Information Systems 69 (2017) 40–58

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

A Two-Stage Machine learning approach for temporally-robust text classification

Thiago Salles[a,*], Leonardo Rocha[b], Fernando Mourão[b], Marcos Gonçalves[a], Felipe Viegas[a], Wagner Meira Jr.[a]

[a] Universidade Federal de Minas Gerais, Department of Computer Science, Belo Horizonte, MG, Brazil
[b] Universidade Federal de São João Del Rei, Department of Computer Science, São João Del Rei, MG, Brazil

ARTICLE INFO

Article history:
Received 23 March 2017
Revised 20 April 2017
Accepted 21 April 2017
Available online 27 April 2017

Keywords:
Automatic document classification
Temporal weighting function
Fully-Automated machine learning process

ABSTRACT

One of the most relevant research topics in Information Retrieval is Automatic Document Classification (ADC). Several ADC algorithms have been proposed in the literature. However, the majority of these algorithms assume that the underlying data distribution does not change over time. Previous work has demonstrated evidence of the negative impact of three main temporal effects in representative datasets textual datasets, reflected by variations observed over time in the class distribution, in the pairwise class similarities and in the relationships between terms and classes [1]. In order to minimize the impact of temporal effects in ADC algorithms, we have previously introduced the notion of a *temporal weighting function* (TWF), which reflects the varying nature of textual datasets. We have also proposed a procedure to derive the TWF's expression and parameters. However, the derivation of the TWF requires the running of explicit and complex statistical tests, which are very cumbersome or can not even be run in several cases. In this article, we propose a machine learning methodology to *automatically learn* the TWF without the need to perform any statistical tests. We also propose new strategies to incorporate the TWF into ADC algorithms, which we call *temporally-aware classifiers*. Experiments showed that the fully-automated temporally-aware classifiers achieved significant gains (up to 17%) when compared to their non-temporal counterparts, even outperforming some state-of-the-art algorithms (e.g., SVM) in most cases, with large reductions in execution time.

1. Introduction

Text classification is still one of the major information retrieval problems [2], due to its wide applicability in a wide variety of domains, such as document filtering, organization and retrieval. Text classifiers are the core component of many important applications such as automated topic tagging (that is, assigning labels to documents), building topic directories, identifying the writing style of a document, organizing digital libraries, improving the precision of Web searching, and even helping users to interact with search engines. Nowadays, several factors challenge the development of accurate text classifiers, due to the increasing complexity and scale of current application scenarios, such as the Web. Thus, developments in accurate classification models continue to be in great need, motivating a continuum of important advances in this matter.

Similarly to other machine learning techniques, Automatic Document Classification (ADC) usually follows a supervised learning strategy: a classification model is first "learned" using previously labeled documents (training set), and then used to classify unseen documents (the test set). The main challenge we address in this work is related to the common assumption of independent and identically distributed (i.i.d.) data: most of supervised algorithms consider that all training documents provide equally important information to discriminate classes from a new independent data (i.e., test data). However, this may not be true in practice due to several factors such as the document's timeliness, the venue in which it was published, its authors, among other factors [3].

In here, we are particularly concerned with the impact that *temporal effects* may have on ADC algorithms. Since knowledge and even languages are inherently dynamic, the characteristics of a textual dataset may change over time. For example, the relative proportion of documents belonging to different classes may change as consequence of the so-called virtual concept drift [4]. Thus, density-based classifiers, which are sensitive to class distribution, may not work well under such variations, since the "assumed"

* Corresponding author
E-mail addresses: tsalles@dcc.ufmg.br (T. Salles), lcrocha@ufsj.edu.br (L. Rocha), fhmourao@ufsj.edu.br (F. Mourão), mgoncalv@dcc.ufmg.br (M. Gonçalves), frviegas@dcc.ufmg.br (F. Viegas), meira@dcc.ufmg.br (W. Meira Jr.).

http://dx.doi.org/10.1016/j.is.2017.04.004
0306-4379/© 2017 Elsevier Ltd. All rights reserved.

class frequencies observed from the training set may not represent the “true” frequencies observed when the (unseen) test document was created [5,6]. In fact, not only the temporal variations in class frequencies may affect classification effectiveness: variations in the relationships between terms and classes may also hamper classification effectiveness [1]. That is, the distribution of terms among classes may differ over time, due to changes in writing style, term usage, and so on. Consider, for instance, the terms *pheromone* and *ant colony*. Before the 1990s, they referred exclusively to documents in the area of Natural Sciences. However, after the introduction of the technique *Ant Colony Optimization* in the area of Artificial Intelligence, these terms became relevant for classifying Computer Science documents too. In such scenarios, the classification effectiveness may deteriorate over time. Therefore, the temporal dynamics of the data at hand is an important aspect to be taken into account when learning accurate classification models.

Accordingly, our goal is to *minimize* the impact of the temporal effects in ADC algorithms. To this end, we have previously proposed what we call a *temporal weighting function* (TWF) [7–9], which was defined based on a series of statistical tests performed to determine its expression, and a curve fitting procedure to determine its parameters. The principal aspect is that the proposed TWF was defined *according the data at hand*, reflecting the dynamic nature of the data itself.

Our previous investigations showed that the TWF followed a (log-)normal distribution for two real-word datasets that suffer with the temporal effects, namely ACM-DL and MEDLINE, each characterized by a set of parameters learned from the data. Such parameters precisely defined the behavior of the underlying data variations. However, the set of statistical tests performed to define the TWF expressions for these two datasets were not able to properly define the TWF expression regarding a third dataset—AG-NEWS—which does not follow a (log-)normal distribution. Indeed, the required tests may not even be feasible to perform, as they can be prohibitively complex, depending on the dataset characteristics. This may significantly reduce the practical applicability of the proposed strategy.

Accordingly, the *main contribution* of this article is the proposal of a fully automated machine learning procedure to learn the TWF, without the need to perform such explicit statistical tests. Our hypothesis is that the variations observed in the data over time are smooth and thus can be effectively modeled by the learners. Thus, the proposed strategy can learn the temporal weights from the training data in a totally automated fashion, effectively reflecting the varying nature of the observed data, independently of their distributions. This greatly improves the applicability and generality of the proposed solutions.

It is also important to incorporate the TFW into the learning algorithms in order to improve classification effectiveness. Thus, as a *second contribution*, we also propose and thoroughly evaluate several (new) strategies to incorporate the TWF into ADC algorithms. All proposed strategies follow a lazy classification approach, and some of them include the capability of dealing of temporal skewness, as we shall see.

We evaluated our strategies using three real-world textual datasets that span for decades (ACM-DL and MEDLINE) or for several months (AG-NEWS). The fully-automated temporally-aware classifiers achieved significant improvements (up to 17%) in terms of classification effectiveness when compared to their non-temporal counterparts, even tying or outperforming the state of the art SVM classifier in some cases with a drastically reduced execution time.

The remainder of this article is organized as follows. Section 2 covers related work. Section 3 provides background on the design of the statistically-defined *temporal weighting function* (TWF) and introduces our fully-automated procedure to derive it. Section 4 describes how we introduced the TWF as an extension to the traditional classifiers. Section 5 presents our experimental results. Finally, Section 6 closes the paper.

2. Related work

In the last few decades, machine learning algorithms robust to varying data distributions have attracted the interest of the scientific community, specially in text mining tasks, such as classification[10]. was one of the first works that systematically studied the effects of varying distributions in text mining. In that work, the authors presented an in-depth characterization of varying data distributions in the textual data domain, along with a first attempt to clearly state important causes of such variations. Three main types of data variations are analyzed: *(i)* shifting class distribution, which is reflected by the observed variations over time in the proportion of documents assigned to each class; *(ii)* shifting subclass distribution, which accounts for varying feature distributions; and, finally, *(iii)* the fickle concept drift, which denotes the cases where documents are assigned to distinct classes at different moments. A real textual dataset, composed by news articles, was characterized according to the three mentioned drifting patterns, and it was shown to be a very dynamic dataset.

Following the same research line, based on discussions presented in [11], recently the work presented in [1] provided evidence of the existence of temporal effects in three textual data sets, reflected by variations observed over time in: *(i)* the class distribution, which considers the variation of the relative frequencies of the classes over the impact of temporal evolution;*(ii)* the relationships between terms and classes, which refers to changes in the representativeness of the terms with respect to the classes as time goes by; and, finally, *(iii)* the pairwise class similarities, which observes the variation over time in the term-class relationship, as well as, similarity among classes. The authors quantified, using a series of full factorial design experiments, the impact of these effects on four well-known TC algorithms. They showed that these temporal effects (negatively) affect each analyzed data set differently and that they restrict the performance of each considered TC algorithm at different extents. These reported quantitative analyses ate the basis for the some of the strategies proposed and evaluated in this paper.

Regarding attempts to *detect* significant changes in the underlying data distribution due to temporal effects, [12] proposed an online classifier to detect changes in the distribution of the training instances. Thus, this approach performs a sequence of trials to improve the classification. In each trial, it makes some predictions and receives a feedback accounting for the classification error in order to detect significant changes in the observed data. This method is able to detect both gradual and abrupt changes. Similarly, [13] presented a system composed by a set of offline and online classifiers to detect and predict data variations. Furthermore, the system also performs a clustering step to allow the prediction of future variations. Other studies explored the use of statistical tests to detect drift [14,15]. In [14], for instance, the authors proposed three adaptive tests that are capable of adapting to different (gradual or abrupt) changing behaviors. In [15], the authors proposed to classify a set of instances belonging to a recent time window, and compared the achieved accuracy against the one obtained with a global classifier that considers all available data. The main idea is that statistically significant loss of accuracy suggest data variations. This solution is able to quickly detect drifts when the window size is small, at the cost of being susceptible to data sparseness (since small windows lead to small training sets).

Previous efforts to *deal* with varying data distributions may be categorized into two broad areas, namely, adaptive document classification and concept drift. Adaptive document classification

[16] embodies a group of techniques to handle with changes in the underlying data distribution so as to improve the effectiveness of document classifiers through incremental and efficient adaptation of the classification models. Adaptive document classification brings three main challenges to document classification [17]. The first one is the definition of a context and strategies to build more accurate classification models. A context is a semantically significant set of documents, and previous research suggests that contexts may be determined through at least two strategies: the identification of neighbor terms to a certain keyword [18], and the identification of terms that indicate the scope and semantics of the document [19]. The second challenge is associated to the construction of the incremental classification models [20], whereas the third challenge is related to the efficiency of the classifiers, in terms of computational costs.

Concept or topic drift [4] comprises another relevant set of efforts to deal with varying data distributions in classification. A prevailing approach to address concept drift is to completely retrain the classifier according to a sliding window, which ultimately involves sample selection techniques. A number of previous studies fall into this category. For example, the method presented in [21] employs a window with instances sufficiently “close” to the current target concept, and automatically adjusts the window size in order to minimize the estimated generalization error. In [22], a classification model is built using training instances which are close to the test in terms of both time and space. The methods presented in [23] either maintain an adaptive time window on the training data, or select representative training instances, or weight them [24]. described a set of algorithms that react to concept drift in a flexible way and are able exploit the reappearance of contexts. The main idea of these algorithms is to keep track of only one window of currently trusted samples and hypotheses, storing the associated concept descriptions in order to reuse them if a context reappears. In [25], the authors introduced the concept of *temporal context*, defined as a subset of the training dataset that minimizes the impact of temporal effects on the classification effectiveness. They also proposed a general algorithm, named *Chronos*, to identify these contexts based on the stability of the terms in the training set. Two instantiations of this general algorithm were proposed. Temporal contexts are used to sample the training instances for the classification process, and instances considered to be outside the temporal context are discarded by the classifier.

Unlike previous efforts that used a single window to determine drift in the data, [26] introduced a method that uses three windows of different sizes to estimate changes in the data. While algorithms that use a fixed-size window impose hard constraints on drift patterns, those that use heuristics to adjust the window size to the current duration of the concept drift often require the calibration of several parameters. In order to provide some theoretical basis for the choice of the window size, [27] developed a framework for modeling the classification error as a function of the window size, aiming at determining an optimal window size choice. Such optimal choice leads to statistically significant improvements in window-based strategies. In the same direction, [28] proposed a window-based strategy for dealing with drifting data streams that automatically chooses the optimal window size, called ADWIN. This approach keeps a window W with the most recent data, and splits it into two adjacent sub-windows W_0 and W_1. Using statistical tests to compare both windows, it detects when a drift occurred. In this case, all possible adjacent sub-windows must be considered. Clearly, this operation is very expensive in terms of time and memory. In [29], the authors presented an improvement of ADWIN2, called ADWIN2. preserving the effectiveness of ADWIN and more efficient data structures.

A second approach to handle concept drift focuses on the combination of various classification models generated from different algorithms (ensembles) for classification, pruning or adapting the weights according to recent data [30–32]. proposed a boosting-like method to train a classifier ensemble from data streams. It naturally adapts to concept drift and allows one to quantify the extent of the observed variations drift of its base learners. The algorithm was shown to improve the effectiveness of learning algorithms that do not consider concept drift. In the same direction, [31] proposed a strategy that maintains an ensemble of base learners, predicts instance classes using a weighted-majority vote of these “experts”, and dynamically creates and removes experts in response to performance changes. Additionally, [32] proposed an ensemble of classifiers using genetic programming that is capable of inductively generating decision trees. In spite of these prior proposals, one important challenge for ensemble classifiers is the efficient management of multiple models, a non-trivial issue.

Finally, a third approach to deal with concept drift consists of properly weighting training instances while building the classification model in order to reflect the temporal variations in the underlying data distribution. The motivation for weighting training instances according to some time-based utility function is that window-based approaches are too rigid and may miss valuable information laying outside the window. Following this direction, [33] defined a linear time-based utility function to account for variations in the data distribution such that the impact of the instances on the classification model decreases with time. Experimental evaluation employing the algorithms Naive Bayes and ID3 demonstrated the effectiveness of such approach. In [34], the authors defined an exponential time-based function in order to weight examples based on their age. The reported experimental evaluation showed that weighting instances in drifting scenarios leads to significant improvements over fixed-window strategies, while being outperformed by an adaptive-window approach. In [35], the authors proposed an ensemble of random trees capable of handling stream data with concept drift. Such result was achieved by means of an exponential weighting function applied to each tree node to gradually reduce the influence of past data in the final classification decision. However, such a time-based utility functions are typically defined in a very ad-hoc manner (e.g., linear functions, exponential functions), without any theoretical justification based on changes in data patterns. Indeed, such a time-decaying function is ideally defined in terms of the varying behavior of the data at hand.

Therefore, the following question still needs to be answered: *how can we properly define such a time-based utility function based on the data at hand?* To properly answer such a question, one has to consider not only the temporal distance between training and test instances, but also the varying characteristics of the underlying data distribution[7]. was one of the first to deal with such an issue. In that work, the authors proposed a statistical analysis of the temporal effects on two textual datasets, towards to define a *temporal weighting function* which properly models the changing behavior of the underlying data distribution, thus reflecting its varying nature. Such time-based function relates the temporal distance between training and test instances to the degree of variation observed in the characteristics of the dataset. Two instance weighting strategies that employ the temporal weighting function to deal with these variations were developed and applied to three well known textual classifiers, namely, Rocchio, KNN and Naive Bayes. Experimental results showed that these so-called temporally-aware classifiers improve significantly the classification effectiveness over their traditional counterparts.

As discussed in [7], the “ideal” definition of the Temporal Weighting Function (TWF) demands some statistical procedures that may not be suitable to perform in practice, due to the high complexity of the tests that may be needed to define its expression. Moreover, approaches that somehow assume some structural properties of the observed data, such as the TWF definition based on dominance [8] or the oversampling strategy proposed in [9], lack the generality of statistically sound definitions. On the other hand, such statistical procedures may not be applicable in all cases. For example, the widely used procedures for independence and normality tests of random variables failed when applied to the one of our datasets (AG-NEWS), since its temporal weighting function does not follow a Gaussian process, even in the log-transformed space. Furthermore, automatic data-mining processes focused on classification may need some fully-automated ways to determine the temporal weighting function, which can not be effectively achieved if one depends of interactive statistical analysis. As a matter of fact, for the sake of temporally-aware classification, one just needs to know the positive real valued weights associated with each temporal distance.

Attempting to guarantee the practical applicability of TWF-based temporally-aware learners, without losing generality, we here propose a technique to automatically quantify the TWF, without performing any statistical test and relying on mild assumptions on the data. Our goal in this article is to develop a function which, given a set of pre-classified documents, returns a $TWF_{\text{EST}} : \delta \mapsto [0, 1]$ that ultimately models the underlying data variations. As mentioned before, our assumption is that the variations observed over time are smooth and thus can be effectively modeled by the learners themselves. As we shall see, such assumptions hold for our explored real world datasets, providing results comparable to those obtained by using the statistically-defined TWF, with a negligible computational cost.

3. The Temporal Weighting Function (TWF)

Recall that our main goal is to *minimize* the impact that temporal effects may have on ADC algorithms. Our work hypothesis is that the varying behavior of textual data, reflected by the mentioned temporal effects, may violate the common assumption of stationary data distributions, limiting the performance of textual classifiers. As previously described, the class distribution variation relates to the observed variations over time in the representativeness of classes, whereas the term distribution variation and the class similarities variation effects relate to the observed variations over time in the term-class relationships and to variations in the pairwise class similarities, respectively. As we shall detail in the upcoming sections, we address such challenges by means of two general strategies: the class distribution variation effect is tackled in a finer grained basis, at document level. The remaining two effects are handled in a coarser grained basis, at dataset level.

In order to incorporate temporal awareness to document classifiers, we have previously introduced a weighting function that we call *temporal weighting function*—or simply TWF—aiming at addressing the previously discussed temporal effects [7–9]. Such weighting function was modeled according to the observed evolution of the term-class relationships over time. That is, it considers the varying behavior of the analyzed data. This aspect is key to avoid the definition of ad-hoc approaches, which considers the temporal distance between training and test examples, without taking into account the actual varying regime of the data. Briefly, the TWF quantifies the influence of a training document while classifying a test document, according to the observed variations of the dataset, as a function of the temporal distance between their creation times. The greater the difference between the data distributions between the training and test creation time, the lesser is the influence of such training example into the classification model. In the following, we provide background on the statistical based approach, which is capable of uncovering the properties of the unknown phenomena that governs the data variation. Next, we proceed to describe the proposed fully-automated approach to devise the temporal weights, the main contribution of this article.

3.1. Background: The statistically-defined TWF (SD-TWF)

We distinguish two major steps required to define the TWF: its expression and its parameters. Expression is usually harder to determine, since it may express the generative process behind the function, that is, it may express the fundamental properties of the data variation phenomena—which can be smooth (possibly following some linear nature), abrupt (maybe with some exponential behavior) or even periodic—while the parameters are usually obtained using approximation strategies.

Intuitively, given a test document to be classified, the TWF must set higher weights to training documents that are more similar to that test document with relation to the strength of term-class relationships. Conversely, it must set smaller weights to training documents belonging to time points whose data distribution had substantially differed from the data distribution observed when the test example was created. Such smaller weights would ultimately minimize the effect of those training examples in a weighted classification model. One metric that expresses the term-class relationship strength is the *dominance*, as presented in [11], since the more exclusive a term is to a given predefined class, the stronger is this relationship. Formally, let $T = \{t_1, t_2, t_3, \ldots, t_N\}$ be the set of terms associated with a collection; $C = \{c_1, c_2, \ldots, c_K\}$ be the set of categories (classes) that occur in a collection; $df(t_i, c_j)$ be the number of documents in the training set associated with class c_j that contain t_i. We define the Dominance of the class c_j on term t_i as:

$$Dominance(t_i, c_j) = \frac{df(t_i, c_j)}{\sum_{l=1}^{K} df(t_i, c_l)}$$

The simplest approach to model the function that governs such variations would be to use a unit pulse function $\sqcap(\delta)$ at temporal distance 0,

$$\sqcap(\delta) = \begin{cases} \alpha & \text{if } \delta = 0, \\ 0 & \text{if } \delta \neq 0, \end{cases}$$

with the pulse magnitude α proportional to the observed term dominance associated with the training documents created in the same point in time of the test document. However, considering a larger time interval instead of a single point in time is preferable, since it better handles smooth data variations and does not discard potentially useful information regarding stable terms. We then need to determine the time period that must be considered when modeling the underlying data variations, and this can be accomplished by the notion of stability period, defined below.

The stability period $\mathbb{S}_{t,p_r}$ of a term t, considering the reference point in time p_r, in which the test document was created, consists of the largest continuous period of time, starting from p_r and growing both to the past and the future, where $Dominance(t, c) > \alpha$ (for some predefined α and any class c). In the case of the explored datasets, we investigated different values for α when computing stability periods and, as they lead to similar results, we adopted $\alpha = 50\%$, ensuring that the terms will have a high degree of exclusivity with some class.

Notice that the stability period of a term depends on a reference point in time and thus a term may present different stability periods, one for each point in time in which it occurred in the dataset. We first determine the stability period for each term and then combine them, as follows. In order to handle such situation, we mapped all the time points in a stability period to temporal distances, where the reference year is considered as distance 0. For instance, a term t_1 may have different stability periods when considering the years 1989 or 2000 as a reference. More specifically, if

Table 1
Adopted class identifiers for each reference dataset.

ACM-DL		MEDLINE		AG-NEWS	
0.	General Literature	0.	Aids	0.	Business
1.	Hardware	1.	Bioethics	1.	Science & Technology
2.	Computer Systems Organization	2.	Cancer	2.	Entertainment
3.	Software	3.	Complementary Medicine	3.	Sports
4.	Data	4.	History	4.	United States
5.	Theory of Computation	5.	Space Life	5.	World
6.	Mathematics of Computing	6.	Toxicology	6.	Health
7.	Information Systems			7.	Top News
8.	Computing Methodologies			8.	Europe
9.	Computer Applications			9.	Italia
10.	Computing Milieux			10.	Top Stories

the stability period of t_1 is {1999,2000,2001} regarding $p_r = 2000$, and {1988,1989,1990} regarding $p_r = 1989$, these periods would be both mapped to {−1,0,1}. Considering $\mathbb{S}'_t$ as the set of temporal distances that occur on the stability periods of term t (considering all reference moments p_r), then $\mathbb{S}'_t = \{\delta \leftarrow p_n - p_r | \forall p_r \in \mathbb{P} \text{ and } p_n \in \mathbb{S}_{t,p_r}\}$. Making the stability periods easily comparable is important because our real interest is to know what kind of distribution the temporal distances follow with respect to different terms.

The next step is to determine the function expression and, towards this goal, we considered the stability period of each term as a random variable (RV), where the occurrence of each possible temporal distance in its stability period is an event. More formally, as Table 3 shows, we are interested in the frequencies of the temporal distances δ_1 to δ_n, for terms t_1 to t_k. An interesting property that we may test is whether these RV's are independent. This hypothesis can be corroborated by the Fisher's Exact Test [36] to assess the independence of each RV_i and RV_j, $\forall i \neq j$, where, as mentioned, each RV represents the occurrence of a temporal distance δ for a term t.

3.1.1. Uncovering the SD-TWF

The three reference datasets considered to illsutrate the statistical derivation of the TWF in our study consist of sets of textual documents, each one assigned to a single class (a single label problem). For clarity purposes, throughout this paper we refer to each class by a corresponding identifier, as listed, for each dataset, in Table 1. The considered datasets are:

ACM-DL: a subset of the ACM Digital Library with 24.897 documents containing articles related to Computer Science created between 1980 and 2002. We considered only the first level of the taxonomy adopted by ACM, including 11 classes, which remained the same throughout the period of analysis. The distribution of the 24.897 documents among the 11 classes, in the entire time period, is presented in Fig. 1a.

MEDLINE: a derived subset of the MEDLINE dataset, with 861.454 documents classified into 7 distinct classes related to Medicine, and created between the years of 1970 and 1985. The class distribution of the 861.454 documents during the entire time period is depicted in Fig. 1b.

AG-NEWS: a collection of 835.795 news articles, classified into 11 distinct classes, that spans over 573 days. This dataset presents some interesting characteristics that are typical of news datasets. For instance, some topics appear and disappear very suddenly due to periodical or ephemeral events. Moreover, there is a higher variability in the meaning of the terms, along with a greater extent of class imbalance, due to the very dynamic nature of the news domain. The class distribution, spanning the whole 573 day period, is shown in Fig. 1c.

These datasets potentially present distinct evolution patterns, due to their own characteristics. In particular, we expect that MEDLINE exhibits a more stable behavior, in comparison to the other two datasets, since it represents a more consolidated knowledge area. Thus, we expect a tendency of newly inserted terms becoming stable along the years. In contrast, we expect a higher dynamism in AG-NEWS, a natural behavior of news datasets, which tend to present higher variability in their characteristics (for example, variations in class distributions according to transient events, hot topics, and so on).

Table 2
D'Agostino's D-Statistic Test of normality. Bold-face for tests that we can not reject the null hypothesis of normality.

Data	ACM-DL	MEDLINE
Original	$4.497e^{-6}$	0.002762
Log-Transformed	**0.2144**	**0.6802**

We start by assessing the independence of the RVs, by applying the Fisher's Exact Test. For the first two datasets, ACM-DL and MEDLINE, we obtained a p-value of 0.99 through a Monte Carlo simulation, which allows us to state that their associated random variables are indeed independent. Thus, the observed variability of occurrences of δ for different terms is a result of independent effects [37]. For the AG-NEWS dataset, this independence does not hold, as indicated by the low p-value obtained (10^{-4}), and some other hypothesis should be tested. This highlights the difficulty faced when defining the function that best models the varying behavior of the data at hand, motivating the development of a fully-automated strategy that overcomes the need to explicitly determine the TWF expression. As a matter of fact, one can afford to avoid the explicit determination of the TWF expression and parameters since, for the sake of temporally-aware ADC, just the TWF image matters (that is, the weights associated with each temporal distance). Clearly, avoiding to determine the TWF expression and parameters comes at the cost of missing the opportunity to discover the TWF's properties—which is revealed when determining its expression and parameters. With this trade-off in mind, in next Section, we describe a fully-automated strategy to determine the weights of each temporal distance.

Turning our attention to the ACM-DL and MEDLINE datasets (that passed in the independence tests), it is still not clear whether the effects responsible for the observed variability in the temporal distance distribution D_δ can be additive (leading to a normal distribution) or multiplicative (leading to a lognormal distribution). In Fig. 2 we show the D_δ distribution, scaled to the [0, 1] interval. We then apply a statistical normality test to both the original and log-transformed distribution. According to D'Agostino's D-Statistic Test of Normality [38], with 99% confidence, we found that the lognormal distribution best fits both the ACM-DL and MEDLINE collections, as presented in Table 2.

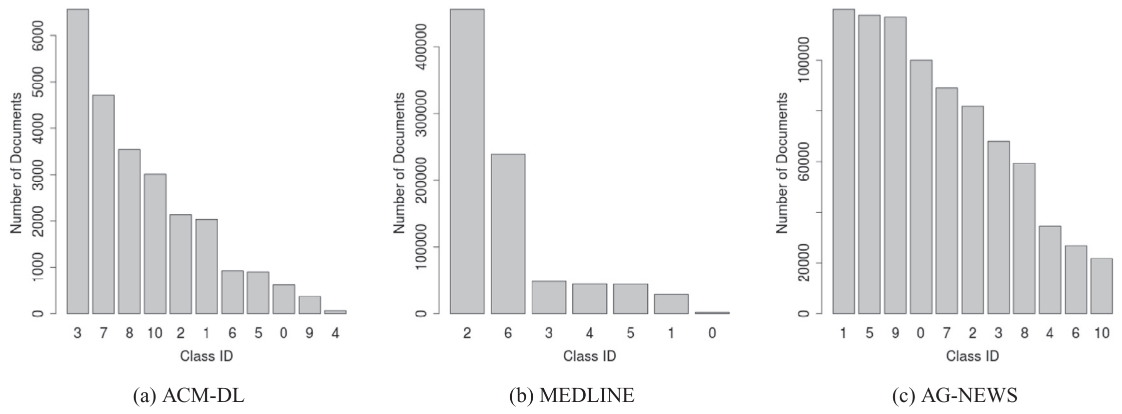


Fig. 1. Class Distributions in the Three Reference Datasets.

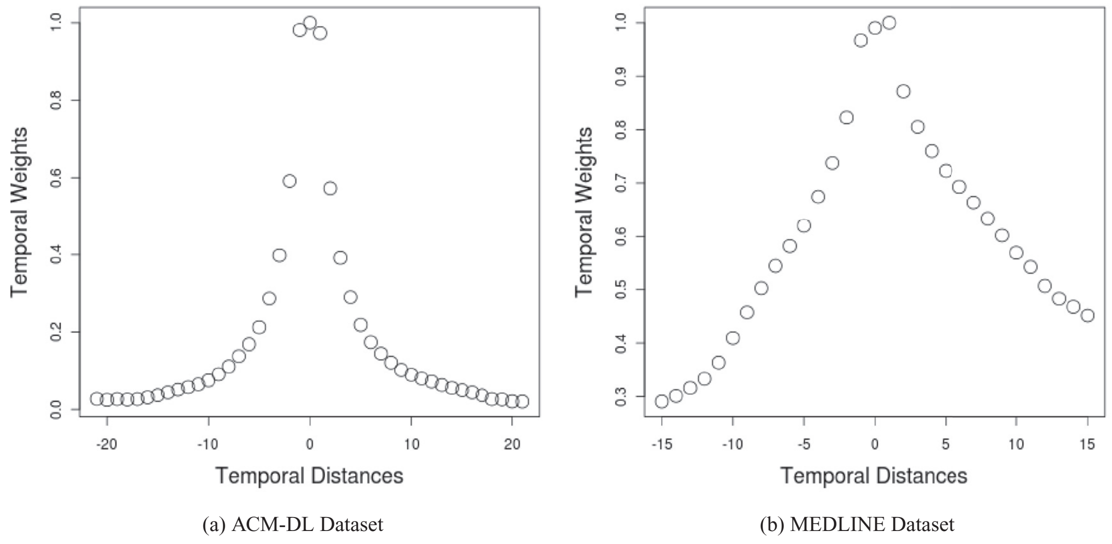


Fig. 2. D_δ Distribution (Scaled to [0, 1] Interval).

Table 3
Temporal Distances *versus* Terms.

	t_1	t_2	...	t_k	D_δ
δ_1	f_{11}	f_{12}	...	f_{1k}	$\sum_{i=1}^{k} f_{1i}$
δ_2	f_{21}	f_{22}	...	f_{2k}	$\sum_{i=1}^{k} f_{2i}$
$\vdots$					
δ_n	f_{n1}	f_{n2}	...	f_{nk}	$\sum_{i=1}^{k} f_{ni}$

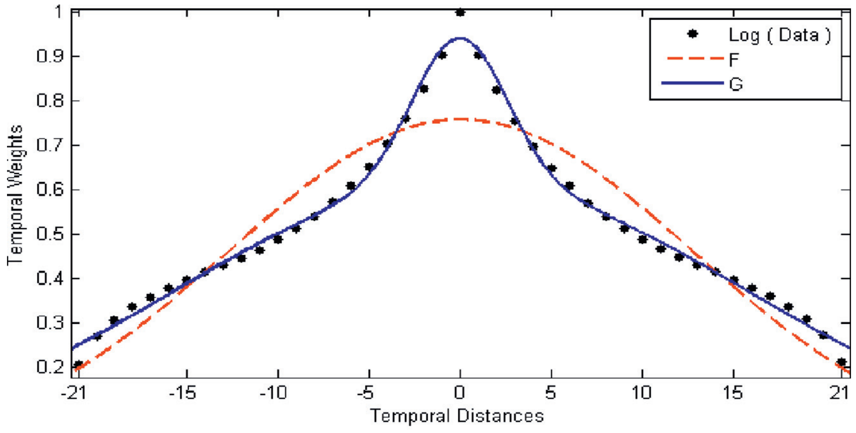
Table 4
Estimated parameters for both datasets, with 99% confidence intervals.

Parameters	ACM-DL		MEDLINE	
	Value	Confidence Interval	Value	Confidence Interval
a_1	0.325	(0.288, 0.362)	0.089	(0.066, 0.113)
b_1	−0.028	(−0.309, 0.253)	−0.013	(−0.349, 0.324)
c_1	3.636	(3.117, 4.154)	1.635	(1.099, 2.17)
a_2	0.616	(0.589, 0.643)	0.901	(0.891, 0.911)
b_2	0.037	(−0.395, 0.470)	0.092	(−0.130, 0.314)
c_2	20.14	(20.93, 23.35)	24.51	(23.71, 25.3)
R^2		0.990		0.992

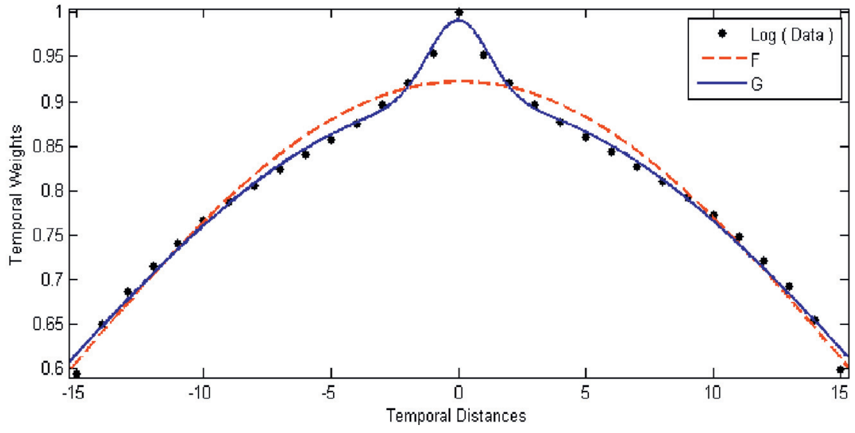
Consider that the distribution D_δ related to the occurrences of the temporal distances δ in the stability periods, which represents the distribution of each δ_i over all terms t, is lognormally distributed if lnD_δ is normally distributed. More generally, since the temporal distances δ_i are RV's under the independence assumption with finite mean and variance, then, by the Central Limit Theorem, $lnD_\delta = \sum_{i=1}^{n} \ln \delta_i$ will asymptotically approach a normal distribution and, by definition, D_δ converges to a lognormal distribution [39]. For a lognormal distribution, the asymptotically most efficient method for estimating its associated parameters relies on a log-transformation [37]. Using a Maximum Likelihood method, we estimated those parameters for both collections, and then back-transformed them, as shown in Table 4. We considered a 3-parameter Gaussian function,

$$F = a_i e^{-\frac{(x-b_i)^2}{2c_i^2}},$$

where the parameter a_i is the height of the curve's peak, b_i is the position of the center of the peak, and c_i controls the width of the curve. The last one, also called the shape parameter, reflects the nature of the variations of term-class relationships over time. Since abrupt or smooth variations lead to small or greater stability periods, respectively, the shape of the distribution changes accordingly, being a matter of parameter estimation to capture such dis-



(a) ACM-DL Dataset



(b) MEDLINE Dataset

Fig. 3. Fitted Temporal Weighting Function with Log-Transformed Data.

tinct natures. We performed two curve fitting procedures, considering a single Gaussian F and a mixture of two Gaussian, given by $G = G_1 + G_2$, where each G_i denotes a Gaussian function. The last one was the model that best fitted D_δ, and its parameters are presented in Table 4, along with the goodness of fitting measure R^2. The R^2 measure denotes the percentage of variance explained by the model and, for both collections, the obtained model explains 99% of such variance.

The greater the frequency of δ on stability periods, the more suitable training documents created in δ are to build an accurate classification model, making the modeling of the Temporal Weighting Function as a lognormal distribution an effective strategy.

Fig. 3 shows the distribution of temporal scores for each possible temporal distance between the creation time of test document d' and the training documents for the ACM-DL and the MEDLINE datasets.

3.2. The proposed fully-Automated TWF (FA-TWF)

Clearly, to determine the expression and parameters of a function that effectively models the underlying data variations is an important task, since it reveals the properties of the data variations and offer substantial knowledge that can be exploited towards the development of accurate classification models. However, to define the TWF function demands some statistical procedures that may not be suitable for a practitioner to perform, due to the diversity and sophistication of the tests that may be needed to define its expression. As discussed before, the most straightforward procedures for independence testing of random variables failed when applied to the AG-NEWS dataset. Thus, unlike the TWF's associated with the ACM-DL and MEDLINE datasets, AG-NEWS' TWF does not follow a Gaussian process and some other (possibly more complex) tests should be performed to assess its expression. However, it may be prohibitively hard for a practitioner, hurting the practical applicability of the proposed framework to devise the TWF and, consequently, the applicability of the temporally-aware classifiers described in Section 4. Furthermore, automatic data-mining processes focused on classification may need automatic ways to determine the TWF. As a matter of fact, for the sake of temporally-aware ADC, one just needs to know the positive real valued weights associated with each temporal distance. While the proposed statistical framework is able to uncover the properties of the function that underlies the data variations, it may not be applicable for the two mentioned scenarios, and strategies to overcome this issue are desirable.

To cover such practical scenarios, in this section we describe a technique to automatically determine the TWF, without the needs to perform any statistical test. Hence, we describe a straightforward and suitable way to devise the TWF by a practitioner or by some other automated data-mining process. Our goal is thus to develop a procedure which, given a set of already classified documents, outputs a function $TWF_{\text{EST}} : \delta \mapsto [0, 1]$ that ultimately models the underlying data variations. More specifically, the ADC algorithms themselves are used to devise such mapping.

Let $\mathbb{D}$ be the training set composed by already classified documents $d_i = (\langle \overrightarrow{x_i}, p_i \rangle, c_i)$, where $\overrightarrow{x_i}$ is the vectorial (bag of words) representation of d_i, p_i denotes its creation point in time and c_i denotes its associated class. The first step of our procedure consists of changing the associated class of each document to its creation point in time, that is, we represent d_i as $d'_i = (\overrightarrow{x_i}, p_i)$. Then, the training set is randomly partitioned into two subsets, $\mathbb{D}_t$ and $\mathbb{D}_v$, and a classification procedure is performed, using $\mathbb{D}_t$ as a training set and $\mathbb{D}_v$ as a validation set. Our basic assumption is that, due to the temporal effects previously described, the underlying data distribution may be different for each point in time p_i, and so we expect that the classifier may be able to learn some structural properties observed in each of them. Clearly, the classifier will not achieve high accuracy, since, as discussed earlier, the variations observed due to the temporal effects are typically smooth and hence it is unlikely that the observed changes produce enough variations to enable discriminating data between each point in time. However, the classifier will potentially predict nearby points in time, since data from nearby points in time tend to have similar underlying distributions.

Formally, an ADC algorithm is used to learn the underlying relationships between the data and their creation points in time, expressed by the a posteriori probability distribution $P(p_i|d_i)$. Thus, if documents created at a point in time p_i share some structural properties with data from a reference point in time p_r (namely, when documents from $\mathbb{D}_v$ were created), then p_i will receive a higher score than some other uncorrelated point in time $p_j \neq p_i$. Since we aim at associating a real valued weight to the temporal distances $\delta_i = p_i - p_r$, we adopt the following rule to devise the TWF:

$$TWF(\delta_i) = \frac{\sum_{j=1}^{N} I(p_j - p_r = \delta_i)}{N},$$

where $I(\bullet)$ is an indicator function which returns 1 if the predicate $\bullet$ is true and returns 0 otherwise, p_r is the actual creation point in time of the classified documents from $\mathbb{D}_v$ (that is, the reference point in time), p_j is the predicted point in time (which received the highest score by the classifier) and $N = |\mathbb{D}_v|$ denotes the number of documents classified. Intuitively, temporal distances with higher weights contain the most useful documents to build the classification model, since they provide data sampled from similar distributions to the ones that govern the test data. On the other hand, temporal distances with smaller weights tend to have more unstable data, which may induce the classifier to misleading predictions. The described procedure to automatically determine the TWF is listed in Algorithm 1.

Algorithm 1 Automatic TWF Determination.

```
1: function LEARNTWF($\mathbb{D}$)
2:     $\mathbb{D}' \leftarrow \{d'_i | d'_i = (d_i.x_i, d_i.p_i) \wedge d_i \in \mathbb{D}\}$
3:     $(\mathbb{D}_t, \mathbb{D}_v) \leftarrow$ RANDOMSPLIT($\mathbb{D}'$)
4:     $p[\ ] \leftarrow$ CLASSIFY($\mathbb{D}_t, \mathbb{D}_v$)
5:     $TWF(\delta_i) = \frac{\sum_{j=1}^{N} I(p_j - d'_j.p_j = \delta_i)}{N}$, where $d'_j \in \mathbb{D}_v$
6:     return $TWF$
7: end function
```

We show in Fig. 4 the FA-TWFs for each explored textual datasets, estimated by each explored classifier. They were obtained during the 10-Fold Cross Validation procedure to assess the effectiveness of the temporally-aware algorithms, as reported in Section 5. The reader should notice the similarities among the TWF's learned by each classifier. In fact, it does not matter which classifier is employed in line 4 of Algorithm 1, as we shall discuss in Section 5.

4. Temporally-aware algorithms for automatic document classification

The final step of our solution is to incorporate the TWF into the ADC algorithms. We propose three weighting strategies for doing so. The first two were proposed along with the statistically-defined TWF (SD-TWF) [7–9], but have not been tested with the fully-automated proposal, while the third strategy, which accounts for the *temporal skewness*, is a novel contribution of this work.

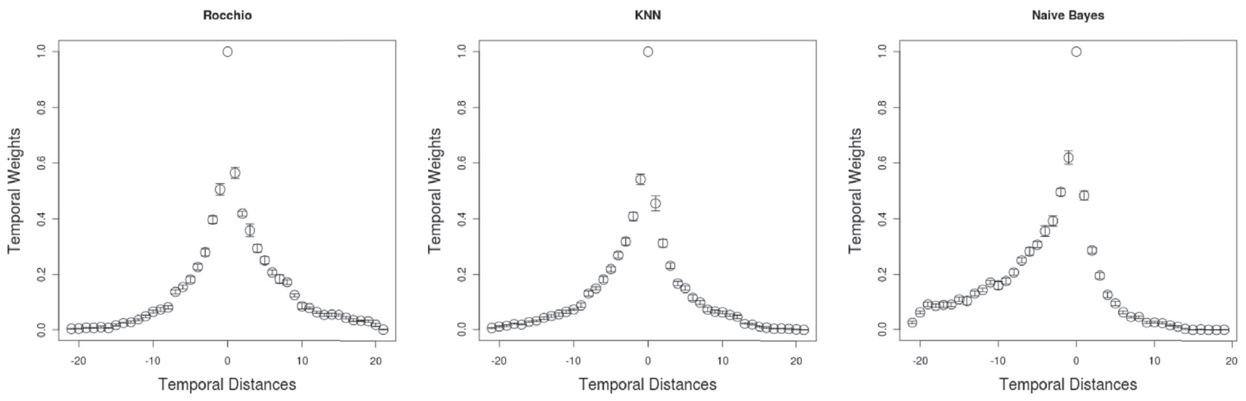
In these strategies, the weights assigned to each example depend on the notion of a temporal distance δ. The temporal distance is defined as the difference between a point in time p, in which a training example was created, and a reference point in time p_r, in which the test example was created. Such weights reflect the observed variability in the data distribution, as captured by the TWF. The first strategy, named *temporal weighting in documents*, weights training instances according to δ, ultimately addressing the explored term distribution variation (*TD*) effect—since the TWF considers the observed variations over time in the term-class relationships.

However the class distribution variation (*CD*) and the class similarity variation (*CS*) effects also play an important role for some datasets, and we should minimize their impact. Towards this end, we propose a second strategy, called *temporal weighting in scores*, which is based on a mapping in the class domain $c \mapsto \langle c, p \rangle$, in which the training documents' classes are mapped to derived classes $\langle c, p \rangle$ where c denotes the actual document's class and p denotes its creation point in time. In this case, the scores (for example, similarities, probabilities) learned by a traditional classifier applied to the modified training set are weighted according to $\delta = p - p_r$, where p is the point in time associated to the derived class $\langle c, p \rangle$ and p_r is the point in time in which the test document was created—that is, $score_c = \sum_p score_{c,p}(c, p)\,\mathrm{TWF}_\delta$. The combined weighted scores for each class are then used to take the final classification decision. This strategy is motivated by the fact that, when considering each point in time in isolation we ultimately isolate the temporal effects. But, since it is not always plausible to consider just the documents created in the reference point in time p_r (due to data scarcity), we aggregate the obtained scores for each point in time, using the TWF to account for the variations in the term-class relationships (the only potential effect reflected when aggregating the intermediate scores). However, this strategy has a somewhat undesirable property related to the mapping $c \mapsto \langle c, p \rangle$. As the number of documents from the derived class $\langle c, p \rangle$ is typically much smaller than the number of documents belonging to class c, the class imbalance artificially increases. Since class imbalance may be harmful for document classifiers [40], we propose a third strategy aimed at ameliorating this, namely, the *extended temporal weighting in scores*. In this strategy the training set $\mathbb{D}$ is partitioned in sub-groups of documents $\mathbb{D}_p$ with the same creation point in time p. Then, a classification model is built based on each $\mathbb{D}_p$ in isolation. The classes' scores are then produced for each $\mathbb{D}_p$ and, as before, they are aggregated using the TWF to weight them. By construction, the class imbalance problem is bounded by the imbalance observed in the class distribution observed in $\mathbb{D}_p$, which is usually smaller. We detail these solutions next.

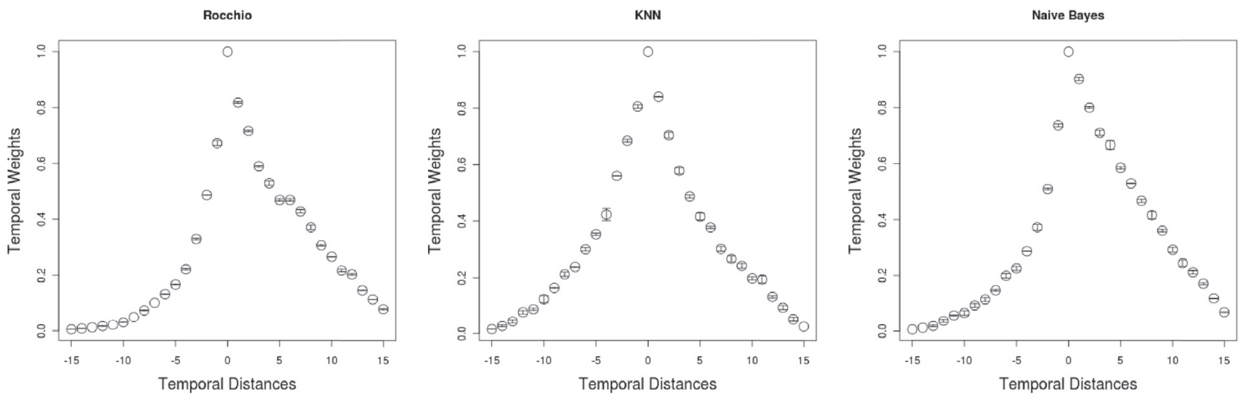
4.1. Temporal weighting in documents

The temporal weighting in documents strategy weights each training document by the temporal weighting function according to its temporal distance to the test document d', as represented in Fig. 5.

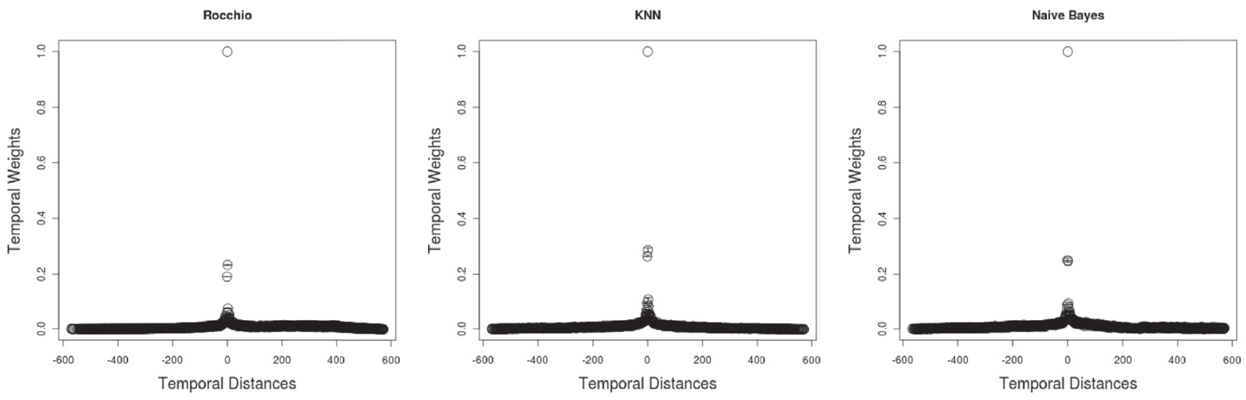
The strategy to incorporate the weight of each training document to a given classifier depends inherently on the characteristics of the classification algorithm being modified. In the case of



(a) ACM-DL Dataset



(b) MEDLINE Dataset



(c) AG-NEWS Dataset

Fig. 4. Estimated Temporal Weighting Function.

distance-based classifiers, the temporal weighting function can be easily applied when calculating the distance between the training and test documents, by weighting each training document (vectorial representation) by its associated temporal weight. In the case of the Naive Bayes, the temporal function can be used to weight the impact of each training example in both the a priori and conditional probabilities estimation (that is, to weight its impact on the counts), in order to generate a more accurate a posteriori probability.

The Rocchio classifier [41] uses the centroid of a class to find boundaries between classes. As an eager classifier, which uses the training set to build a single model that is used to classify all test set, Rocchio does not require any information from a test document d' to create a classification model. When classifying a new document d', Rocchio associates it to the class represented by the centroid closest to d', where the centroid is given by the vector average of the training documents belonging to that class.

We thus adapt Rocchio to become a lazy classifier in order to use the temporal weighting function, since the weights depends on the creation point in time of a test document. We explicitly change the separation boundaries of classes according to the temporal weights produced by the TWF function, that is, the class centroids are calculated according to the creation point in time p_r of a test document d'.

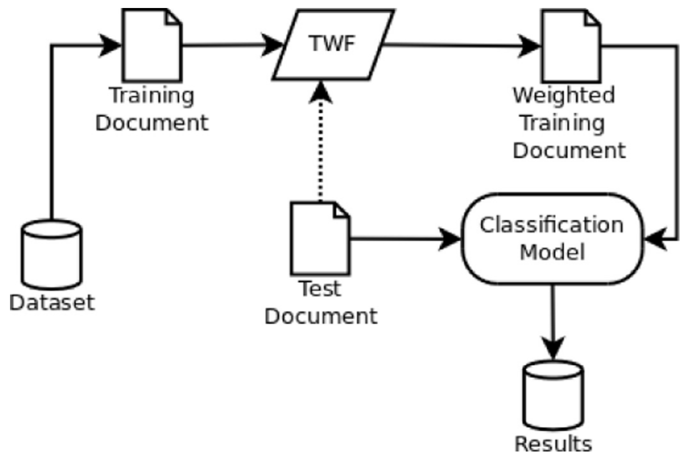


Fig. 5. Graphical Representation of TWF in documents.

Consider the set $\mathbb{D}_c \subseteq \mathbb{D}$ of training documents that belong to the class c. This set can be partitioned into subgroups $\mathbb{D}_{c,p} \subseteq \mathbb{D}_c$ of documents created at the same point in time $p \in \mathbb{P}$. The centroid $\vec{\mu}_c$ for class c is thus defined by weighting the documents vector representations with the score produced by the temporal function $TWF(\delta)$, obtained using the temporal distance δ between the creation point in time of $d \in \mathbb{D}_{c,p}$ and d', for all $p \in \mathbb{P}$. Thus, a centroid $\vec{\mu}_c$ is given by:

$$\vec{\mu}_c = \frac{1}{\|D_c\|}\left(\sum_{\vec{d}_p \in \mathbb{D}_{c,p}} \vec{d}_p \cdot TWF(\delta)\right),$$

where $\|D_c\|$ is the number of documents in class c, $\mathbb{D}_{c,p}$ is the set of documents belonging to class c that were created at point in time p and δ is the temporal distance between $\vec{d}_p$ and the test document d'.

This approach redefines the centroid's coordinates in the vectorial space considering document's representativeness on class c w.r.t. the reference point in time p_r. Both training and classification procedures are presented in Algorithm 2.

Algorithm 2 Rocchio-TWF-Doc: Rocchio with Temporal Weighting in Documents.

```
 1: function Train($\mathbb{C}$, $\mathbb{D}$, $d'$, $TWF$)
 2:     for each $c \in \mathbb{C}$ do
 3:         $\vec{\mu}_c \leftarrow \frac{1}{\|D_c\|}\left(\sum_{\vec{d}_p \in \mathbb{D}_{c,p}} \vec{d}_p \cdot TWF(p - d'.p)\right)$
 4:     end for
 5:     return $\{\vec{\mu}_c : c \in \mathbb{C}\}$
 6: end function
 7: function Classify($\mathbb{D}$, $\mathbb{C}$, $d'$)
 8:     $\{\vec{\mu}_c : c \in \mathbb{C}\} \leftarrow$Train($\mathbb{D}$, $\mathbb{C}$, $d'$)
 9:     return $\arg\max_c \cos(\vec{\mu}_c, \vec{d'})$
10: end function
```

KNN [42] is a lazy classifier that assigns to a test document d' the majority class among those of its k nearest neighbor documents in the vector space. Determining the test document's class from the k nearest neighbors training documents may not be ideal in the presence of term-class relationships that vary considerably over time. To deal with it, we apply the proposed temporal weighting function during the computation of similarities among d' and the documents in the training set, aiming to select the closest documents, in terms of both similarity and timeliness.

Let s be the cosine similarity between a training document d and d'. If d is similar to d' but is temporally distant (in terms of dataset characteristics between both points in time), then it is moved away from d', reducing the probability of being among the k nearest documents of d'. Let $TWF(\delta)$ be the temporal weight associated with the temporal distance between the time of creation of documents d and d'. Then, the documents' similarity is given by:

$$sim(d, d') \leftarrow \cos(d, d') \cdot TWF(\delta).$$

Both training and classification procedures are presented in Algorithm 3.

Algorithm 3 KNN-TWF-Doc: KNN with Temporal Weighting in Documents.

```
 1: function KNearestNeighbors($\mathbb{D}$, $d'$, $k$, $TWF$)
 2:     for each $d \in \mathbb{D}$ do
 3:         $\delta \leftarrow d.p - d'.p$
 4:         $sim(d, d') \leftarrow \cos(d, d') \cdot TWF(\delta)$
 5:         $priorityQueue.insert(sim, d)$
 6:     end for
 7:     return $priorityQueue.first(k)$
 8: end function
 9: function Classify($\mathbb{D}$, $d'$, $k$)
10:     $knn \leftarrow$ KNearestNeighbors($\mathbb{D}$, $d'$, $k$)
11:     return $\{\arg\max_c \sum_c knn.nextDoc(c)\}$
12: end function
```

Similarly to the previously defined "temporal weighting in documents" approaches, in the Naive Bayes classifier we apply the temporal weighting function on the information used by the learning method, namely the relative frequencies of documents and terms, as follows:

$$P(d'|c) = \eta \cdot \frac{\sum_p (N_{cp} \cdot TWF(\delta))}{\sum_p (N_p \cdot TWF(\delta))} \cdot \prod_{t \in d'} \frac{\sum_p (f_{tcp} \cdot TWF(\delta))}{\sum_p \sum_{t' \in \mathbb{V}} (f_{t'cp} \cdot TWF(\delta))},$$

where η denotes a normalizing factor, N_{cp} is the number of training documents of $\mathbb{D}$ assigned to class c and created at the point in time p, N_p is the number of training documents created at the point in time p, f_{tcp} stands for the frequency of occurrence of term t in training documents of class c that were created on point in time p and, finally, δ denotes the temporal distance between p and the creation time of d' (a.k.a., the reference point in time).

The main goal of this strategy is to reduce the impact that temporally distant information have when estimating a posteriori probabilities. Algorithm 4 presents this strategy.

Algorithm 4 Naive Bayes TWF-Doc: Naive Bayes with Temporal Weighting in Documents.

```
1: function Classify($\mathbb{D}$, $d'$, $TWF$)
2:     for each $c \in \mathbb{C}$ do
3:         $aPriori[c] \leftarrow \frac{\sum_p (N_{cp} \cdot TWF(\delta))}{\sum_p (N_p \cdot TWF(\delta))}$
4:         $termCond[c] \leftarrow \prod_{t \in d'} \frac{\sum_p (f_{tcp} \cdot TWF(\delta))}{\sum_p \sum_{t' \in \mathbb{V}} (f_{t'cp} \cdot TWF(\delta))}$
5:     end for
6:     return $\{\arg\max_c \eta \cdot aPriori[c] \cdot termCond[c]\}$
7: end function
```

4.2. Temporal weighting in scores

A more sophisticated approach to exploit the temporal weighting function considers the "scores" produced by the traditional

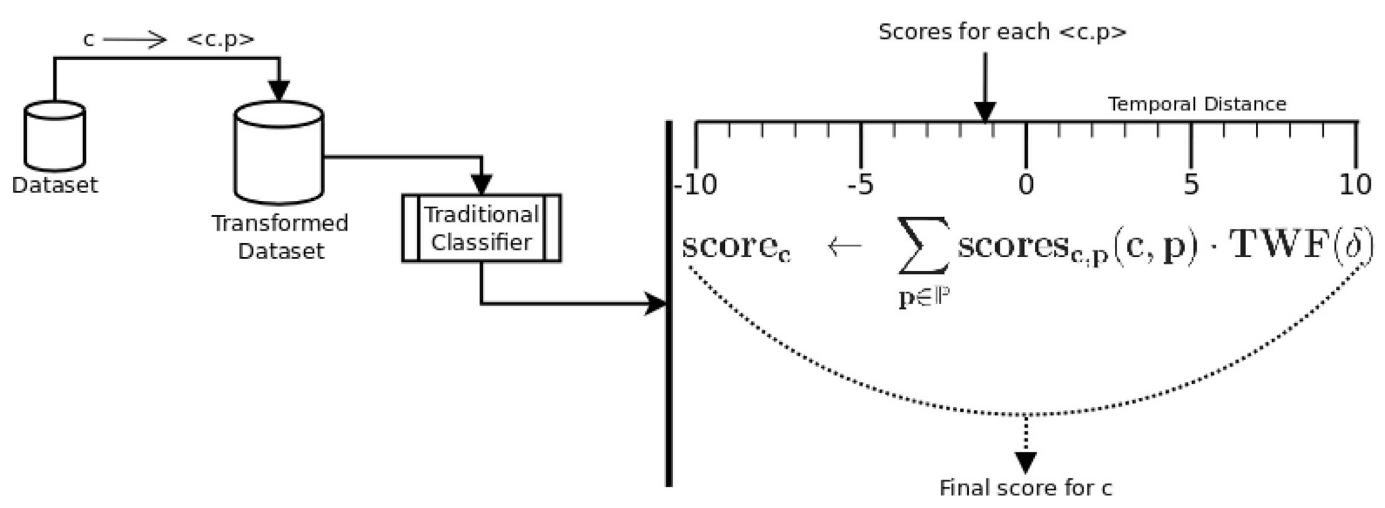


Fig. 6. Graphical representation of TWF in scores.

classifiers, as represented in Fig. 6. By score we mean: (*i*) the smallest distance from the test document d' to a class centroid for Rocchio; (*ii*) the smallest sum of the distances of the K-nearest neighbors to document d' assigned to class c in the case of KNN; or (*iii*) the probability to generate d' with the model associated to some class c for Naive Bayes. From now on, we refer to this approach as *temporal weighting in scores*.

Let $\mathbb{C}$ and $\mathbb{P}$ be the set of classes and creation points in time of the training documents. First, each training document class $c \in \mathbb{C}$ is associated with the corresponding creation point in time $p \in \mathbb{P}$, generating a new class defined as $\langle c, p\rangle \subseteq \mathbb{C} \times \mathbb{P}$. Then, we use a traditional classification algorithm to generate scores for each new class $\langle c, p\rangle$. Thus, the first step of this strategy consists of generating a new training set $\mathbb{D}_{c,p}$ with the class domain transformed from $\mathbb{C}$ to $\mathbb{C} \times \mathbb{P}$. Then, the test document d' is classified by a traditional classifier applied to this new training set, ultimately generating scores for each $\langle c, p\rangle$. Note that this scenario isolates term-class relationship variations, since it ties together the predictive relationships of the patterns observed in each class c at the point in time p, in which the patterns were observed. To decide to which class c the document d' should be assigned to, the learned scores for each $\langle c, p\rangle$ are summed up, for all $p \in \mathbb{P}$, weighting them by the *TWF*(δ), where $\delta = p - p_r$ corresponds to the temporal distance between p and the time of creation p_r of d', that is,

$$scores_{c,p} \leftarrow \textsc{TraditionalClassifier}(d', \mathbb{D}_{c,p}),$$
$$score_c \leftarrow \sum_{p\in\mathbb{P}} scores_{c,p}(c, p)\cdot TWF(\delta),$$

where $\mathbb{D}_{c,p}$ is the new set of training documents generated by mapping each document's class c to the derived class $\langle c, p\rangle$, according to its creation point in time. At the end of this process, d' will be assigned to the class c with highest score, as listed in Algorithm 5.

Algorithm 5 TWF-Sc: Temporal Weighting in Scores.

```
1: function $\textsc{Classify}(d', \mathbb{C}, \mathbb{P}, \mathbb{D}, TWF)$
2:     $\mathbb{D}_{c,p} \leftarrow \{d_{c,p} = (\overrightarrow{d.x}, \langle d.c, d.p\rangle) | d \in \mathbb{D}\}$
3:     $scores_{c,p} \leftarrow \textsc{TraditionalClassifier}(d', \mathbb{D}_{c,p})$
4:     for each $c \in \mathbb{C}$ do
5:         $\delta \leftarrow p - d'.p$
6:         $score_c \leftarrow \sum_{p\in\mathbb{P}} scores_{c,p}(c, p)\cdot TWF(\delta)$
7:     end for
8:     return $\{\arg\max_c score_c\}$
9: end function
```

4.3. Extended temporal weighting in scores

When generating the scores for the derived class $\langle c, p\rangle$ during the classification of a test document, the number of positive training documents (that is, training documents belonging to $\langle c, p\rangle$) is usually outnumbered by the number of negative training documents (that is, training documents belonging to a derived class $\langle c, p\rangle' \neq \langle c, p\rangle$). This is known as the class imbalance problem, and is an issue for classifiers with some bias towards the majority classes. Indeed, this is a problem inherent to the classification task and the majority of automatic classifiers is affected by this issue. The "*in scores*" strategy becomes vulnerable to the class imbalance problem since it artificially increases the imbalance when mapping the classes to $\langle c, p\rangle$. Several works have already proposed strategies to minimize such problem, for example, strategies to under-sample the majority classes [43] or over-sample the minority classes [40]. We address this issue by modifying the "*in scores*" strategy in order to minimize the class imbalance problem.

More formally, let $\mathbb{D}_c$ denote the set of documents belonging to class c and $\mathbb{D}_c^p \subseteq \mathbb{D}_c$ denote the set of documents created at the point in time p that also belongs to the class c. Clearly, $|\mathbb{D}_c^p| \leq |\mathbb{D}_c|$. Now, consider our previously proposed "*in scores*" strategy, in which a classifier is used to learn the scores for $\langle c, p\rangle$. The difference between $|\mathbb{D}_c^p|$ (the number of positive documents) and $|\mathbb{D}_c \setminus \mathbb{D}_c^p|$ (the number of negative documents) is expected to be greater than the difference between $|\mathbb{D}_c|$ and $|\mathbb{D} \setminus \mathbb{D}_c|$. In other words, the number of negative documents observed in the transformed class domain ($\mathbb{C} \times \mathbb{P}$) outnumber the number of positive documents in a much more expressive way than when considering the original class domain ($\mathbb{C}$). Thus, the "*in scores*" strategy artificially increases the class imbalance when considering the derived classes $\langle c, p\rangle$, and such imbalance is greater than the observed when considering the original class distribution.

Based on this observation, the extended version of the "*in scores*" aims at mitigating the class imbalance problem by considering each point in time in isolation, as represented in Fig. 7, employing a series of classifiers to associate scores for the classes considering only documents belonging to each point in time independently, but belonging to all classes. The scores obtained by each classifier are then aggregated with the corresponding TWF weight, according to the temporal distance between the point in time associated to each classifier and the creation time of the test document. Let $\mathbb{D}_p$ denote the set of documents created at the point in time p. Since $\mathbb{D}_c^p \subseteq \mathbb{D}_p$ then $|\mathbb{D}_c^p| \leq |\mathbb{D}_p|$ and the majority class sizes observed in $\mathbb{D}_p$ is bounded by $|\mathbb{D}_p|$. In the "*in scores*" strategy, the

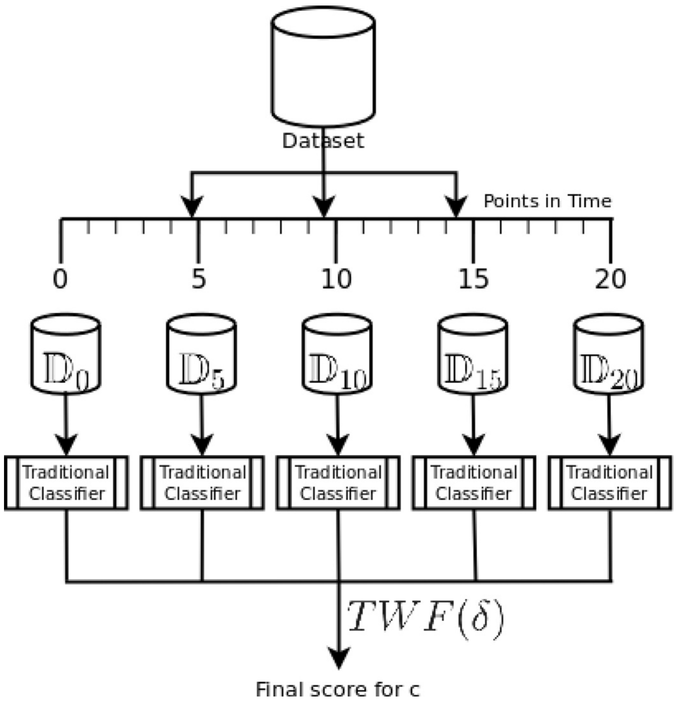


Fig. 7. Graphical representation of Extended TWF in scores.

majority class size is bounded by $|\mathbb{D}_{c,p}| = |\mathbb{D}|$. Consequently, the class imbalance observed in the first approach is smaller than the class imbalance observed in the second one.

There are two rather subtle differences between this new strategy and the previous *in scores* approach. First, as mentioned, by construction the class imbalance problem is bounded by the class imbalance observed in $\mathbb{D}_p$, since it is the set considered when training the intermediate classifiers. Second, consider the traditional classification procedure performed by both strategies. While in the "*in scores*" strategy such classification is performed considering the modified training set $\mathbb{D}_{c,p}$, in the extended version only documents belonging to $\mathbb{D}_p$ are considered, and this implies that documents created at a point in time $p' \neq p$ do not influence in the learned scores. As an example, consider the class *Top Stories* of the AG-NEWS dataset. In the 50th week, none of the documents belong to such class. Now assume that the classification procedure of both strategies will attempt to learn a score for this class when classifying a test document belonging to the 1st week. The scores learned by the classifier considering the "*in scores*" strategy will consider all training documents, including those belonging to the 50th week. Thus, these negative documents ultimately influence the learned scores. On the other hand, in the "*extended in scores*" strategy, the classification procedures applied to each point in time will act in isolation: the classifiers assigned to the points in time within the 50th week will output scores equal to zero for this class and just the classifiers assigned to points in time with documents belonging to this class will output non-zeroed scores. The *extended in scores* procedure is listed in Algorithm 6.

5. Experimental evaluation

Having presented our strategies to determine the TWF and our temporally-aware classifiers, we now report our experimental evaluation to assess the effectiveness of the temporally-aware classifiers in minimizing the impact of the temporal effects observed in the three explored textual datasets. Recall that we found that for the ACM-DL and MEDLINE datasets the TWF follows a lognormal distribution, unlike the TWF associated with the AG-NEWS dataset, whose expression is still unknown. Hence, we start by evaluating the temporal algorithms using the SD-TWF applied to the ACM-DL and MEDLINE datasets. These experiments also serve as a baseline to compare with the experiments with the FA-TWF (described in Section 3.2) under the same experimental setup and conditions. In this last case, since complex statistical tests are not necessary anymore, we are able to determine the TWF for the three textual datasets, in a fully-automated way.

Algorithm 6 TWF-Sc-Ext: Extended Temporal Weighting in Scores.

```
1: function CLASSIFY($d'$, $\mathbb{C}$, $\mathbb{P}$, $\mathbb{D}$, $TWF$)
2:     for each $p \in \mathbb{P}$ do
3:         $\mathbb{D}_p \leftarrow \{d \in \mathbb{D} | d.p = p\}$
4:         $\delta \leftarrow p - d'.p$
5:         $score_c(c)$ += TRADITIONALCLASSIFIER($d'$, $\mathbb{D}_p$) $\cdot$ $TWF(\delta)$
6:     end for
7:     return $\{\arg\max_c score_c\}$
8: end function
```

In order to evaluate the impact that the proposed TWF has on the classification task, we compare both the traditional and temporally-aware versions of Rocchio, KNN and Naive Bayes in the three adopted datasets (ACM-DL, MEDLINE and AG-NEWS). For comparison we use two standard information retrieval measures: micro averaged F_1 (MicroF_1) and macro averaged F_1 (MacroF_1). While the MicroF_1 measures the classification effectiveness over all decisions made by the classifier, the MacroF_1 measures the classification effectiveness for each individual class and averages them. All experiments were executed using a 10-fold cross-validation [44] procedure considering training, validation and test sets. The parameters were set using the validation set, and the effectiveness of the algorithms measured in the test partition.

We start by reporting the parameter setup performed in order to conduct our experimental evaluation, in Section 5.1. Then, in Section 5.2 we report and analyze the results obtained when using SD-TWF for the ACM-DL and MEDLINE datasets. Finally, in Section 5.3 we evaluate the use of the fully-automated strategy to devise the TWF to feed our temporally-aware classifiers and discuss some important aspects regarding its efficiency in terms of runtime. All the experiments were ran using a Quad-Core AMD Opteron^{TM} CPU with 16*GB* of RAM.

5.1. Parameter settings

An important aspect to be considered when dealing with the temporally-aware classifiers is that the TWF scale must be compatible with the values weighted by the TWF. Clearly, this is algorithm specific and should be properly set to ensure that the TWF effectively improves the classifier's decision rules without compromising them. To explicitly control the TWF scale (without modifying its shape), we introduce a scaling factor β which should be properly calibrated over the training set.

Hence, in order to run the experiments, two important parameters had to be set: the value of K for KNN and the scaling factor β. We first performed some experiments with KNN to define the value of K. This parameter significantly impacts the quality of classifier, and must be carefully chosen. The following values were tested, by means of a crossed validation over the training set, for each version of the traditional and temporally-aware algorithms: 3, 10, 30, 50, 150, and 200. For the traditional version of the algorithm $k = 30$ achieved better results, while for both the *in documents* and *in scores* versions of KNN the bestvalue of k was 50. The intuition for the traditional KNN to perform better with smaller values of K is that, as the number of neighbors increases, the variation on term-class relationships also increases, and the probability

Table 5
Results obtained when incorporating the SD-TWF to Rocchio, KNN, and Naive Bayes—ACM-DL.

Algorithm	Rocchio		KNN		Naive Bayes	
Metric	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)
Baseline	57.39	68.24	58.48	71.84	57.27	73.24
TWF	60.02	70.64	59.92	73.84	60.78	74.11
in documents	(+4.58) ▲	(+3.52) ▲	(+2.46) ▲	(+2.78) ▲	(+6.13) ▲	(+1.19) ●
TWF	59.85	72.47	62.02	74.45	44.85	63.93
in scores	(+4.29) ▲	(+6.20) ▲	(+6.05) ▲	(+3.63) ▲	(−27.69) ▼	(−14.56) ▼
TWF	59.27	71.39	59.78	73.85	56.23	72.35
in scores ext.	(+3.28) ▲	(+4.62) ▲	(+2.22) ▲	(+2.80) ▲	(−1.84) ●	(+1.23) ●

Table 6
Results Obtained when incorporating the SD-TWF to Rocchio, KNN, and Naive Bayes—MEDLINE.

Algorithm	Rocchio		KNN		Naive Bayes	
Metric	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)
Baseline	54.26	69.27	72.49	82.86	64.61	80.82
TWF	54.08	69.48	74.10	83.36	66.75	82.87
in documents	(−0.33) ●	(+0.30) ●	(+2.22) ▲	(+0.60) ●	(+3.31) ▲	(+2.54) ▲
TWF	63.95	77.63	75.89	86.35	58.12	80.49
in scores	(+17.86) ▲	(+12.07) ▲	(+4.69) ▲	(+4.21) ▲	(−10.04) ▼	(−0.41) ●
TWF	63.63	77.28	74.45	84.96	63.41	81.06
in scores ext.	(+17.27) ▲	(+11.56) ▲	(+2.70) ▲	(+2.53) ▲	(−1.89) ●	(+0.30) ●

of misclassification increases. On the other hand, when setting $K < 30$ the traditional version of KNN performed poorly due to overfitting. When considering temporal information by means of the proposed temporal weights, in contrast, more consistent information becomes available (due to a larger K), allowing a more accurate model. Finally, the *extended in scores* version of KNN performed better with $K = 3$ in the ACM-DL dataset and $K = 10$ in the MEDLINE and AG-NEWS datasets (recall that in this strategy, the KNN's classification model is built from reduced training sets, composed by documents belonging to the same point in time, which justifies this smaller value). For β, we empirically tested three values: 1, 10, and 100. The best value of each version of each classifier was considered. For Rocchio and KNN, the best results were obtained with $\beta = 1$. For Naive Bayes, the best value was $\beta = 10$.

5.2. Experiments with the SD-TWF

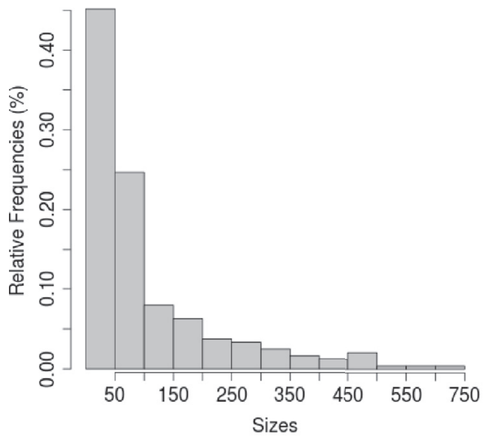
In this section, we report our experiments to compare the traditional and the proposed temporally-aware versions of Rocchio, KNN and Naive Bayes, using the SD-TWF for the ACM-DL and MEDLINE datasets. We defer the analysis regarding the AG-NEWS dataset to Section 5.3, when we discuss the obtained results using the FA-TWF. The results obtained for the ACM-DL and MEDLINE datasets are reported in Tables 5 and 6, respectively. In both tables, each line presents the results achieved by the versions of the classifiers identified in the first row and column. The values obtained for MacroF$_1$ ("macF$_1$") and MicroF$_1$ ("micF$_1$") are reported, as well as the percentage difference between values achieved by the temporally-aware methods and the traditional version of the classifiers. This percentage difference is followed by a symbol that indicates whether the variations are statistically significant according to a 2-tailed paired t-test, given a 99% confidence level. ▲ denotes a significant positive variation, ● a non significant variation and ▼ a significant negative variation. This notation will be adopted also in Section 5.3.

As we can see in Tables 5 and 6, all modified versions of Rocchio and KNN achieved better results than the baselines in ACM-DL. In MEDLINE, the versions "*in scores*" and "*extended in scores*" achieved statistically significant gains, while the versions "*in documents*" were statistically tied with the baseline. In particular, Rocchio with TWF *in scores* presents the most significant improvements in both datasets, with gains up to +17.86% and +12.07% for MacroF$_1$ and MicroF$_1$, respectively. Similarly, KNN with TWF *in scores* achieves the best results among all KNN variations, with gains of +6.05% and +4.21% for MacroF$_1$ and MicroF$_1$, respectively. In the case of Rocchio, the improvements achieved using the TWF can be explained by the fact that, in the traditional version, the documents are summarized in a unique representative vector (centroid), aggregating documents from distinct creation points in time, ultimately affecting the prediction ability of the classifier. In the case of KNN, the definition of class boundaries is done considering each training document independently. KNN assumes that documents of same class are located close by on the vectorial space. By using the TWF, the k nearest documents are reorganized, and the most temporally relevant documents are placed closer to the document being classified, according to the temporal distance between them.

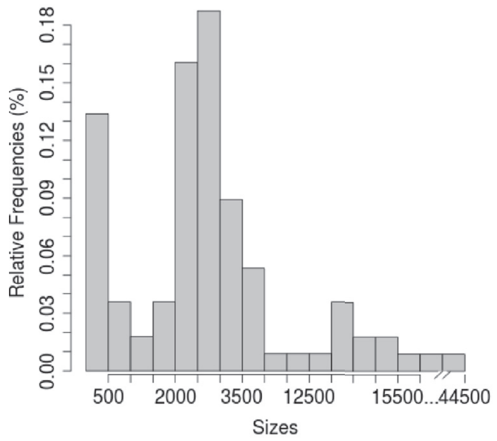
The Naive Bayes with TWF *in documents* presents better results for MacroF$_1$ on both ACM-DL and MEDLINE, and better MicroF$_1$ in the MEDLINE dataset. Note that the best improvement was achieved in MacroF$_1$, pointing out that this strategy effectively reduces the Naive Bayes bias towards the most frequent classes and, consequently, improves the effectiveness of such classifier when predicting documents from the smaller classes. However, in contrast with Rocchio and KNN, the Naive Bayes with TWF *in scores* performs poorly in both datasets. A closer look to the "*in scores*" strategy reveals that if such strategy is built upon a traditional classifier whose decision rule is strongly influenced by the negative documents (as KNN and Naive Bayes), its performance is deemed to be poor when applied to datasets with skewed $\langle c, p\rangle$ distributions. Although in KNN this problem can be ameliorated by a proper tuning of the parameter K, in Naive Bayes this is not possible. Thus, we attribute the poor performance of the Naive Bayes with TWF *in scores* to two major weaknesses of traditional Naive Bayes version. First, when facing skewed data distributions, the traditional version of Naive Bayes unwittingly prefers larger classes over others, causing decision boundaries to be biased (in this case, the prediction of the smaller classes is influenced by the negative documents belonging to the major classes). Second, when data is scarce, there is not enough information to perform accurate estimates, leading to bad results.

Table 7
Results obtained for the least and most frequent classes ⟨*c*, *p*⟩ sampling for Naive Bayes—MEDLINE.

Naive Bayes	Least frequent classes ⟨*c*, *p*⟩			Most frequent classes ⟨*c*, *p*⟩		
Metric	CV	$macF_1$(%)	$micF_1$(%)	CV	$macF_1$(%)	$micF_1$(%)
Baseline	0.57	74.72	80.42	0.43	88.70	87.65
TWF in scores		78.49 (+5.04) ▲	84.75 (+5.38) ▲		91.16 (+2.77) ▲	89.66 (+2.29) ▲



(a) ACM-DL Dataset



(b) MEDLINE Dataset

Fig. 8. Relative ⟨*c*, *p*⟩ sizes.

The skewness of data distribution among classes ⟨*c*, *p*⟩ can be quantified by the Coefficient of Variation $CV = \frac{\sigma}{\mu}$ of their sizes, where σ and μ stand for the standard deviation and mean. To explore the impact of data skewness on Naive Bayes, we sampled MEDLINE, creating two sub-collections composed by the least and most frequent classes ⟨*c*, *p*⟩, minimizing data skewness. While the entire collection presents $CV = 1.33$, the sub-collections with the least and most frequent classes present *CV* equal to 0.57 and 0.43, respectively. As we can observe in Tables 6 and 7, the greater the CV, the worse are the results.

Fig. 8 shows the histogram with the percentages of classes ⟨*c*, *p*⟩ with up to a given size (specified in the x-axis) for the ACM-DL and MEDLINE datasets. As we can observe in Fig. 8a, the data scarcity is prominent in the ACM-DL dataset, contributing to the poor performance of the Naive Bayes with TWF *in scores*. Notice that 70% of classes ⟨*c*, *p*⟩ have less than 100 documents, a number too low to guarantee accurate estimates. This is also observed in the MEDLINE dataset (see Fig. 8b), but at a smaller extent—more specifically, 13% of the classes ⟨*c*, *p*⟩ are composed by less than 500 documents, whereas 35% are composed by 2500 to 3000 documents. In addition, ACM-DL has an even more skewed data distribution over each time point (with a *CV* equal to 1.69 regarding the ⟨*c*, *p*⟩ sizes), preventing us to sample it in sub-collections with smaller CV, as performed with the MEDLINE dataset.

Recall that the main motivation behind the "*extended in scores*" strategy is to ameliorate the class imbalance problem, that negatively impacts the "*in scores*" effectiveness. In the "*extended in scores*" strategy, the influence of negative documents is bounded when considering data from each point in time in isolation. More specifically, the class imbalance is not given by the ⟨*c*, *p*⟩ distribution as in the "in scores" strategy, but by the observed class imbalance *within each point in time*. In fact, this class distribution is typically more evenly distributed than the artificial ⟨*c*, *p*⟩ distribution. As we can observe in the reported results, the *extended in scores* version of Naive Bayes performed better than its *in scores* version. However, it still did not performed better than the baseline (with statistically equivalent results in all cases), due to the discussed data scarcity problem, which prevents this classifier to learn accurate estimates about the class densities. Strategies to handle data scarcity (for instance, by oversampling the training set) are one of the current research focuses and we plan to further investigate this matter as future work.

In general, using TWF *in scores* in most cases led to better results than those applying TWF *in documents*. This is due to the fact that the "*in scores*" strategy addresses simultaneously the three discussed temporal effects, namely, the class distribution variation (*CD*), the pairwise class similarity variation (*CS*) and the term distribution variation (*TD*), whereas the "*in documents*" strategy takes into account just the *TD* effect, as discussed next. Furthermore, as we can observe in Table 6 regarding the MEDLINE dataset, the "*in documents*" strategy the results obtained were statistically equivalent to the baselines in almost all cases, with the Naive Bayes being an exception. As will be discussed in the following, this is due to the MEDLINE characteristics with respect to the extent of the *TD* effect.

Recall that the *temporal weighting in documents* strategy weights each training document by the TWF according to its temporal distance to the test document. The TWF is modeled according to the observed variations over time in the term-class relationships for each dataset, ultimately addressing the *TD* aspect. Furthermore, recall that the both the temporal weighting in scores and its extended version ties together the observed patterns to both class and temporal information. While the "*in scores*" transforms the class domain from $\mathbb{C}$ to $\mathbb{C} \times \mathbb{P}$ (generating a new training set), the "*extended in scores*" strategy groups training documents into partitions composed by documents created at the same point in time, performing a traditional classification procedure over each partition. Both strategies assume that the temporal effects may be safely neglected within a single point in time and thus the classification models learned considering each point in time in isolation are not affected by them. However, as previously stated, considering only the data related to a single point in time may disregard valuable information to learn an accurate classification model. Thus, the second step of these strategies consists of aggregating the information learned for each point in time, weighting the obtained classification scores by the TWF. Aggregating the obtained scores for each point in time is affected by the *TD* effect, since the scores reflect the relationships between terms and classes. In order to overcome the observed variations in the term-class relationships across the different points in time, the TWF is used to weight them according to the temporal distance between the points in time associated to each partition and the creation time of the test document. Thus, while the first step addresses the *CD* and *CS* effects, the second step addresses the *TD* aspect observed when aggregating the scores.

Analyzing the reported results, we can observe that, for ACM-DL, the three strategies achieved significant gains. Considering the *temporal weighting in documents* approach, we can justify its gains due to the high impact of *TD* in that dataset. Moreover, since ACM-DL is also subject to a high impact of both *CD* and *CS*, both the temporal weighting in scores and its extended version also performed well, since they address such effects, as previously discussed. In MEDLINE, in contrast, since the impact of *TD* is smaller than the impact of the other two effects, we should expect less significant gains achieved by *temporal weighting in documents*. Indeed, this was the observed behavior: such approach achieved statistical ties compared to baselines in almost all cases. However, as both *CD* and *CS* are important factors in that dataset, we can observe statistically significant improvements in classification effectiveness when the temporal weighting in scores and its extension are applied. Furthermore, the largest improvements are achieved when the temporal weighting in scores is applied with the Rocchio classifier, which, as discussed in [45], is the most affected by both *CD* and *CS* in that dataset.

5.3. Experiments with the FA-TWF

In this section, we report our experimental evaluation to assess the effectiveness of the proposed temporally-aware classifiers using FA-TWF. The goal here is to increase the applicability of the temporally-aware classifiers. For example, even if uncertain about the expression (and parameters) of the TWF associated with the AG-NEWS dataset, we can still determine the weights associated with each temporal distance using the procedure described in Algorithm 1, and use our temporally-aware classifiers with the learned TWF. Recall that, in this case, the TWF is learned from the training set $\mathbb{D}$. An interesting aspect to be analyzed refers to the amount of data required to accurately estimate this function. *Is the whole training set needed to learn the TWF ?* In order to have a first glance on this matter, we evaluate our strategies using the TWF learned from the entire $\mathbb{D}$ and from a sample composed by 10% of $\mathbb{D}$, selected by a per point in time random sampling (to guarantee that each point in time will have at least one document).

We start by comparing the results obtained using the SD and FA-TWFs, considering the ACM-DL and MEDLINE datasets. We stress here that the results obtained estimating the TWF using the three classifiers (line 4 of Algorithm 1) were statistically equivalent, as could be expected by the observed similarities in Fig. 4. Thus, we report just the results obtained by estimating the TWF using the Rocchio classifier.

An important conclusion that can be drawn from Tables 8 and 9, is that using FA-TWF led to statistically equivalent results to the use of SD-TWF. This was assessed by a 2-tailed paired *t*-test, with 99% confidence level. In fact, there is an interesting similarity between the distribution of temporal distances used to determine the TWF expression, illustrated in Fig. 2, and the FA-TWFs for both datasets, illustrated in Figs. 4a and 4b. This implies that we can adopt the automated procedure to determine the TWF without affecting the effectiveness of the temporally-aware algorithms. Similarly, the poor performance of both the in scores versions of the Naive Bayes classifier is also attributed to the class imbalance problem (for the TWF *in scores* strategy) and to the lack of training documents associated to each class (for the extended TWF *in scores* strategy), just as before.

Another important aspect to be observed in Tables 8 and 9 is that it does not matter using either the entire training set $\mathbb{D}$ or just 10% of it to learn the TWF. In fact, both cases led to statistically equivalent results (assessed by a 2-tailed paired *t*-test with 99% confidence), in all cases. This is important property of Algorithm 1 since the smaller is the training set (that is, its input) the smaller the expected runtime to learn the TWF (Table 10).

We now turn our attention to the AG-NEWS dataset. In this case, all experiments consider the FA-TWF version. Similarly to the ACM-DL and MEDLINE datasets, the temporally-aware versions of the Rocchio classifier present the most significant improvements over the baseline, with gains up to 6.29% and 14.00% for MacroF$_1$ and MicroF$_1$, respectively. As in the MEDLINE dataset, both the “*in scores*” versions of the Rocchio classifier performed better than its “*in documents*” version. This is due to the nature of this dataset, which presents more prominent variations in the class distribution (*CD*) and the class similarities (*CS*) than in the term distribution (*TD*). The impact of the *TD* effect is consistently lower than the impact of *CD* (or *CS*) on all four algorithms in this dataset. However, unlike the temporally-aware versions of Rocchio, the “*in documents*” versions of KNN and Naive Bayes were statistically tied with their baselines (with the Naive Bayes with TWF *in documents* being an exception, with statistically significant gains in the MacroF$_1$ of up to 2.40%). This is justifiable by the smaller extent of the *TD* effect in the AG-NEWS dataset. Accordingly, their ”*in scores*” versions should perform better and this was not the observed. Indeed, both the KNN and Naive Bayes with TWF *in scores* led to significant losses in both MacroF$_1$ and MicroF$_1$. Again, we attribute this to both the class imbalance and the data scarcity problems. In order to provide evidence for this problem, in Fig. 9 we show the histogram of the $\langle c, p\rangle$ sizes (as done with the other two datasets). In fact, we can observe that 72% of the $\langle c, p\rangle$ sizes are smaller than 200 (with 46% of the $\langle c, p\rangle$ classes composed by at most 100 documents), with CV equal to 1.85—a much more skewed and sparse distribution than of the other two datasets.

Recall that the “*extended in scores*” strategy aims at minimizing the influence of the class imbalance problem in the classification effectiveness. In fact, this strategy outperformed the “*in scores*” strategy in all cases. However, it was not able to outperform the baselines. This is due to the previously discussed data scarcity problem, which is, by the way, more pronounced in the AG-NEWS dataset than in the other two datasets. In contrast to the improvements obtained by the “*extended in scores*” version of the KNN classifier in the MEDLINE dataset, in the AG-NEWS it incurred in statistical ties. Furthermore, in contrast to the statistical ties obtained by the *extended in scores* version of the Naive Bayes classifier, in the AG-NEWS dataset it produced statistically significant losses. We conjecture that the adoption of strategies to overcome the data scarcity problem may improve the effectiveness of such strategy. Again, we leave this matter for future work.

Finally, we also compared the best temporally-aware classifiers (using FA-TWF) to the state-of-the-art *Support Vector Machine* [46] classifier. We adopted an efficient SVM implementation, SVM_Perf [47], which is based on the maximum-margin ap-

Table 8
Results obtained when incorporating the FA-TWF to Rocchio, KNN, and Naive Bayes—ACM-DL.

Algorithm	Rocchio		KNN		Naive Bayes	
Metric	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)
Baseline	57.39	68.24	58.48	71.84	57.27	73.24
TWF (100% of $\mathbb{D}$)	60.21	70.70	60.08	73.88	61.38	74.60
in documents	(+4.91) ▲	(+3.60) ▲	(+2.74) ▲	(+2.84) ▲	(+7.18) ▲	(+1.86) ●
TWF (10% of $\mathbb{D}$)	60.52	70.88	61.02	74.27	61.44	74.24
in documents	(+5.45) ▲	(+3.87) ▲	(+4.84) ▲	(+3.82) ▲	(+7.28) ▲	(+1.36) ●
TWF (100% of $\mathbb{D}$)	60.47	72.90	61.88	74.53	45.16	64.55
in scores	(+5.47) ▲	(+6.83) ▲	(+5.81) ▲	(+3.74) ▲	(−26.82) ▼	(−13.46) ▼
TWF (10% of $\mathbb{D}$)	59.68	72.40	61.37	73.77	44.47	64.58
in scores	(+3.99) ▲	(+6.10) ▲	(+4.94) ▲	(+2.69) ▲	(−28.78) ▼	(−13.41) ▼
TWF (100% of $\mathbb{D}$)	59.96	71.99	59.80	73.95	56.28	72.73
in scores ext.	(+4.48) ▲	(+5.49) ▲	(+2.26) ▲	(+2.94) ▲	(−1.76) ●	(−0.70) ●
TWF (10% of $\mathbb{D}$)	59.85	71.79	59.76	73.85	56.19	72.70
in scores ext.	(+4.29) ▲	(+5.20) ▲	(+2.19) ▲	(+2.80) ▲	(−1.89) ●	(−0.74) ●

Table 9
Results obtained when incorporating the FA-TWF to Rocchio, KNN, and Naive Bayes—MEDLINE.

Algorithm	Rocchio		KNN		Naive Bayes	
Metric	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)
Baseline	54.26	69.27	72.49	82.86	64.61	80.82
TWF (100% of $\mathbb{D}$)	54.03	69.48	73.96	82.76	67.95	82.98
in documents	(−0.43) ●	(+0.30) ●	(+2.03) ▲	(−0.12) ●	(+5.17) ▲	(+2.67) ▲
TWF (10% of $\mathbb{D}$)	55.01	70.35	73.63	82.87	67.84	82.89
in documents	(+1.38) ●	(+1.56) ●	(+1.57) ●	(+0.01) ●	(+5.00) ▲	(+2.56) ▲
TWF (100% of $\mathbb{D}$)	64.47	77.12	75.99	86.33	58.20	80.48
in scores	(+18.82) ▲	(+11.33) ▲	(+4.83) ▲	(+4.19) ▲	(−9.92) ▼	(−0.42) ●
TWF (10% of $\mathbb{D}$)	64.25	77.03	75.88	86.36	58.23	80.51
in scores	(+18.41) ▲	(+11.20) ▲	(+4.68) ▲	(+4.22) ▲	(−9.87) ▼	(−0.38) ●
TWF (100% of $\mathbb{D}$)	64.53	77.16	74.63	85.07	64.64	81.12
in scores ext.	(+18.93) ▲	(+11.39) ▲	(+2.95) ▲	(+2.67) ▲	(−0.05) ●	(+0.37) ●
TWF (10% of $\mathbb{D}$)	64.32	77.24	74.74	84.99	64.74	81.10
in scores ext.	(+18.54) ▲	(+11.51) ▲	(+3.10) ▲	(+2.57) ▲	(+0.20) ●	(+0.35) ●

Table 10
Results obtained when incorporating the FA-TWF to Rocchio, KNN, and Naive Bayes—AG-NEWS.

Algorithm	Rocchio		KNN		Naive Bayes	
Metric	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)	macF$_1$(%)	micF$_1$(%)
Baseline	54.89	58.16	60.05	68.49	60.92	67.83
TWF (100% of $\mathbb{D}$)	58.34	62.34	58.96	67.45	62.24	68.46
in documents	(+6.29) ▲	(+7.19) ▲	(-1.85) ●	(-1.54) ●	(+2.17) ▲	(+0.93) ●
TWF (10% of $\mathbb{D}$)	58.35	62.29	58.91	67.35	62.38	68.55
in documents	(+6.30) ▲	(+5.68) ▲	(-1.90) ●	(-1.66) ●	(+2.40) ▲	(+1.06) ●
TWF (100% of $\mathbb{D}$)	57.82	66.26	58.36	64.94	51.65	61.91
in scores	(+5.34) ▲	(+13.93) ▲	(-2.90) ▼	(-5.47) ▼	(-15.22) ▼	(-8.73) ▼
TWF (10% of $\mathbb{D}$)	58.01	66.30	58.15	64.84	51.69	61.97
in scores	(+5.68) ▲	(+14.00) ▲	(-3.16) ▼	(-5.33) ▼	(-15.15) ▼	(-8.64) ▼
TWF (100% of $\mathbb{D}$)	57.72	66.12	59.12	68.93	56.43	65.21
in scores ext.	(+5.16) ▲	(+13.69) ▲	(-1.57) ●	(+0.64) ●	(-7.37) ▼	(-3.86) ▼
TWF (10% of $\mathbb{D}$)	57.69	65.99	59.08	68.77	56.47	65.22
in scores ext.	(+5.10) ▲	(+13.46) ▲	(-1.61) ●	(+0.41) ●	(-7.30) ▼	(-3.85) ▼

proach and can be trained in linear time. We used an one-against-all [see 48] methodology to adapt binary SVM to multi-class classification. Such comparison is presented in Table 11. For ACM-DL dataset (Table 11a),there are significant gains of 3.29% and 2.45% in macroF$_1$ (and statistical ties in microF$_1$), for KNN with TWF *in scores* and Naive Bayes with TWF *in documents*, respectively. Furthermore, both the Rocchio with TWF *in scores* and KNN with TWF *in documents* obtained results statistically equivalent to the SVM results. In all these three cases, the temporally-aware classifiers were faster than the SVM by two orders of magnitude. As we shall discuss next, for the MEDLINE dataset (Table 11b), the most significant gains are of 2.03% and 2.48% in MacroF$_1$ and MicroF$_1$, respectively, obtained by the KNN with TWF *in scores*. The *extended in scores* version of KNN achieved statistically tied results. As we shall discuss in Section 5.4, both classifiers are significantly faster than SVM. Considering that SVM is a state of the art classifier, and that both datasets are imbalanced, our results evidence the quality of the proposed solution. Considering the AG-NEWS dataset (Table 11c), the best performing temporally-aware classifier was unable to outperform the SVM due to the limitations already discussed. However, it is worth to say that our temporally-aware classifier was not drastically outperformed, and there exists room for improvements.

5.4. Runtime analysis

Now we turn our attention to the efficiency of our proposed classifiers, in terms of execution time. We start by considering the average time spent by the classifiers in each iteration of the K-

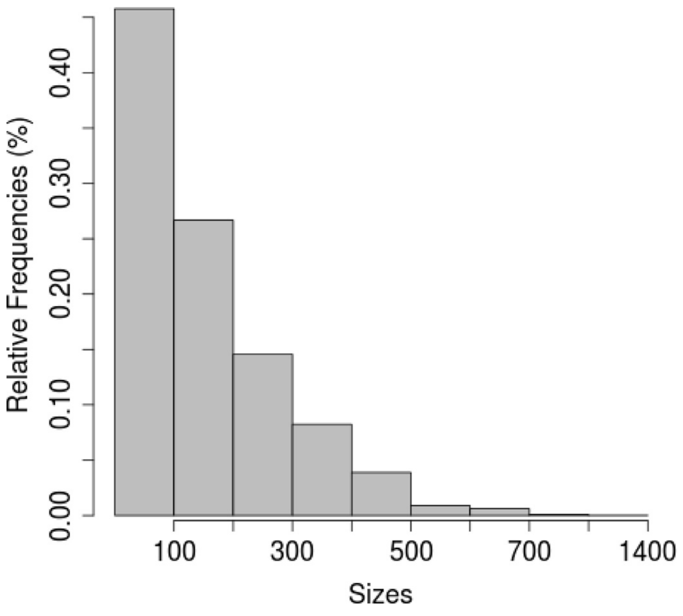


Fig. 9. Relative ⟨*c*, *p*⟩ Sizes for AG-NEWS dataset.

Table 11
Effectiveness comparison: Best performing temporally-aware classifiers *versus* SVM.

Algorithm	Metric	
	macF$_1$(%)	micF$_1$.(%)
SVM	59.91	73.88
Rocchio with TWF *in scores*	60.47 (+0.93) ●	72.90 (−1.34) ●
KNN with TWF *in documents*	59.78 (−0.22) ●	73.88 (+0.00) ●
KNN with TWF *in scores*	61.88 (+3.29) ▲	74.53 (+0.88) ●
Naive Bayes with TWF *in documents*	61.38 (+2.45) ▲	74.60 (+0.97) ●

(a) ACM-DL Dataset

Algorithm	Metric	
	macF$_1$(%)	micF$_1$.(%)
SVM	74.48	84.24
KNN with TWF *in scores*	75.99 (+2.03) ▲	86.33 (+2.48) ▲
KNN with TWF *in scores ext.*	74.63 (+0.20) ●	85.07 (+0.98) ●

(b) MEDLINE Dataset

Algorithm	Metric	
	macF$_1$(%)	micF$_1$.(%)
SVM	64.94	72.59
Naive Bayes with TWF *in documents*	62.38 (−4.10) ▼	68.55 (−5.56) ▼

(c) AG-NEWS Dataset

Fold cross validation, and comparing the temporally-aware algorithms (with FA-TWF) with both their traditional counterparts and the state-of-the-art *Support Vector Machine* [46] classifier (using the previously described SVM_Perf implementation). Next, we analyze the additional cost associated with the automatic determination of the FA-TWF.

We report in Table 12 the average execution time of each traditional classifier (rows entitled "*traditional*", including the measurement for the SVM classifier), their temporally-aware versions (lines "In Documents", "In Scores" and "Extended In Scores"), along with the standard deviation over the mean value (reported after the ± symbol). We consider the execution time measured for the overall classification task, comprised by both the training and test stages.[1] The measurements regarding the ACM-DL dataset refers to the classification of 2,490 documents using 19,918 training documents, while the measurements regarding the MEDLINE dataset refers to the testing of 86,145 documents with a classification model learned from 689,163 training documents. Finally, the measurements regarding the AG-NEWS dataset refers to the classification of 83,580 documents based on a classification model learned from 668,636 documents. Clearly, the columns of this table are not comparable, and we consider the measurements for each dataset independently.

As one could expect, our temporally-aware classifiers are typically slower than their traditional counterparts. This comes as no surprise, since there is the overhead of considering and managing the temporal information. Furthermore, the temporally-aware classifiers are, by nature, lazy classifiers, which comes at the cost of a higher runtime. Furthermore, our temporally-aware versions incurred in a higher increase in execution time in the AG-NEWS dataset, due to the higher number of points in time of this dataset. However, in almost all cases our lazy temporally-aware classifiers were more efficient, in terms of execution time, than the SVM classifier.

We also compared the best version of the methods previously proposed, for example, the KNN with TWF *in scores* and Naive Bayes with TWF *in documents*, to the SVM classifier, in terms of efficiency (execution time). Such comparison is presented in Table 13. For ACM-DL dataset (Table 13a), our best performing classifiers were up to 13 times faster than SVM, while outperforming the SVM effectiveness in MacroF$_1$ and tying with it in MicroF$_1$ (as reported in Table 11a). For the MEDLINE dataset (Table 13b), the KNN with TWF *in scores* was more than two times faster than SVM, and, as previously reported, outperformed this classifier in both MacroF$_1$ and MicroF$_1$. The *extended in scores* version of KNN achieved was more than three times faster than SVM. Considering the AG-NEWS dataset (Table 13c), our best performing temporally-aware classifier was almost eight times faster than the SVM (but unable to outperform the SVM classifier in terms of effectiveness).

Finally, we now consider the efficiency of the FA-TWF determination algorithm.[2] Recall from Algorithm 1 that it is necessary to perform a classification over the training set in order to estimate the a posteriori probability distribution $P(p_i d_i)$ and then determine the FA-TWF. There are two key aspects to be considered. First, since it does not matter which of the three classifiers are used to learn the FA-TWF (since they led to statistically significant results), it is advisable to use the Rocchio classifier for doing so, since it is, by far, the most efficient one (as can be observed in Table 12, by comparing the traditional versions of each of them). Second, it is clear that if the training set size increases, the cost involved at determining the FA-TWF also increases and can be potentially prohibitive. To better understand the dependency between the execution time of the FA-TWF determination and the training set size, we measured the execution time spent to determine the FA-TWF using the entire training set $\mathbb{D}$ and a per point in time sample of $\mathbb{D}$, by randomly selecting 10% of the documents of $\mathbb{D}$. We then compared such measurements with the time spent by the fastest temporally-aware classifiers and the SVM classifier, for each explored dataset. This comparison is reported in Table 14. As we can observe, the time required to automatically learn the FA-TWF is negligible when compared to the time spent by the classification task. In addition to this efficiency aspect, the FA-TWF determination is inherently an offline procedure, guaranteeing its practical applicability.

[1] Recall that in our experimental setup, using the 10-fold cross validation strategy, one fold is used as test set, another one is retained as the validation set and the remaining folds are used as training set.

[2] Reminding that this step can be performed offline.

Table 12
Execution time (in seconds) of each explored ADC algorithm.

Algorithm		Runtime (seconds) per Dataset		
		ACM-DL	MEDLINE	AG-NEWS
SVM	Traditional	144.10 ± 5.30	26955.0 ± 2356.0	28667.0 ± 1151.0
Rocchio	Traditional	2.00 ± 0.00	111.0 ± 0.0	96.5 ± 0.5
	In Documents	6.60 ± 0.52	209.5 ± 12.5	4615.5 ± 89.5
	In Scores	9.00 ± 0.00	300.5 ± 3.5	5287.5 ± 9.5
	Extended In Scores	7.20 ± 0.42	263.5 ± 0.5	3807.0 ± 29.0
KNN	Traditional	8.90 ± 0.32	13442.5 ± 79.5	8154.0 ± 60.0
	In Documents	11.03 ± 0.48	15949.0 ± 51.0	10368.5 ± 8.5
	In Scores	10.10 ± 0.31	12557.5 ± 630.5	8630.5 ± 349.5
	Extended In Scores	8.40 ± 0.75	7753.5 ± 78.5	4711.5 ± 46.5
	Traditional	5.00 ± 0.00	213.0 ± 7.0	186.5 ± 0.5
Naive	In Documents	9.10 ± 0.32	293.0 ± 2.0	3780.0 ± 95.0
Bayes	In Scores	63.80 ± 1.32	1311.0 ± 1.0	43570.0 ± 85.0
	Extended In Scores	60.50 ± 1.18	656.5 ± 6.5	38966.5 ± 108.5

Table 13
Execution time Comparison: Best performing temporally-aware classifiers *versus* SVM.

Algorithm	Time (s)
SVM	144.10 ± 5.30
Rocchio with TWF *in scores*	9.00 ± 0.00
KNN with TWF *in documents*	11.03 ± 0.48
KNN with TWF *in scores*	10.10 ± 0.31
Naive Bayes with TWF *in documents*	9.10 ± 0.32
(a) ACM-DL Dataset	
Algorithm	Time (s)
SVM	26955.0 ± 2356.0
KNN with TWF *in scores*	12557.5 ± 630.5
KNN with TWF *in scores ext.*	7753.5 ± 78.5
(b) MEDLINE Dataset	
Algorithm	Time (s)
SVM	28667.0 ± 1151.0
Naive Bayes with TWF *in documents*	3780.0 ± 95.0
(c) AG-NEWS Dataset	

Table 14
Execution time of the FA-TWF Estimation using the Rocchio classifier.

Dataset	Runtime			
	TWF Determination		Fastest Temporally-Aware	
	10% of $\mathbb{D}$	Entire $\mathbb{D}$	Classifiers	
ACM-DL	**0.77 ± 0.02**	4.49 ± 0.04	**6.60 ± 0.52**	Rocchio *in documents*
MEDLINE	**31.00 ± 3.00**	180.00 ± 28.00	**209.50 ± 12.50**	Rocchio *in documents*
AG-NEWS	**120.00 ± 8.00**	1560.00 ± 25.00	**3780.00 ± 95.00**	Naive Bayes *in documents*

Finally, it is important to mention that we have made all codes and datasets used in our experiments and analysis available through the link http://www.lbd.dcc.ufmg.br/lbd/collections/a-two-stage-machine-learning-approach-for-temporally-robust-text-classification.

6. Conclusions

In this work, we proposed a strategy to minimize the impact of the temporal effects in three ADC algorithms, aiming at improving their classification effectiveness by incorporating the temporal information into document classifiers. We proposed a fully-automated approach to learn from the training data a temporal weighting function (FA-TWF) that captures changes on term-class relationships over time, effectively reflecting the varying nature of the explored data. Moreover, we proposed different approaches to incorporate the FA-TWF into three traditional classifiers, namely Rocchio, KNN and Naive Bayes. The proposed FA-TWF greatly expands the applicability and generality of our previous solutions which relied on complex, sometimes unfeasible, statistical tests.

We evaluated our proposals in three real-world datasets that suffer with temporal effects, namely ACM_DL, MEDLINE and AG-NEWS, comparing them with the traditional versions of the classifiers. Our results show that temporally-aware classifiers significantly improve the results over the traditional ones, with gains up to 17%. We also studied efficiency issues regarding the incorporation of the TWF into the classifiers. We found, for instance, that sampling 10% of the training documents to learn the FA-TWF provided the same gains in the temporally-aware classifiers as using all the whole training set. This reduces even more the additional cost in the runtime of the classification task. Also, both temporally-aware KNN and Naive Bayes achieved more effective results than SVM for two datasets (ACM and MEDLINE), also with better overall performance (i.e., up to 13 times faster than SVM). Considering that SVM is a state-of-the-art classifier, and that collections are very unbalanced, our results evidence the quality of our temporal-aware strategies.

As future work, we intend to exploit more complex concepts such as compound features (a.k.a., *c-features*), proposed in [49], as well as word2vec representations [50] to understand how they temporally evolve and how they can be used in our solutions. We also intend to incorporate the proposed TWF in recent classification algorithms that represent the state-of-the-art in text classification (e.g., [51]) and see how they behave under the temporal effects.

Acknowledgments

This work was partially supported by CNPq, CAPES, FINEP, Fapemig, MASWEB, INCT-Cyber and INWEB.

References

[1] T. Salles, L. Rocha, M.A. Gonçalves, J.M. Almeida, F. Mourão, W. Meira, F. Viegas, A quantitative analysis of the temporal effects on automatic text classification, J. Assoc. for Inf. Sci. Technol. 67 (7) (2016) 1639–1667, doi:10.1002/asi.23452.
[2] C. Aggarwal, C. Zhai, A survey of text classification algorithms, in: C.C. Aggarwal, C. Zhai (Eds.), Mining Text Data, Springer US, 2012, pp. 163–222.
[3] J.R.M. Palotti, T. Salles, G.L. Pappa, F. Arcanjo, M.A. Gonçalves, W.M. Jr., Estimating the credibility of examples in automatic document classification, J. Inf. Manage. 1 (3) (2010) 439–454.

[4] A. Tsymbal, The Problem of Concept Drift: Definitions and Related Work, Technical Report, Department of Computer Science, Trinity College, Dublin, Ireland, 2004.
[5] C. Yang, J. Zhou, Non-stationary data sequence classification using online class priors estimation, Pattern Recognit. 41 (8) (2008) 2656–2664.
[6] Z. Zhang, J. Zhou, Transfer estimation of evolving class priors in data stream classification, Pattern Recognit. 43 (9) (2010) 3151–3161.
[7] T. Salles, L. Rocha, G.L. Pappa, F. Mourão, M.A. Gonçalves, W.M. Jr., Temporally-aware algorithms for document classification, in: Proceedings of the International ACM SIGIR Conference on Research & Development of Information Retrieval, Genebra, Switzerland, 2010a, pp. 307–314.
[8] T. Salles, L. Rocha, F. Mourão, G.L. Pappa, L. Cunha, M.A. Gonçalves, W.M. Jr., Automatic document classification temporally robust, J. Inf. Data Manage. 1 (2) (2010b) 199–212.
[9] T. Salles, T.N.C. Cardoso, V.C. de Oliveira, L.C. da Rocha, M.A. Gonçalves, Tackling temporal effects in automatic document classification, JIDM 2 (3) (2011) 417–432.
[10] G. Forman, Tackling concept drift by temporal inductive transfer, in: Proceedings of the International ACM SIGIR Conference on Research & Development of Information Retrieval, Washington, USA, 2006, pp. 252–259.
[11] F. Mourão, L. Rocha, R. Araújo, T. Couto, M. Gonçalves, W. Meira Jr., Understanding temporal aspects in document classification, in: Proceedings of the International Conference on Web Search and Web Data Mining, Palo Alto, USA, 2008, pp. 159–170.
[12] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of the Brazilian Symposium on Artificial Intelligence, São Luís, Brazil, 2004, pp. 286–295.
[13] K. Nishida, K. Yamauchi, Learning, detecting, understanding, and predicting concept changes, in: Proceedings of the International Joint Conference on Neural Networks, Atlanta, USA, 2009, pp. 283–290.
[14] A. Dries, U. Rückert, Adaptive concept drift detection, Stat. Anal. Data Min. 2 (5–6) (2009) 311–327.
[15] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: Proceedings of the International Conference on Discovery Science, Sendai, Japan, 2007, pp. 264–269.
[16] W.W. Cohen, Y. Singer, Context-sensitive learning methods for text categorization, ACM Trans. Inf. Syst. 17 (2) (1999) 141–173.
[17] R.-L. Liu, Y.-L. Lu, Incremental context mining for adaptive document classification, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 2002, pp. 599–604.
[18] S. Lawrence, C.L. Giles, Context and page analysis for improved web search, IEEE Internet Comput. 2 (4) (1998) 38–46.
[19] N.H.M. Caldwell, P.J. Clarkson, P.A. Rodgers, A.P. Huxor, Web-based knowledge management for distributed design, IEEE Intell. Syst. 15 (3) (2000) 40–47.
[20] Y.S. Kim, S.S. Park, E. Deards, B.H. Kang, Adaptive web document classification with MCRDR, in: Proceedings of the International Conference on Information Technology: Coding and Computing, Las Vegas, USA, 2004, pp. 476–480.
[21] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: Proceedings of the International Conference on Machine Learning, Stanford, USA, 2000, pp. 487–494.
[22] I. Žliobaitė, Combining time and space similarity for small size learning under concept drift, in: Proceedings of the International Symposium on Foundations of Intelligent Systems, Prague, Czech Republic, 2009, pp. 412–421.
[23] R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Intell. Data Anal. 8 (3) (2004) 281–300.
[24] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.
[25] L.C. da Rocha, F. Mourão, H. Mota, T. Salles, M.A. Gonçalves, W.M. Jr., Temporal contexts: effective text classification in evolving document collections, Inf. Syst. 38 (3) (2013) 388–409.
[26] M.M. Lazarescu, S. Venkatesh, H.H. Bui, Using multiple windows to track concept drift, Intell. Data Anal. 8 (1) (2004) 29–59.
[27] L.I. Kuncheva, I. Žliobaitė, On the window size for classification in changing environments, Intell. Data Anal. 13 (6) (2009) 861–872.
[28] A. Bifet, R. Gavaldà, Kalman filters and adaptive windows for learning in data streams, in: Discovery Science, Barcelona, Spain, 2006, pp. 29–40.
[29] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the SIAM International Conference on Data Mining, Minneapolis, USA, 2007, pp. 443–448.
[30] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intell. Data Anal. 11 (1) (2007) 3–28.
[31] J.Z. Kolter, M.A. Maloof, Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift, Technical Report, Department of Computer Science, Georgetown University, Washington, USA, 2003.
[32] G. Folino, C. Pizzuti, G. Spezzano, An adaptive distributed ensemble approach to mine concept-drifting data streams, in: Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, Patras, Greece, 2007, pp. 183–188.
[33] I. Koychev, Gradual forgetting for adaptation to concept drift, in: Proceedings of the ECAI Workshop Current Issues in Spatio-Temporal Reasoning, Berlin, Germany, 2000, pp. 101–106.
[34] R. Klinkenberg, S. Rüping, Concept drift and the importance of example, in: Text Mining - Theoretical Aspects and Applications, Physica-Verlag, 2003, pp. 55–78.
[35] X. Kong, P.S. Yu, An ensemble-based approach to fast classification of multi-label data streams, in: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, Orlando, USA, 2011, pp. 95–104.
[36] D.B. Clarkson, Y.-a. Fan, H. Joe, A remark on algorithm 643: FEXACT: an algorithm for performing fisher's exact test in r x c % contingency tables, ACM Trans. Math. Software 19 (4) (1993) 484–488.
[37] E. Limpert, W.A. Stahel, M. Abbt, Log-normal distributions across the sciences: keys and clues, Bioscience 51 (5) (2001) 341–352.
[38] P.E. D'Agostino R.B., Tests for departure from normality, Biometrika 60 (1973) 613–622.
[39] S.K. Crow EL, Log-Normal Distributions: Theory and Application, New York: Dekker, 1988.
[40] E. Chen, Y. Lin, H. Xiong, Q. Luo, H. Ma, Exploiting probabilistic topic models to improve text categorization under class imbalance, Inf. Process. Manag. 47 (2) (2011) 202–214.
[41] Y.-Q. Miao, M. Kamel, Pairwise optimized rocchio algorithm for text categorization, Pattern Recognit. Lett. 32 (2) (2011) 375–382.
[42] S. Tan, Neighbor-weighted k-nearest neighbor for unbalanced text corpus, Expert Syst. Appl. 28 (4) (2005) 667–671.
[43] Z. Lin, Z. Hao, X. Yang, X. Liu, Several SVM ensemble methods integrated with under-sampling for imbalanced data learning, in: Proceedings of the International Conference on Advanced Data Mining and Applications, Beijing, China, 2009, pp. 536–544.
[44] L. Breiman, P. Spector, Submodel Selection and Evaluation in Regression - the X-Random Case, Int. Stat. Rev. 60 (3) (1992) 291–319.
[45] T. Salles, Automatic Document Classification Temporally Robust, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil, 2011 Master's thesis.
[46] T. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999, pp. 169–184.
[47] T. Joachims, Training linear svms in linear time, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, USA, 2006, pp. 217–226.
[48] C.D. Manning, P. Raghavan, H. Schtze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, 2008.
[49] F. Figueiredo, L.C. da Rocha, T. Couto, T. Salles, M.A. Gonçalves, W.M. Jr., Word co-occurrence features for text classification, Inf. Syst. 36 (5) (2011) 843–858.
[50] G. Lev, B. Klein, L. Wolf, In defense of word embedding for generic text representation, in: Natural Language Processing and Information Systems - 20th International Conference on Applications of Natural Language to Information Systems, NLDB 2015 Passau, Germany, June 17–19, 2015 Proceedings, 2015, pp. 35–50.
[51] T. Salles, M.A. Gonçalves, V. Rodrigues, L.C. da Rocha, BROOF: exploiting out-of-bag errors, boosting and random forests for effective automated classification, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9–13, 2015, 2015, pp. 353–362.