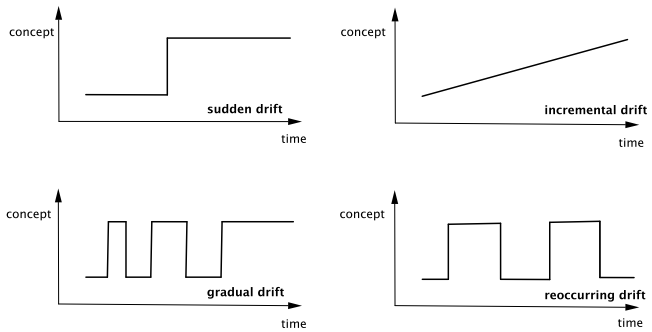

Fig. 1: Types of drift.

- **Sudden drift** occurs when the speed of the drift is high and a concept appears or disappears in an abrupt way. Although it is usually stated as sudden or abrupt drift, it is also sometimes referred as concept change.
- **Gradual drift** is another type of drift characterized by a low drift rate. Occurs when the probability of a given context to be associated with a concept decreases during a certain period of time. Moreover, the probability to be associated with another context increases proportionally.
- **Incremental drift**, also considered a subgroup of gradual drift, can be considered differently because the change between the two concepts is very slow being just perceived when looking to what is occurring during a larger period of time.
- **Reoccurring drift** occurs when previously active concept reappears after a certain period of time. It is noteworthy referring that the seasonality of the change must be previously unknown, otherwise the core assumption of the uncertainty about the future would be compromised.

To build drift aware datasets with blended temporal distributions is a challenging task. Therefore, it is not straightforward to find acceptable benchmarks in dynamic environments. To tackle this issue, in this paper we propose the Drift Oriented Tool System (DOTS), a framework that allows for the definition and

generation of text-based datasets and can be used to simulate a set of different drift patterns with a temporal basis. The datasets obtained are then used to evaluate and validate learning strategies used in dynamic environments, including pre-processing strategies.

To validate DOTS we also present a case study using the Twitter stream that shows DOTS can be effectively used to define datasets and test learning algorithms and techniques.

The rest of the paper is organized as follows. We start in Section 2 by presenting the DOTS framework. We then proceed in Section 3 with a Twitter stream case study to show the effectiveness of using DOTS. Finally, in Section 4 we present the main conclusions of the work and delineate some directions for future work.

2 Drift Oriented Tool System

2.1 Introduction

DOTS is a drift oriented framework developed to dynamically create datasets with drift. It is a simple-to-use freeware application with a friendly interface as shown in Fig. 2. It can be download at `http://dotspt.sourceforge.net/`.

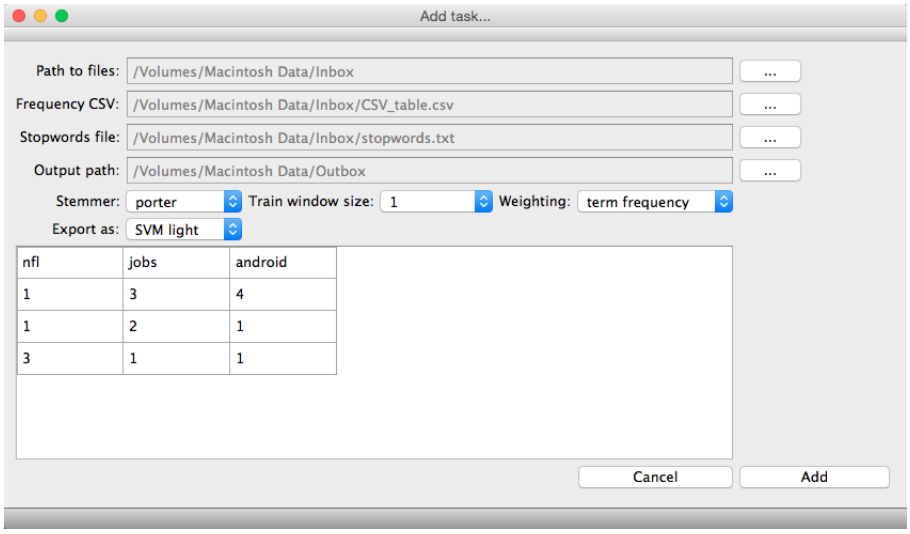

Fig. 2: DOTS interface.

The main purpose of DOTS framework is to represent drift patterns in a text-based dataset. Therefore, the framework input is a set of text document files, each representing a class, and a frequency table representing the drift patterns (see Fig. 3). The initial phase of DOTS processing includes building an INDRI index, provided by INDRI API, from the Lemur Project[4].

[4] `http://www.lemurproject.org/`

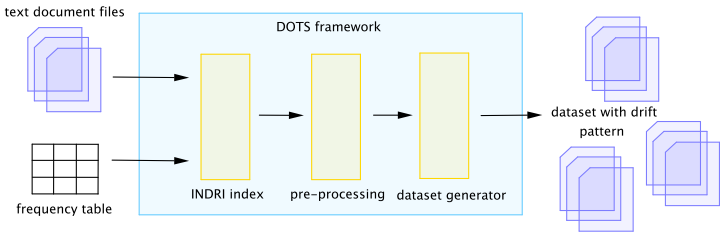

Fig. 3: The DOTS framework.

The DOTS framework represents each document in a vector space model, also known as *bag of words*, a commonly used text document representation.

Two problems arise when a vector with one element for each term occurring in the whole connection is used to represent a document: space high-dimensionality and overfitting. High dimensional space can cause computational problems and overfitting can easily prevent the classifier to generalize and thus the prediction ability can be endangered. To tackle both problems pre-processing methods were also integrated in the DOTS framework and are the second phase of the processing. Those methods are part of the INDRI API and aim at reducing the size of the document representation and prevent the mislead classification. Besides stopwords removal, DOTS also permits stemming. This pre-processing method consists in removing case and inflection information of a word, reducing it to the word stem. Stemming does not alter significantly the information included, but it does avoid feature expansion. Two important stemming algorithms were included: the Porter algorithm [8] and Krovetz algorithm [9].

It is also possible to define the weighting scheme used to represent each word of a document, that is the weight of each feature of a document. Two weighting schemes were defined, namely term frequency (*tf*) and term frequency-inverse document frequency, commonly known as *tf-idf*. Considering the defined input, DOTS will create a word index that allows users to use different strategies of filtering and analyzing data.

A major characteristic of DOTS is the possibility of defining the exact time, more precisely the exact time window, where each document appears, being thus possible to define time drifts. This is done by the frequency table that is also an input of the DOTS framework (see Fig. 3). DOTS takes no regard on the real counterpart dimension of the time window, since it constitutes an abstract realization of an amount of time (in Twitter streams it can represent seconds, but in other applications it can represent hours). The main idea is to use the frequency to reproduce artificial drifts.

DOTS output is thus a set of datasets, including the defined strategies related to vector space model, pre-processing strategies, feature representation and document division in time-windows.

Three output formats were implemented: Comma-Separated Values (CSV) file format, the Attribute-Relation File Format (ARFF) used in the widely used WEKA software, and the SVMLight file format. The possibility of using the

concept of training windows is also possible, as users can define for each time-window the previous amount of data that should also be considered. By defining different training window sizes one can represent the memory properties of the training models, because it mimics a storage mechanism.

2.2 Specific features of DOTS

The input of the DOTS framework is two fold, as seen in Fig. 3, and is composed by text documents and a frequency table. Each text document file represents the documents of the same class and the frequency table is use to define the drift patterns of the scenario. The frequency table must be in the CSV format and each row corresponds to a time instance. It is not important if a time instance represents a minute, an hour, or a day, but it is assumed that all of them correspond to the same amount of time. The first row contains the name of the class, and each cell of all the other rows contain the number of documents of a given class that occur in a given time instance. Consider Fig. 2 that represents a task to be added to the framework. By using as input a frequency table as the example given, we represent three classes : `nfl`, `jobs` and `android`, and 3 time-windows. As depicted in this example, in the first time-window there is 1 document of the class `nfl`, 3 from the class `jobs` and 4 of the class `android`.

Additional parameters can also be defined, like a stopword file and a stemmer algorithm. The stopwords file is a text file containing stopwords that will not be considered in the documents representation, and the stemmer algorithm will be used to reduce the document inflected words to their root form. Two stemmer algorithms were implemented: `porter` and `korvetz`.

It is also possible to define multiple training window sizes, multiple weighting schemes, and multiple export file formats. The training window size will define in each time-window how many previous time-windows will be considered, as this can be important for testing learning models with its memory capabilities. For instance, to perceive for how long it is relevant to keep previously gathered information and how that can affect the learning model capabilities. Weighting scheme will be used to define the document representation weighting scheme, like `term-frequency` or `tf-idf`. By exporting in multiple file formats, DOTS permits to create datasets that can be used in different classification frameworks, like `SVM Light` and `Weka`.

As it is often relevant to define various testing scenarios, DOTS will also permit adding tasks using INI files. INI files are structured files with `‘‘key=value’’` pairs, that will allow the definition of multiple tasks at once. A complete tutorial about DOTS can be download at `http://dotspt.sourceforge.net/`.

DOTS framework tries to fill an existing gap in machine learning research for text applications, by making possible to generate benchmark datasets with thoroughly controlled drift characteristics. We will now present an example of the potential use of the framework in a Twitter stream case study.

3 Case study: Twitter stream

3.1 Drift example in Twitter

To validate the proposed framework and to show how DOTS can be used, we will present a Twitter stream case study. It 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 network, 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 concept of *hashtag*. 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 [10] as *micro-meme*, i.e. an idea, behavior or style that spreads from person to person within a culture [11]. 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 [12].

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 [13], spreading an idea [14], get affiliated with a community [15], or bringing together other Internet resources [16].

3.2 Twitter classification

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 [17], 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.

3.3 Using DOTS to Twitter classification

In this particular case, DOTS is used to create datasets able to test multiple learning scenarios. DOTS receives a document set for each class of tweets

containing the same hashtag. A CSV table with different drift patterns was also defined, reproducing artificial drifts, like sudden, gradual, incremental, reoccuring and normal. As an example, a sudden drift might be represented by tweets from a given hashtag that in a given temporal moment start to appear with a significant frequency. Each tweet was represented by DOTS as a vector space model and pre-processing methods were applied, like stopword removal and stemming. We have exported, for our convenience, in SVMLight format, as is required to use Support Vector Machines (SVM) as the learning model of this case study. Two different weighting scheme in document representation were tested, term-frequency and *tf-idf*. Table 1 presents the obtained results of the performance obtained using both weighting schemes. In order to evaluate the possible outcomes of the classification, we used the van Rijsbergen F_β measure: $F_\beta = \frac{(\beta^2+1)P \times R}{\beta^2 P + R}$ with $\beta = 1$. F_1 is an harmonic average between precision and recall as is widely used in text classification problems. The results shed light on the use of term-frequency to classify Twitter messages in the presence of drift. By using term-frequency to represent document features one can improve the overall classification performance even in the presence of different drift patterns.

Drift	Hashtag	F_1 using tf	F_1 using tf-idf
Sudden	#syrisa	79.37%	75.32%
Gradual	#airasia	57.88%	56.08%
Incremental	#isis	83.49%	81.75%
Reoccurring	#android	60.66%	59.49%
Normal	#sex	74.88%	74.56%

Table 1: Performance measure for the results obtained with Twitter stream.

4 Conclusions and Future Work

In this paper we propose DOTS, a drift oriented framework developed to dynamically create text datasets with drift. This tool is specifically designed to text classification and its major goal is to create text labeled datasets that can be used to simulate different drift patterns. DOTS performs pre-processing methods like stopword removal and stemming and can be used to evaluate and validate learning strategies in dynamic environments.

We have also presented a case study based on a Twitter drift scenario. The aim of the study is to classify Twitter messages using their hashtags. We have generated multiple datasets and have tested different classification strategies, to define the best characteristic of a learning model in this particular scenario.
We have presented results that show DOTS capabilities with a Twitter scenario on testing different learning strategies in a dynamically created set of test based datasets, in which examples have a temporal meaning. DOTS is a very flexible tool as it can be used to validate learning strategies with text based datasets in different contexts rather than Twitter.

Regarding future work we will look at including new features in the DOTS framework, namely, how to automatically add noise to the datasets and new file exporting types.

Acknowledgment

We gratefully acknowledge iCIS project (CENTRO-07-ST24-FEDER - 107002003).

References

1. Q. Wu, W. Hu, B. Wang, Z. Han, and Y. Qi, "Software aging mechanism analysis and rejuvenation," *International Journal of Digital Content Technology and its Applications*, vol. 6, no. 22, p. 552, 2012.
2. 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.
3. 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.
4. 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.
5. A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, vol. 9, no. 1, pp. 56–68, 2008.
6. G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.
7. I. Zliobaite, "Learning under concept drift: an overview," Vilnius University, Faculty of Mathematics and Informatic, Tech. Rep., 2010.
8. P. Willett, "The porter stemming algorithm: then and now," *Program*, vol. 40, no. 3, pp. 219–223, 2006.
9. R. Krovetz, "Viewing morphology as an inference process," in *Proc. 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1993, pp. 191–202.
10. 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.
11. (2012, October) Merriam-webster's dictionary.
12. M. Zappavigna, "Ambient affiliation: A linguistic perspective on Twitter," *New Media & Society*, vol. 13, no. 5, pp. 788–806, 2011.
13. S. Johnson, "How Twitter will change the way we live," *Time Magazine*, vol. 173, pp. 23–32, 2009.
14. 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.
15. 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.
16. 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.
17. 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.