Temporally-Aware Algorithms for Document Classification*

Thiago Salles
Fed. Univ. of Minas Gerais
Computer Science Dep.
Belo Horizonte, Brazil
tsalles@dcc.ufmg.br

Leonardo Rocha
Fed. Univ. São João Del Rei
Computer Science Dep.
of São João Del Rei, Brazil
lcrocha@ufsj.edu.br

Gisele L. Pappa
Fed. Univ. of Minas Gerais
Computer Science Dep.
Belo Horizonte, Brazil
glpappa@dcc.ufmg.br

Fernando Mourão
Fed. Univ. of Minas Gerais
Computer Science Dep.
Belo Horizonte, Brazil
fhmourao@dcc.ufmg.br

Wagner Meira Jr.
Fed. Univ. of Minas Gerais
Computer Science Dep.
Belo Horizonte, Brazil
meira@dcc.ufmg.br

Marcos Gonçalves
Fed. Univ. of Minas Gerais
Computer Science Dep.
Belo Horizonte, Brazil
mgoncalv@dcc.ufmg.br

ABSTRACT

Automatic Document Classification (ADC) is still one of the major information retrieval problems. It usually employs a supervised learning strategy, where we first build a classification model using pre-classified documents and then use this model to classify unseen documents. The majority of supervised algorithms consider that all documents provide equally important information. However, in practice, a document may be considered more or less important to build the classification model according to several factors, such as its timeliness, the venue where it was published in, its authors, among others. In this paper, we are particularly concerned with the impact that temporal effects may have on ADC and how to minimize such impact. In order to deal with these effects, we introduce a *temporal weighting function* (TWF) and propose a methodology to determine it for document collections. We applied the proposed methodology to ACM-DL and Medline and found that the TWF of both follows a lognormal. We then extend three ADC algorithms (namely kNN, Rocchio and Naïve Bayes) to incorporate the TWF. Experiments showed that the temporally-aware classifiers achieved significant gains, outperforming (or at least matching) state-of-the-art algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.4 [**Applications**]: Text processing;

General Terms

Algorithms, Experimentation

*This work was partially supported by CNPq, CAPES, FINEP, Fapemig, and INWEB.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR'10, July 19–23, 2010, Geneva, Switzerland.
Copyright 2010 ACM 978-1-60558-896-4/10/07 ...$10.00.

Keywords

Classification and Clustering, Text Mining

1. INTRODUCTION

Text classification is still one of the major information retrieval problems, and developing robust and accurate classification models continues to be a relevant demand, as a consequence of the increasing complexity and scale of current application scenarios, such as the Web. The task of Automatic Document Classification (ADC) aims to create models that associate documents with semantically meaningful categories, and these models are key for building spam filters and topic directories, identifying documents writing style, creating digital libraries, and guiding a user's search on the World Wide Web.

ADC usually follows a supervised learning strategy, in which a classification model is built using some training (pre-classified) documents and later employed to classify a new set of unseen documents. The majority of supervised algorithms consider that all documents provide equally important information. However, in practice, a document may be considered more or less important to build the classification model according to several factors, such as its timeliness, the venue where it was published in, its authors, among others.

In this work we are particularly concerned with the impact that temporal effects may have on ADC and how to minimize such impact. 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 of *Ant Colony Optimization* in the area of Artificial Intelligence, these terms became relevant for classifying Computer Science documents too. Previous work has demonstrated that temporal effects, such as the variation of the strenght of term-class relationship over time, may have a significant impact on ADC, and strategies for assessing these effects and their impact on ADC have already been devised [19].

In general, methods proposed to deal with temporal effects are based on three main approaches: instance selection, instance weighting, and ensembles. Instance selection [20] uses heuristics to decide which instances should be used (or the time intervals that contain those instances) to create a classification model. However, tuning these heuristics to select the most relevant documents is a challenge, since we

may easily gather too many or too few documents. Methods based on instance weighting [11] may ameliorate this problem [1]. However, this strategy raises the additional challenge of determining the weighting functions and their parameters, which are collection-dependent and usually performed in ad-hoc way. Finally, approaches based on ensembles, that correspond to the combination of various classification models generated from different classification algorithms, present the challenge of how to manage, efficiently, several models simultaneously [7].

This paper proposes a strategy to incorporate temporal models to document classifiers, aiming to address the two main drawbacks of instance selection and instance weighting approaches. Our strategy is based on the evolution of the term-class relationship over time, captured by a metric of *dominance*. We start by determining a *temporal weighting function* for a collection according to its characteristics. We found that this function follows a lognormal distribution for the datasets we used.

The next step is to incorporate the temporal weighting function to ADC algorithms and we propose two strategies that follow a lazy classification approach. In both strategies, the weights assigned to each example depend on the notion of a temporal distance δ, defined as the difference between the time of creation p of a training example and a reference time point p_r. The first strategy, named *temporal weighting on documents*, weights training instances according to δ. The second strategy, called *temporal weighting on scores*, is based on an ensemble of classifiers, one for each pair $\langle$class c,time point $p\rangle$. In this case, the scores (e.g., similarities, probabilities) returned by the respective classifiers for each pair $\langle c, p\rangle$ are weighted according to δ. The combined weighted scores are then used to take the final classification decision. We specifically show how these two strategies are implemented in three traditional ADC algorithms, namely, Rocchio, k Nearest Neighbors (KNN), and Naïve Bayes.

We evaluated our strategies using two actual digital libraries that span for decades, ACM-DL and MedLine, and achieved significant improvements on classification effectiveness for all classifiers. For instance, the temporal-aware version of Naïve Bayes outperformed by up to 10% the state-of-the-art classifier (SVM), while presenting an execution time up to hundreds of times faster.

2. RELATED WORK

Although document classification is a widely studied subject, the analysis of temporal aspects in this class of algorithms is quite recent – it has been studied just in the last decade. As previously mentioned, strategies to deal with these effects involve one or more of three approaches, namely: instance selection, instance weighting, and ensembles. Here we review the most relevant works of two broad areas where there has been significant efforts in terms of temporal effects on classification: adaptive document classification and concept drift.

Adaptive Document Classification [4] encompasses a set of techniques related to temporal aspects with the goal of improving the effectiveness of document classifiers through their incremental and efficient adaptation. Adaptive Document Classification brings three main challenges to text mining [17]. The first one, and most relevant to this research, is the notion of context and how it may be exploited towards better classification models. Previous research in document classification identified two essential forms of context: neighbor terms that are close to a certain keyword [14] and terms that indicate the scope and semantics of the document [2]. The second challenge is creating the models incrementally [10]. The third challenge has to do with the computational efficiency of the document classifiers. Methods for Adaptive Document Classification usually follow an instance selection approach, as they select semantic contexts based on, for instance, the co-occurrence of terms, not taking into account all documents in the training set.

Concept or topic drift [22] comprises another relevant set of efforts to deal with temporal effects in classification. To deal with concept drift, a prevailing approach in the literature is to completely retrain the classifier according to a sliding window. This involves instance selection and instance weighting techniques [12, 11, 23, 15]. The method presented in [12], for instance, maintains a window with documents sufficiently "close" to the current target concept and automatically adjusts the window size so that the estimated generalization error is minimized. In [11], the methods presented either maintain an adaptive time window on the training data, select representative training examples, or weight the training examples. In [23] the authors describe a set of algorithms that react to concept drift in a flexible way and can take advantage of situations where contexts reappear. The main idea of these algorithms is to keep only a window of currently trusted examples and hypothesis, and store concept descriptions in order to reuse them if a previous context reappears. Unlike previous works, which use a single window to determine drift in the data, in [15] the authors present a method that uses three windows of different sizes to estimate the change in the data. While algorithms that use a window of fixed size impose hard constraints over drift patterns, those that use heuristics to adjust the window size to the current extent of concept drift often involve lots of parameters to be calibrated. The approach proposed in this paper relies on statistical properties of the collection to assess the temporal effects, solving such drawbacks, while promoting a high quality classification.

Other common approach to deal with 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 [21, 13, 7]. In [21], the authors propose a boosting-like method to train a classifier ensemble from data streams. It naturally adapts to concept drift and allows to quantify the drift in terms of its base learners. The algorithm was shown to outperform learning algorithms that ignore concept drift. In this same direction, Kolter et al. [13] present a technique that maintains an ensemble of base learners, predicts instance classes using a weighted-majority vote of these "experts", and dynamically creates and deletes experts in response to changes in performance. In [7], a method that builds an ensemble of classifiers using Genetic Programming (GP) to inductively generate decision trees is presented. However, how to manage, efficiently, several models simultaneously remains a challenge. To address such drawback, we propose an approach based on the combination of vari-

[1]The first strategy may be seem as an instance weighting strategy with binary weights.

ous classification models, but with a simpler way to manage them.

Another approach that can be easily compared to ours, and is based on instance selection, is the one proposed in [20]. In [20], the authors introduce the concept of *temporal context*, defined as a subset of the documents collection that minimizes the impact of temporal effects in classifiers' performance. An algorithm named *Chronos* was proposed to identify these contexts based on the stability of the terms in the training set. The temporal contexts were then used to sample the training documents for the classification process. Hence, training documents that were considered to be outside the temporal context were discarded by the classifier.

In contrast with the aforementioned works, here we propose an approach to classify documents in scenarios where we may have information about both the past and the future, and this information may change over time. It should be noticed, however, that our approach may be easily adapted for scenarios where we only have past information, such as Adaptive Document Classification and Concept Drift. Moreover, we address the drawbacks of which instances to select by approximating a *temporal weighting function* using a lognormal distribution, and may easily tune its parameters using statistical methods.

3. TEMPORAL WEIGHTING FUNCTION

As mentioned before, the potential impact that certain temporal effects have on term-class relationships may have a great influence on the results of the classification process, as showed in [19]. Thus, incorporating information about these changes into the classification process has the potential to improve its effectiveness.

We address this issue through a temporal weighting function (TWF) that quantifies the influence of a training document while classifying a test document, as a function of the temporal distance between their creation times. We distinguish two major steps in determining such function: its expression and its parameters. The expression is usually harder to determine, since it may express the generative process behind the function, 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 w.r.t. the strength of term-class relationships. One metric that expresses such strength is the *dominance* [20], since the more exclusive a term is to a given predefined class, the stronger this relationship. Dominance can be formally defined as:

$$Dominance(t, c) = \frac{N_{tc}}{\sum_{c'} N_{tc'}},$$

where N_{tc} stands for the number of documents in class c that contain term t.

For ease of understanding, before we continue the discussion about the temporal weighting function, we describe the two document collections for which we want to determine the functions: ACM Digital Library (ACM-DL) and the MedLine. The ACM-DL has 24.897 documents containing articles related to Computer Science created between the years of 1980 and 2002. We considered only the first level of the taxonomy adopted by ACM, including 11 categories, which did not vary during this period of time. The second one is derived from the MedLine collection, and has 861.454 documents, classified into 7 distinct classes related to Medicine and created between the years of 1970 and 1985. In both collections, each document is assigned to a single class.

We start by defining the temporal granularity of the weighting function, which should be the minimum time interval between relevant changes in the collection (e.g., days, weeks, or years). Since both collections contain scientific documents, it is intuitive that a year granularity is representative, once documents are usually published yearly (scientific conferences are usually annual).

The simplest approach would be to use a pulse function at temporal distance 0, that is, the pulse magnitude is proportional to the term dominance associated with the training documents produced in the same year of the test document. However, as pointed by [19], considering a larger time interval instead of a single time point is better, since the influence decreases with the increase of the temporal distance. We then need to determine the time period that must be considered, which we call stability period. Notice that each term may present a different stability period for each year when it occurred in the collection. We first determine the stability period for each term and then combine them, as follows.

One approach for the first step is presented in [20], where a stability period $S_{t,r}$ of a term t, considering the reference time point 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 collections ACM-DL and Medline, 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.

We then combine the stability periods $S_{t,r}$ for each term t and each reference time point p_r in the collection. A difficulty in this case is related to the fact that a term may present different stability periods for different reference years. In order to avoid this problem, 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 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 S'_t as the set of temporal distances that occur on the stability periods of term t (considering all reference moments r, then $S'_t = \{\delta \leftarrow p_n - p_r | \forall r p_n \in S_{t,r}\}$. Making the stability periods easily comparable is important because our real interest is to know what kind of distribution this temporal distances follow w.r.t. 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 1 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 to assess the independence of each RV_i and RV_j, $\forall i \neq j$ [3], where, as

	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 1: Temporal distances versus terms

mentioned, each RV represents the occurrence of a temporal distance δ for a term t.

We applied this test to both ACM-DL and Medline and obtained a p-value of 0.99 through a Monte Carlo simulation, which allows us to state that the random variables considered are indeed independent. Thus, the observed variability of occurrences of δ for different terms is a result of independent effects [16]. However, it is still not clear whether the effects responsible for the observed variability can be additive (leading to a normal distribution) or multiplicative (leading to a lognormal distribution). We then apply a statistical normality test. According to D'Agostino's D-Statistic Test of Normality [6], with 0.01 significance level, we found that the lognormal distribution best fits both the ACM-DL and Medline collections, as presented in Table 3.

Consider that the RV D_δ related to the occurrences of δ, which represents the distribution of each δ_i over all terms t, is lognormally distributed if lnD_δ is normally distributed. More generally, since 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, converges to a lognormal distribution [5]. For a lognormal distribution, the asymptotically most efficient method for estimating its associated parameters relies on a log-transformation [16]. Using a Maximum Likelihood method, we estimated those parameters for both collections, and then back-transformed them, as shown in Table 2. We considered a 3-parameter gaussian function, $F = a_i e^{-\frac{(x-b_i)^2}{2c_i^2}}$. The parameter a_i is the height of the curve's peak, b_i is the position of the centre 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 distinct natures. We performed two curve fitting procedures, considering a single gaussian F and a mixture of two gaussians, 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 2, along with the goodness of fitting measure *Adjusted-*R^2. The *Adjusted-*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 modelling of the Temporal Weighting Function as a lognormal distribution an effective strategy. To account for the problem faced when the scale of the score is not compatible with the algorithm input, we include a scaling factor $\beta \in \mathbb{R}$, that is algorithm specific and will be defined in Section 5.

Figure 1 shows the distribution of temporal scores, when

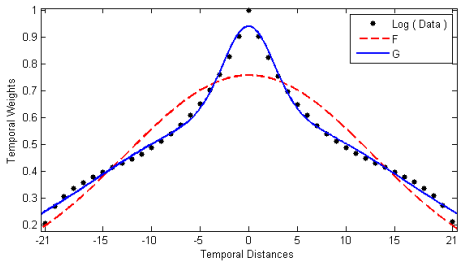
Param.	ACM-DL		Medline	
	Value	Conf. Interval	Value	Conf. Interval
a_1	0.325	(0.288, 0.362)	0.089	(0.066, 0.113)
b_1	-0.028	(-0.309, 0.253)	-0.013	(-0.349, 0.324)
c_1	3.636	(3.117, 4.154)	1.635	(1.099, 2.17)
a_2	0.616	(0.589, 0.643)	0.901	(0.891, 0.911)
b_2	0.037	(-0.395, 0.470)	0.092	(-0.130, 0.314)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

Table 2: Estimated parameters for both collections, with 99% confidence intervals.

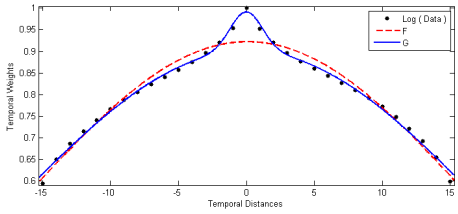
$\beta = 1$, for each possible temporal distance between the creation time of test document d' and the training documents for both the ACM-DL and the Medline collections.

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

Table 3: D'Agostino's D-Statistic Test of Normality. Bold-face for tests that we can not reject the null hypothesis of normality.



(a) ACM-DL Collection



(b) MedLine Collection

Figure 1: Fitted temporal weighting function with log-transformed data.

4. TEMPORALLY-AWARE ADC

This section shows how three well-known text classifiers, namely Rocchio, KNN and Naïve Bayes [18], can be modified to take into account the temporal weighting function defined in Section 3. The three algorithms are modified following two strategies: temporal weighting on documents and temporal weighting on scores, as detailed below.

4.1 Temporal Weighting on Documents

The temporal weighting on documents strategy weights each training document by the temporal weighting function according to its temporal distance to the test document d', as detailed next.

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. For example, while Rocchio and KNN classify new instances

based on a distance metric, Naïve Bayes is a probabilistic classifier that assigns to a test document the most probable class that would have generated d', adopting some naïve assumptions such as positional and conditional independence of terms.

In the case of 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 (TF-IDF vector) by its associated temporal weight. In the case of the Naïve Bayes, the temporal function can be used to weight the impact of each training example in both the a priori and conditional probabilities, in order to generate a more accurate a posteriori probability.

Rocchio Rocchio is an eager classifier that uses the centroid of a class to find boundaries between classes. As an eager classifier, Rocchio does not require any information from d' to create a classification model. Hence, we will have to adapt it to become a lazy classifier when using the temporal weighting function, since the weights depends on the creation time of a test document.

The centroid of a class is defined as the average value of all its training examples. When classifying a new document d', Rocchio associates it to the class represented by the centroid closest to d'. In order to make Rocchio a lazy classifier, we have to change the separation boundaries of classes according to the temporal weights produced by our function.

Hence, it needs to calculate each Rocchio's class centroid based on the creation time p_r of a test document d'. Consider the set of training documents d of class c. This set can be partitioned into subgroups of documents created at the same temporal distance δ from p_r, i.e., subgroups of documents created at a temporal distance of 1 or -1 , 2 or -2, and so on. The centroid $\overrightarrow{\mu}_c$ for class c is 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 time point of d and d'. Thus, a centroid $\overrightarrow{\mu}_c$ is given by:

$$\overrightarrow{\mu}_c = \frac{1}{\|D_c\|} \sum_{d \in D_c} \left(\sum_{\delta \in \Delta} \overrightarrow{d_\delta} \cdot TWF(\delta) \right),$$

where D_c is the number of documents in class c, Δ is the set of all possible temporal distances between the training documents and the test document d', and $\overrightarrow{d_\delta} \in D_c$ is a training document with temporal distance δ from d'.

This approach redefines the centroid's coordinates in the vectorial space considering document's representativeness on class c w.r.t. the reference time point p_r. Both training and classification procedures are presented in Algorithm 1.

Algorithm 1 Rocchio-TWF-Doc: Rocchio with Temporal Weighting on Documents

```
1: function Train(ℂ, 𝔻)
2:     D_{c,δ} ← {d : ⟨d, ⟨c, δ⟩⟩ ∈ 𝔻}
3:     μ⃗_c ← 1/‖D_c‖ Σ_{d∈D_c} ( Σ_{δ∈Δ} d⃗_δ · TWF(δ) )
4:     return {μ⃗_c : c ∈ ℂ}
5: function Classify({μ_c : c ∈ ℂ}, d′)
6:     return arg max_c cos(μ⃗_c, d⃗′)
```

KNN KNN 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 temporality.

Let s be the cosine similarity between a training document d and d'. If d is similar to d' but is temporally distant, 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 2.

Algorithm 2 KNN-TWF-Doc: KNN with Temporal Weighting on Documents

```
1: function getKNearestNeighbors(𝔻, d′, k)
2:     for each d ∈ 𝔻 do
3:         sim(d, d′) ← cos(d, d′) · TWF(δ)
4:         priorityQueue.insert(sim, d)
5:     return priorityQueue.first(k)
6: function Classify(𝔻, d′, k)
7:     knn ← getKNearestNeighbors (d′, D, k)
8:     return {arg max_c Σ_c knn.nextDoc(c)}
```

Naïve Bayes Naïve Bayes is a probabilistic learning method that aims to infer a model for each class, assigning to d' the class associated to the most probable model that would have generated d'. Here we adapt the Multinomial Naïve Bayes approach [18], since it is widely used for the probabilistic text classification. Similarly to the previously defined "temporal weighting on documents" approaches, here 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 time point p, N_p is the number of training documents created at time point p, f_{tcp} stands for the frequency of occurrence of term t in training documents of class c that were created on time point p and, finally, δ denotes the temporal distance between time point p and the creation time of d'.

The main goal of this strategy is to reduce the impact that temporally distant information have when estimating a posteriori probabilities. Algorithm 3 presents this strategy.

4.2 Temporal Weighting on Scores

A more sophisticated approach to exploit the temporal weighting function considers the "scores" produced by the traditional classifiers, as listed in Algorithm 4. 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

Algorithm 3 Naïve Bayes TWF-Doc: Naïve Bayes with Temporal Weighting on Documents

1: **function** CLASSIFY($\mathbb{D}, d'$)
2: $aPriori[c] \leftarrow \frac{\sum_p (N_{cp} \cdot TWF(\delta))}{\sum_p (N_p \cdot TWF(\delta))}$
3: $termCond[c] \leftarrow \prod_{t \in d'} \frac{\sum_p (f_{tcp} \cdot TWF(\delta))}{\sum_p \sum_{t' \in \mathbb{V}} (f_{t'cp} \cdot TWF(\delta))}$
4: **return** $\{\arg\max_c \eta \cdot aPriori[c] \cdot termCond[c]\}$

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 Naïve Bayes. From now on, we refer to this approach as *temporal weighting on scores*.

Let $\mathbb{C}$ and $\mathbb{P}$ be the set of classes and creation time points of the training documents. First, each training document class $c \in \mathbb{C}$ is associated with the corresponding creation time point $p \in \mathbb{P}$, generating a new class defined as $\langle c, p \rangle$. Then, we use a traditional classification algorithm to generate scores for each new class $\langle c, p \rangle$. Note that this scenario isolates term-class relationship variations, since it considers only one time point. To decide to which class c the document d' should be assigned to, we sum up all the scores $\langle c, p \rangle$, for all $p_r \in \mathbb{P}$, weighting them by the $TWF(\delta)$, where $\delta = p - p_r$ corresponds to the temporal distance between p and the creation moment of d'. At the end of this process, d' will be assigned to the class c with highest score, as listed in Algorithm 4.

Algorithm 4 TWF-Sc: Temporal Weighting on Scores

1: **function** CLASSIFY(d', $\mathbb{C}$, $\mathbb{P}$, $\mathbb{D}$)
2: **for each** $c \in \mathbb{C}$ **do**
3: **for each** $p \in \mathbb{P}$ **do**
4: $c' \leftarrow \langle c.p \rangle$
5: $score[c] += $TRADITIONALCLASSIFIER$(d', \mathbb{D}, c') \cdot TWF(\delta)$
6: **return** $\{\arg\max_c score\}$

5. RESULTS

In order to evaluate the impact that the proposed TWF has on the classification task, we evaluate both the traditional and temporally-aware versions of Rocchio, KNN and Naïve Bayes in the ACM-DL and Medline collections, and contrast them. The methods were compared using two standard information retrieval measures: Accuracy and macro average F_1 (MacroF_1). While the Accuracy measures the classification effectiveness over all decisions, the MacroF_1 measures the classification effectiveness for each individual class and averages them. All experiments were executed using a 10-fold cross-validation [1] 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.

5.1 Parameter settings

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. Four values were tested for each version of the traditional and temporally-aware algorithms: 30, 50, 150, and 200. For the traditional version of the algorithm $k = 30$ presented better results, while for both temporally-aware versions of KNN the best value 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 of misclassification increases. When considering temporal information by means of the proposed temporal weights, in contrast, more consistent information becomes available, allowing a more accurate model.

As discussed in Section 4, the TWF scale must be compatible with the classifiers scores, ensuring that it effectively improves the classifier's decision rules without dismissing them. We empirically tested three values for β: 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 Naïve Bayes, the best value was $\beta = 10$. This is due to the multiplicative nature of this classifier: many conditional probabilities are multiplied, leading to even smaller values.

5.2 Experiments

After setting the parameters, we perform experiments comparing the traditional and the proposed temporally-aware versions of Rocchio, KNN and Naïve Bayes. The results for the ACM-DL and Medline collections are reported in Tables 4 and 5. 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 accuracy ("acc.") 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.

As we can see in Tables 4 and 5, all modified versions of Rocchio and KNN achieved better results than the baseline in ACM-DL. In Medline, the versions *on score* achieved gains, while the versions on documents were statistically the same as the baseline. In particular, Rocchio with TWF on scores presents the most significant improvements in both collections, with gains up to +18.87 and +11.46 for MacroF_1 and Accuracy, respectively. Similarly, KNN with TWF on scores achieves the best results among all KNN variations, with gains of +8.85% and +3.80% for MacroF_1 and Accuracy in the ACM-DL collection. 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 time points, and 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 are placed closer to the example being classified.

The Naïve Bayes with TWF on documents presents better results for MacroF_1 on both ACM-DL and Medline, and better Accuracy in Medline. Note that the best improvement was achieved in MacroF1, pointing out that this strategy effectively reduces the Naïve Bayes bias towards the most

Algorithm	Rocchio		KNN		Naïve Bayes	
Metric	macF$_1$(%)	acc.(%)	macF$_1$(%)	acc.(%)	macF$_1$(%)	acc.(%)
Baseline	55.12	66.42	61.80	72.29	57.33	73.69
TWF *on documents*	57.24 (+3.85) ▲	68.64 (+3.34) ▲	63.40 (+2.59) ▲	73.98 (+2.34) ▲	60.78 (+6.02) ▲	74.14 (+0.61) ●
TWF *on scores*	59.07 (+7.17) ▲	71.54 (+7.71) ▲	64.18 (+3.85) ▲	74.56 (+3.14) ▲	40.62 (-41.14) ▼	48.76 (-51.13) ▼

Table 4: Results obtained when incorporating TWF to Rocchio, KNN, and Naïve Bayes – ACM-DL

Algorithm	Rocchio		KNN		Naïve Bayes	
Metric	macF$_1$(%)	acc.(%)	macF$_1$(%)	acc.(%)	macF$_1$(%)	acc.(%)
Baseline	55.36	70.17	73.33	84.63	76.81	84.69
TWF *on Documents*	56.21 (+1.53) ●	70.83 (+0.94) ●	73.94 (+0.83) ●	84.07 (+0.66) ●	81.58 (+6.21) ▲	87.48 (+3.29) ▲
TWF *on scores*	65.81 (+18.87) ▲	78.21 (+11.46) ▲	76.82 (+4.54) ▲	87.01 (+2.81) ▲	51.54 (-49.03) ▼	48.02 (-76.36) ▼

Table 5: Results obtained when incorporating TWF to Rocchio, KNN, and Naïve Bayes – Medline

frequent classes. However, in contrast with Rocchio and KNN, the Naïve Bayes with TWF on scores performs poorly in both collections. We attribute this to two major weaknesses of traditional Naïve Bayes version. First, when facing skewed data distributions, Naïve Bayes unwittingly prefers larger classes over others, causing decision boundaries to be biased. Second, when data is scarce, there is not enough information to perform accurate estimates, leading to bad results.

The skewness of data distribution among classes $\langle c, p\rangle$ 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 Naïve Bayes, we sampled MedLine, creating two sub-collections composed by the least and most frequent classes $\langle c, p\rangle$, 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 5 and 6, the greater the CV, the worse are the results.

ACM-DL has an even more skewed data distribution over each time point, preventing us to sample it in sub-collections with smaller CV. Figure 2 shows that in the ACM-DL collection data scarcity is also prominent, contributing to the poor performance. Notice that 70% of classes $\langle c, p\rangle$ have less than 100 documents, a number too low to guarantee accurate estimates.

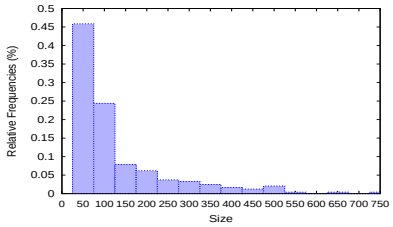


Figure 2: $\langle c, p\rangle$ sizes – ACM-DL

As observed, using TWF on scores in most cases led to better results than those applying TWF on documents. We believe this is because, in many applications, terms present distinct evolutive patterns, and the proposed function neglects this fact. Hence, when the temporal TWF is applied on documents, all terms are multiplied by the same score (i.e., using a fine-grained representation of the document), given in function of the temporal distance δ. Thus, we consider an uniform evolution among terms. In contrast, the TWF on scores minimizes this problem, as it applies the score in a coarse-grained representation of a document. Hence, one could argue that the TWF is more suitable to the "on scores" approach. The definition of a temporal weighting for each term independently is left for future work.

Finally, we also compared the best version of the methods previously proposed, i.e., KNN with TWF on scores and Naïve Bayes with TWF on documents, to the state of the art classifier *Support Vector Machine* [8], in terms of effectiveness (classification quality) and efficiency (execution time). We run an efficient SVM implementation, SVM_Perf [9], which is based on the maximum-margin approach and can be trained in linear time. We used an one-against-all [18] methodology to adapt binary SVM to multi-class classification, since the collections present 11 (ACM-DL) and 7 (MedLine) classes. The results are presented in Table 7. For ACM-DL collection, the significant gains are of 3.74% and 2.66% in macroF$_1$ and accuracy, respectively. For MedLine collection, the most significant gains are of 12.87% and 5.06% in macroF$_1$ and accuracy, respectively. Considering that SVM is a state of the art classifier, and that both collections are very unbalanced (which significantly difficulties classification), our results evidence the quality of the proposed solution, with better performance.

6. CONCLUSION AND FUTURE WORK

This work discussed the impact that temporal effects may have in ADC, and proposed two new strategies for instance weighting that leads to more accurate classification. We started by proposing a methodology to model a Temporal Weighting Function (TWF) that captures changes in term-class relationships for a given period of time. For our real datasets, we showed that TWF follows a lognormal distribution, whose parameters may easily be tuned using statistical methods. In order to incorporate this TWF to classifiers, we presented two approaches: TWF on documents and TWF on scores. TWF on documents weights each training document by the TWF according to its temporal distance to the test document. TWF on scores, in contrast, takes into account scores produced by the traditional classifiers on scenarios without temporal variability on term-class relationships, performing a weighted sum of them, where the weights come from the TWF. Both strategies were incorporated to three traditional classifiers, namely Rocchio, KNN, and Naïve Bayes.

Results with the traditional versions of these classifiers and the temporally-aware ones showed that considering temporal information significantly improves the results of the

Naïve Bayes	Least frequent classes $\langle c, p \rangle$			Most frequent classes $\langle c, p \rangle$		
Metric	CV	macF$_1$(%)	acc.(%)	CV	macF$_1$(%)	acc.(%)
Baseline	0.57	87.32	88.10	0.43	91.09	91.92
TWF on Scores		92.03 (+5.40) ▲	92.21 (+4.66) ▲		93.19 (+2.31) ▲	93.86 (+2.12) ▲

Table 6: Results obtained for the least and most frequent classes $\langle c, p \rangle$ sampling for Naïve Bayes – Medline

Collection	ACM-DL			Medline		
Metric	macF$_1$(%)	acc.(%)	Time (s)	macF$_1$(%)	acc.(%)	Time (s)
SVM	60.07	73.03	4200	72.28	83.27	1300000
KNN with TWF on scores	64.18 (+6.84) ▲	74.56 (+2.09) ▲	62	76.82 (+5.90) ▲	87.01 (+4.49) ▲	4757
Naïve Bayes with TWF on documents	60.78 (+1.18) •	74.14 (+1.52) •	51	81.58 (+12.87) ▲	87.48 (+5.06) ▲	2840

Table 7: Results obtained when adding TWF on scores to KNN and TWF on Documents to Naïve Bayes versus SVM for both collections

traditional classifiers. Also, both temporally-aware KNN and Naïve Bayes achieved better results than SVM, with better performance. Considering that SVM is a state of the art classifier, and that both collections are very unbalanced, our results evidence the quality of our solution, coupled with an efficient implementation.

Given the results obtained when comparing SVM to the temporal versions of KNN and Naïve Bayes, as a future work, we will incorporate temporal information to the SVM classifier, by defining kernel functions that use the proposed TWF. We also plan to refine our TWF, in order to account for distinct evolutive patterns that terms may present, by means of a more sophisticated statistical analysis. Moreover, similarly to time, social and geographical aspects may induce variations on term-class relationships, and we will also explore those dimensions in order to achieve an even better classification quality.

7. REFERENCES

[1] L. Breiman and P. Spector. Submodel selection and evaluation in regression – the x-random case. *International Statistical Review*, 60:291–319, 1992.

[2] N. H. M. Caldwell, P. J. Clarkson, P. A. Rodgers, and A. P. Huxor. Web-based knowledge management for distributed design. *IEEE Intelligent Systems*, 15(3):40–47, 2000.

[3] D. B. Clarkson, Y.-a. Fan, and H. Joe. A remark on algorithm 643: Fexact: an algorithm for performing fisher's exact test in r x c *ACM Trans. Math. Softw.*, 19(4):484–488, 1993.

[4] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Trans. Inf. Syst.*, 17(2):141–173, 1999.

[5] S. K. Crow EL. *Log-normal distributions: Theory and application*. New York: Dekker, December 1988.

[6] P. E. D'Agostino R.B. Tests for departure from normality. *Biometrika*, 60:613–622, 1973.

[7] G. Folino, C. Pizzuti, and G. Spezzano. An adaptive distributed ensemble approach to mine concept-drifting data streams. In *ICTAI '07, Volume 2*, pages 183–188, Washington, DC, USA, 2007. IEEE Computer Society.

[8] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[9] T. Joachims. Training linear svms in linear time. In *Proc. of the 12th ACM SIGKDD Conference*, pages 217–226, New York, NY, USA, 2006. ACM.

[10] Y. S. Kim, S. S. Park, E. Deards, and B. H. Kang. Adaptive web document classification with mcrdr. In *ITCC '04, Volume 2*, page 476, Washington, DC, USA, 2004. IEEE Computer Society.

[11] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3):281–300, 2004.

[12] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In P. Langley, editor, *ICML '00*, pages 487–494, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[13] J. Kolter and M. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. Technical report, Department of Computer Science, Georgetown University, Washington, DC, June 2003.

[14] S. Lawrence and C. L. Giles. Context and page analysis for improved web search. *IEEE Internet Computing*, 2(4), 1998.

[15] M. M. Lazarescu, S. Venkatesh, and H. H. Bui. Using multiple windows to track concept drift. *Intell. Data Anal.*, 8(1):29–59, 2004.

[16] E. Limpert, W. A. Stahel, and M. Abbt. Log-normal distributions across the sciences: Keys and clues. *BioScience*, 51(5):341–352, 2001.

[17] R. Liu and Y. Lu. Incremental context mining for adaptive document classification. In *Proc. of the 8th ACM SIGKDD*, pages 599–604. ACM Press, 2002.

[18] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[19] F. Mourao, L. Rocha, R. Araújo, T. Couto, M. Gonçalves, and W. Meira Jr. Understanding temporal aspects in document classification. In *Proc. of the WSDM '08*, 2008.

[20] L. Rocha, F. Mourao, A. Pereira, M. A. Gonçalves, and W. Meira Jr. Exploiting temporal contexts in text classification. In *Proc. of the CIKM '08*, 2008.

[21] M. Scholz and R. Klinkenberg. Boosting classifiers for drifting concepts. *Intell. Data Anal.*, 11(1):3–28, 2007.

[22] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical report, Department of Computer Science, Trinity College, Dublin, Ireland, December 2004.

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