Automatic Document Classification Temporally Robust

Thiago Salles[1]*, Leonardo Rocha[2], Fernando Mourão[1], Gisele L. Pappa[1], Lucas Cunha[1], Marcos André Gonçalves[1], Wagner Meira Jr.[1]

[1] Dep. de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos 6627 - ICEx - 31270-010 Belo Horizonte, Brasil
{tsalles,fhmourao,glpappa,lucmir,mgoncalv,meira}@dcc.ufmg.br
[2] Dep. de Ciência da Computação - Universidade Federal de São João Del Rei (UFSJ)
Pc. Dr. Augusto Chagas Viegas 17 - DCOMP - 36300-088 São João Del Rei, Brasil
lcrocha@ufsj.edu.br

Abstract. The widespread use of the Internet has increased the amount of information being stored on and accessed through the Web. This information is frequently organized as textual documents and is the main target of search engines and other retrieval tools, which have to classify documents, among other tasks. Automatic Document Classification (ADC) associates documents to semantically meaningful categories, and usually employs a supervised learning strategy, where we first build a classification model using pre-classified documents, and then use the model to classify new documents. One major challenge in building classifiers is dealing with the temporal evolution of the characteristics of the documents and the classes to which they belong. However, most of the current techniques for ADC do not consider this evolution while building and using the models. Previous studies show that the performance of classifiers may be affected by three different temporal effects (class distribution, term distribution and class similarity). In this paper, we propose a new approach that aims to minimize the impact of temporal effects through a *Temporal Adjustment Factor*, in order to devise temporally robust classifiers based on traditional ones (Rocchio and KNN). Experimental results obtained using two real and large textual collections point to significant gains up to 11% of the temporal-aware versions of the classifiers over their traditional counterparts, and up to 4% compared to SVM (with a significantly lower runtime).

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data mining

General Terms: Algorithms, Experimentation

Keywords: automatic document classification, temporal effects

1. INTRODUCTION

The widespread use of the Internet has increased the amount of information being stored on and accessed through the Web. This information is frequently organized as textual documents and is the main target of search engines and other retrieval tools, which have to classify documents, among other tasks. Automatic Document Classification (ADC) [Borko and Bernick 1963] associates documents to semantically meaningful categories. Classification supports and enhances several tasks such as automated topic tagging (i.e., assigning labels to documents), building topic directories, identifying documents writing style, creating digital libraries, improving the precision of Web searching, or even helping users to interact with search engines. Other relevant application scenarios include spam filtering, detection of adult content and plagiarism.

ADC usually employs a supervised learning strategy, where we first build a classification model

*Corresponding author.

This research is partially funded by the MCT/CNPq/CT-INFO project InfoWeb (grant number 55.0874/2007 − 0), by InWeb – The National Institute of Science and Technology for the Web (MCT/CNPq/FAPEMIG grant number 573871/2008 − 6), and by the authors' individual research grants from FAPEMIG, FINEP and CNPq.

using pre-classified documents, i.e., a training set, and then use the model to classify new documents. Building text classification models usually consists of determining a set of characteristics (e.g., terms) and a function that combines them so that the resulting model is able to identify classes of documents. One major challenge in building classifiers is dealing with the temporal evolution of the characteristics of the documents and the classes to which they belong, as a consequence of events such as the creation of new documents, the introduction of new terms, the rise of new fields, and the partition of large fields into more specialized sub-fields. Intuitively, it is interesting to notice that the differences in terms of characteristics, which once were useful for discriminating classes, may be either amplified or reduced, affecting the precision of the classifier that exploits them.

Despite the potential impact of the temporal evolution on the quality of classification models and its recognition as a determinant factor for the model quality, most of the current techniques for ADC do not consider this evolution while building and using the models. In a recent work [Mourão et al. 2008], the authors distinguish three different temporal effects that may affect the performance of automatic classifiers. The first effect is the class distribution variation, which are the temporal-related changes in the class frequencies. The second effect is the term distribution variation, which are the changes in the relationships between terms and classes across time. The third effect is the class similarity variation, which is how the similarity among classes, as a function of the terms that occur in their documents, changes over time. These three effects demonstrated that accounting for the temporal evolution of documents poses a challenge to the classification model, which is usually less effective when it neglects such factors. The same work also found that the greater the temporal distance between the moment of creation of a training document d and the creation moment of a new document to be classified d', the greater is the disparity between the characteristics of the documents from each moment and, therefore, the smaller is the representativeness of d for classifying a new document d'. The representativeness of a training document depends on the extent to which it reflects the characteristics of a collection at a given moment.

In this paper, we propose a new approach that aims to minimize the impact of temporal effects through a *Temporal Adjustment Factor*, which is used to weight training documents (represented as vectors in the p-dimensional vectorial space, or aggregated in scores produced by traditional classifiers at specific creation moments, as discussed in Section 4). The adjustment factor of a document d is a function of its creation moment and the creation moment of d' to be classified. Using this factor, we also propose temporal-aware extensions of two classifiers that are based on the vector space model, *Rocchio* and *K-Nearest Neighbors* (KNN). The first is a centroid-based classifier that calculates representative vectors of each class to define the class boundaries. Those representative vectors (centroids) are given by the average vector of the training documents assigned to the represented class. The centroid's coordinates are weighted according to the temporal distance between them and the document to be classified, thus redefining the class boundaries according to temporal effects. The second classifier considers each training document independently, selecting the k most similar ones to the test document d' in order to determine which class must be assigned to d'. Each vector, which represents the documents in the vector space, are properly scaled using the proposed weighting schema, moving away temporally distant training documents from the document being classified, thus increasing the effectiveness of the k nearest neighbors selection.

We apply the temporal classifiers to two distinct reference collections, the first derived from the digital library of the ACM (ACM-DL), containing documents about Computer Science, and the second from MedLine, a Medicine digital library. We achieved significant gains in accuracy in both algorithms. The temporal classifier based on KNN outperformed the state of the art classifier *Support Vector Machine* (SVM), presenting a significantly lower runtime. As discussed in Section 5, the use of the temporal adjustment factor is a major source of performance improvement. Moreover, there is still room for further improvements, since the temporal adjustment factor may be better defined and provide even better results.

The remainder of this paper is organized as follows: Section 2 discusses some related work. Section 3 presents the temporal adjustment factor and its application to ADC. Section 4 presents the proposed temporally robust classifiers, based on Rocchio and KNN, which use our adjustment factor. In Section 5 we evaluate our approach and, finally, in Section 6 we conclude and discuss future work.

2. RELATED WORK

Although automatic document classification is a widely studied subject, the characterization of temporal effects on textual collections and the strategies to handle them in ADC are recent and under investigation. In this section, we discuss two broad areas where the temporal aspects are explored: Adaptive Document Classification and Concept Drift.

Adaptive Document Classification [Cohen and Singer 1999] embodies a set of techniques to overcome the problems related to temporal aspects, improving the effectiveness of classifiers. Those improvements are achieved through an incremental and efficient adaptation of the classification models [Liu and Lu 2002], and bring at least three major challenges for text mining. The first is the definition of a context, and how it can be exploited to devise better classification models. A context is a semantically significant set of documents, and previous studies suggest that contexts may be determined through, at least, two strategies: identification of neighbor terms to a keyword [Lawrence and Giles 1998] and identification of terms that indicate the scope and semantics of documents [Caldwell et al. 2000]. The second challenge is how to build models incrementally [Kim et al. 2004], and, finally, the third challenge is the computational efficiency of the resulting classifiers.

Concept Drift [Tsymbal 2004] is characterized by temporal changes of both the concepts used in the documents and users' interests, leading to inconsistent classification models as a consequence of old (obsolete) data. Studies related to Concept Drift aim to identify these changes [Dries and Rückert 2009] and keep the model up-to-date with the current concepts. There are three common strategies for dealing with concept drift. The first strategy is to retrain the classifier using a subset of the training documents that are considered more relevant, inside a sliding window of size m (fixed or not) [Klinkenberg and Joachims 2000; Klinkenberg 2004; Lazarescu et al. 2004]. The second strategy weights each training document by a utility function that penalizes obsolete data, prioritizing the most up-to-date documents [Klinkenberg 2004; Widmer and Kubat 1996]; Finally, the third strategy combines several diverse classification models, such as boosting, to classify new documents, pruning or adjusting the weights according to recent data [Scholz and Klinkenberg 2007; Kolter and Maloof 2003; Folino et al. 2007].

Although the aforementioned areas deal with changes in concept or interest in the data across time, these methods do not characterize in detail the variations of the collections' characteristics across time and how they affect the classification task. In this direction, in [Mourão et al. 2008] the authors identify and characterize three distinct temporal effects that may affect the classification effectiveness: class' distribution, terms' distribution and inter-class similarity. Furthermore, the authors investigate how these factors may be used to improve the classification task.

Based on these identified temporal effects, in [Rocha et al. 2008] the authors also introduce the concept of temporal context in order to deal with the temporal evolution of the collections. A temporal context is defined as a subset of documents that minimizes the impact of temporal effects on the classification quality. Further, the *Chronos* algorithm was proposed aiming at identifying those contexts based on the terms' stability on training documents. More specifically, given a test document d' composed by terms t_i, it selects, for the training set, those documents that contain the terms t_i and were created within the stability period associated with t_i (i.e., a time period during which the temporal characteristics of the training set vary within a maximum threshold θ). Unlike other strategies, which do not consider training documents out of the temporal context, our proposed solution considers all available information, reducing the impact of temporal effects through a temporal adjustment factor.

This factor enables us to consider, in a distinctive manner, documents created at different moments, minimizing the impact of temporal evolution on the classifier's effectiveness without missing valuable information regarding the underlying classes' distribution.

3. TEMPORAL ADJUSTMENT FACTOR

In this section we discuss how to account for the time effects in classification models. As mentioned before, documents from different moments may present different characteristics regarding classes' distribution, terms usage, and inter-class similarity [Mourão et al. 2008]. Such variations are due to several aspects, such as an increased interaction between different subjects, natural changes in language, and the emerging of new concepts, among others. For example, consider the sets of terms *pheromone* and *ant colony*, which were almost exclusively associated with the subject of natural sciences. Up to the 1990s, when "Ant Colony Optimization" was proposed, those sets of terms became more common in Computer Science documents. Although this scenario seems to be unusual, it illustrates a more common case where there are subtle variations over time that, combined, make the task of classification harder.

The evolutive behavior of the meaning of the terms and the need for handling the temporal-related variations of characteristics require that classifier models take into account the creation time of the documents. However, the majority of current classifiers are based on training sets that contain all available documents indiscriminately, regardless of their creation time. A major difference of our approach is that we account for the collection's evolution by defining a Temporal Adjustment Factor, which weights the training documents as a function of the temporal distance between them and the document to be classified, d'. Our strategy is based on the fact that the closer in time two documents were created, the more similar are the observed characteristics of the collection [Mourão et al. 2008]. Further, the proposed factor aims to minimize the negative impact on the model associated with documents created long before or after d', while preserving relevant information regarding the classes' distribution in the collection.

In order to properly define the temporal adjustment factor, we must consider two parameters: the reference point and the temporal distance. The reference point is the target moment to which we classify an unseen document d', which is its creation moment. Our function must be aware of such point due to the varying nature of collection's characteristics over time. Temporal distance measures the interval between the creation moment of a training document and the reference point, in time units.

It is important to point out that the definition of the temporal adjustment factor is not unique, but it must capture the evolution of the collection's characteristics, weighting the documents towards minimizing the impact caused by the temporal evolution. There are several weighting strategies that may be employed and they must fulfill for two basic requirements: ability to quantify the temporal stability and use of a relevant analysis period. Next we discuss each of these requirements in detail:

Temporal Stability Measure. One key parameter of the adjustment factor is a temporal stability measure. Such measure must quantify the terms' and classes' robustness to the temporal evolution. The definition of such measure is not unique and is domain specific.

In this work, the temporal stability measure quantifies the resilience to temporal evolution, and, in practice, is the Dominance [Zaïane and Antonie 2002] of a term w.r.t. some class. Formally, let $\mathbb{C}$ be the set of classes observed in the training set, t be a term from a test document and $DF(t, c)$ be the number of training documents of class $c \in \mathbb{C}$ that contains t. The Dominance is defined as:

$$\textsc{Dominance}(t, c) = \frac{DF(t, c)}{\sum_{l=1}^{k} DF(t, c_l)}.$$

Such metric is a good measure of temporal stability since the stronger are the relationships between t and a given c over time, the smaller is the temporal variation of these relationships.

Analysis Period. The analysis period defines the set of documents that are temporally consistent with a document to be classified.

Considering documents from all the available moments to define the temporal adjustment factor is a computationally intensive task and may include documents that will introduce inconsistencies as a consequence of the temporal evolution. On other hand, considering just documents from the reference moment may result in sparse datasets. For example, considering Dominance as a measure, while longer analyses periods may include terms with small dominance values and, consequently, less representativeness for sake of ADC, considering just the documents from the reference point may maximize the terms's dominance, but this will lead to very few documents being selected and therefore will have consequences in the generalization of the model that will be built.

In order to define better analyses periods, we adopt the Stability Period concept [Rocha et al. 2008], which is defined as the longest continuous time period that includes the reference moment and all contiguous moments where their term's dominances are above a threshold α for some class c. We adopted a continuous interval due to the smooth nature of the variations observed in the reference collections [Mourão et al. 2008].

Now we describe the implementation of our proposed strategy. In order to define the temporal adjustment factor, we determined the average popularity of each moment in the terms' stability periods, through cross-validation procedure, varying α from 50% to 90%. As depicted in Figures 1 e 2, the resulting distributions were very similar, differing just in terms of the magnitude of the absolute values. Thus, we fixed $\alpha = 50\%$, ensuring exclusivity between a term and some class, regardless the number of observed classes.

Let m_r be a reference moment, and $\mathbb{V}_r \subset \mathbb{V}$ a subset of terms that occur in test documents created at m_r, where $\mathbb{V}$ denotes the collection's vocabulary. We calculate the continuous stability period $S_{t,r} = \{m_i, \cdots, m_j\}$ for each term $t \in \mathbb{V}_r$, whereas r denotes the reference moment. Notice that a term may present different stability periods for different reference moments. Hence, each moment $m_n \in S_{t,r} (i \leq n \leq j)$ is then mapped to temporal distances $\delta_n \leftarrow m_n - m_r$. Finally, we determine the occurrence frequency f_{δ_n} for each δ_n, considering all terms in $\mathbb{V}_r$. For example, let 1981 be the reference moment and the stability period of term t_1 (present in some training document created at 1981) be $S_{t_1,1981} = \{1980, 1981, 1982\}$, the resulting mapping to temporal distances is $\{-1, 0, 1\}$. Such procedure is repeated for all remaining terms that occur in training documents created at 1981. We then determine the absolute frequency of occurrence for each temporal distance, f_{δ_i} . Finally, each absolute frequency f_{δ_i} is divided by the number of terms occurring in test documents created at the reference moment m_r, leading to the relative frequency of occurrence:

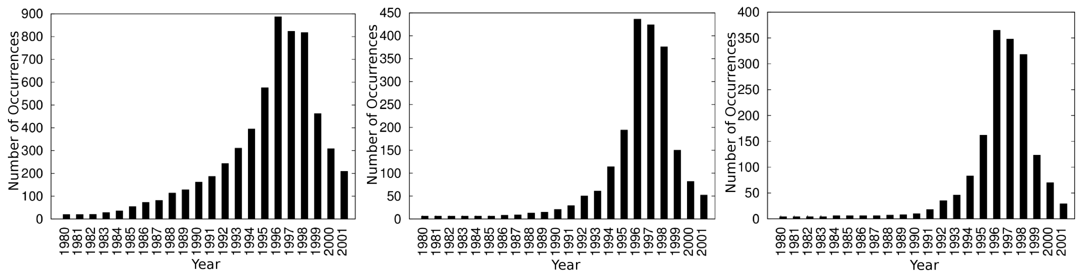
$$F(m_r, f_{\delta_i}) = \frac{f_{\delta_i}}{\|\mathbb{V}_r\|}.$$

Such procedure is applied to all possible reference moments. The adjustment factor associated with the temporal distance δ_i is the average relative frequency $F_{\text{AVG}}(\delta_i)$ for δ_i, considering all possible reference moments, as depicted in Figure 3, and formalized next:

$$F_{\text{AVG}}(\delta_i) = \frac{\sum\limits_{m' \in \mathbb{M}} F(m', f_{\delta_i})}{\|\mathbb{M}\|},$$

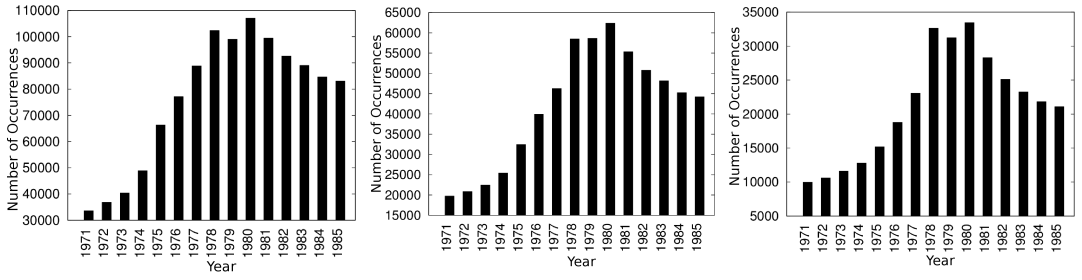
where $\mathbb{M}$ is the set of all possible reference moments for the temporal distance δ_i.

The last step for determining the temporal adjustment factor is its normalization, to avoid distortions associated with the interval lengths and number of documents. Training documents created at temporal distances δ_i with highest $F_{\text{AVG}}(\delta_i)$ characterize more precisely the characteristics of the collection at moment m_r. Thus, we set the highest average frequency to 1 and scaled the remaining values proportionally, according to the F_{AVG} distribution, so that the temporal adjustment factor is in the $[0, 1]$ interval.



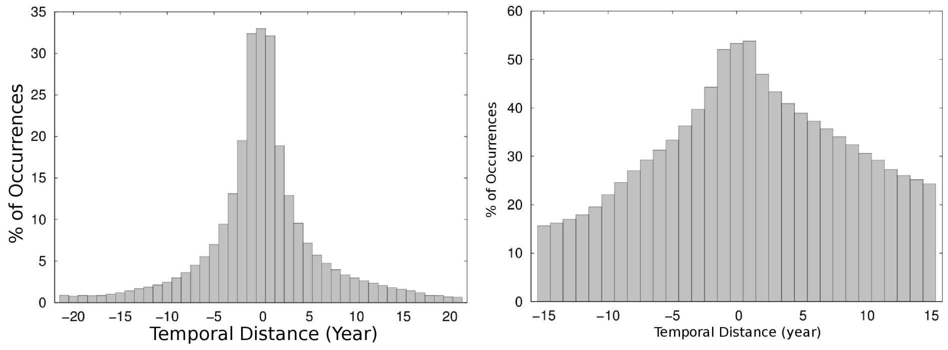
(a) Dominance 50 (b) Dominance 70 (c) Dominance 80

Fig. 1. Popularity of each moment in the stability periods (reference moment: 1996) - ACM-DL



(a) Dominance 50 (b) Dominance 70 (c) Dominance 80

Fig. 2. Popularity of each moment in the stability periods (reference moment: 1981) - MedLine



(a) ACM-DL (b) MedLine

Fig. 3. Relative frequencies of the temporal distances in the stability periods.

4. BUILDING TEMPORAL-AWARE CLASSIFIERS

In this section, we propose two strategies for building temporally robust classifiers based on the vector space model. As discussed before, the characteristics of textual collections tend to vary over time and documents created at different moments provide a variable amount of information for classifying an unseen document. These findings support the use of the temporal adjustment factor, introduced in Section 3, for enhancing existing classifier generation strategies.

One very first observation is that just lazy classifiers[1] may be enhanced by the use of the temporal adjustment factor, since it is a function of the temporal distance between the document to be classified and the documents in the training training set, information that is available just at the classification time. We envise two strategies for building temporal-aware classifiers, as we discuss next.

The first strategy uses the proposed temporal adjustment factor to weight the training documents used to build the classification model, in order to take into account their representativeness at distinct moments. The second strategy is to weight the scores produced by several classifiers which are more robust to temporal evolution, as follows. We partition the set of training documents into subsets $\mathbb{D}_m$ that comprise the training documents created at moment m, assuming that the temporal effects within a time unit are negligible. We then build a classifier for each training subset $\mathbb{D}_m$ and classify an unknown document d', generating class scores for each classifier. We then combine the resulting scores for each class by weighting them according to the temporal adjustment factor.

In terms of implementation, the first approach is straightforward, as will be further detailed in Sections 4.1 and 4.2. However, the second approach requires two transformations in the class domain. Let $\mathbb{C}$ be the set of observed classes in the target collection, and $\mathbb{M}$ the observed moments (in some temporal unit, as year or day). The first transformation, during the training phase, consists of grouping training documents into subsets $\langle c, m\rangle$, which contain documents assigned to the same class $c \in \mathbb{C}$ and created at the same moment $m \in \mathbb{M}$. We express such transformation as $c \rightarrow \langle c, m\rangle$. Next, a traditional classifier is used to classify the test document d' considering that transformed class domain, in order to determine the scores regarding each derived class. To define the target class c' to be assigned to d', an inverse transformation is conducted, considering the temporal distances between the creation moment of d' (m_r) and the moment m associated with the derived classes $\langle c, m\rangle$. Such transformation, referred to as $\langle c, m\rangle \rightarrow c$, is performed by aggregating each derived class' score, weighting them by the temporal adjustment factor λ_{m,m_r}, as follows:

$$c'_{d'} \leftarrow \arg\max_{c} \sum_{m \in \mathbb{M}} \textsc{Score}(\langle c, m\rangle, d') \cdot \lambda_{m,m_r}.$$

There are two key points worth noting. First, as we assume that the temporal effects within a time unit are negligible, we expect a more temporally robust classification when considering only training documents restricted to each single time unit. Second, to avoid undersampling and the potential loss of valuable information regarding class structure (e.g., the distribution of stable terms among the classes), in the second transformation we aggregate the scores associated with each time unit. In this step, we must handle the temporal effects and, to take into account the observed variability in the characteristics of the collection w.r.t. m_r, we perform a weighted sum of scores, with weights given by the temporal adjustment factor.

Having proposed the general steps regarding both strategies, in the next sections we describe how each algorithm can be extended towards an ADC temporally robust.

4.1 Temporal Rocchio

Rocchio is an eager, centroid-based classifier that computes a per-class representative vector to determine class boundaries. The decision rule assigns to an unseen document d' the class whose centroid is the nearest to d' in the vector space.

In order to consider the temporal aspects, we employ Rocchio as a lazy classifier (originally it is

[1]Lazy classifiers store all of the training documents and delays the inductive process until a test document needs to be classified. It differs from eager classifiers, such as Rocchio, which build a general model before receiving new test documents to be classified.

an eager one), since we need the test document's creation moment for applying the factor. Next, we detail both proposed strategies in order to deal with those temporal aspects.

4.1.1 *Temporal Adjustment Factor in Documents.* We employ the factor while calculating the centroid, since the documents considered were created at several different moments. Formally, let $\mathbb{D}$ be the set of training documents, $\mathbb{D}_c \subset \mathbb{D}$ be the set of training documents assigned to class $c \in \mathbb{C}$, $\mathbb{D}_m \subset \mathbb{D}$ be the set of training documents created at moment $m \in \mathbb{M}$ and $\vec{d_m}$ the vectorial representation of a document d created at moment $m \in \mathbb{M}$. The centroid $\vec{\mu}_c$ from class c is the weighted vector average of training documents $\vec{d_m} \in \mathbb{D}_c$, where $\vec{d_m}$ is weighted by the temporal adjustment factor λ_{m,m_r} (which is a function of the temporal distance between $\vec{d_m}$ and the reference moment m_r). In summary, the new centroid definition is given by:

$$\vec{\mu}_c \leftarrow \frac{1}{\|\mathbb{D}_c\|} \sum_{m \in \mathbb{M}} \left(\sum_{\vec{d_m} \in \mathbb{D}_c \cap \mathbb{D}_m} \vec{d_m} \cdot \lambda_{m,m_r} \right)$$

This strategy redefines class boundaries by properly adjusting centroids' coordinates so that they account for temporal effects, according to the representativeness of each $\vec{d_m} \in \mathbb{D}_m$ w.r.t. the reference moment m_r. The training and test stages are listed in Algorithm 1.

Algorithm 1 Temporal Rocchio – Temporal Adjustment in Documents

```
1: function Train($\mathbb{C}$, $\mathbb{M}$, $\mathbb{D}$, $d'$)
2:     $m_r \leftarrow d'_{creationMoment}$
3:     for each $c \in \mathbb{C}$ do
4:         $\mathbb{D}_c \leftarrow \{\vec{d} \in \mathbb{D} : d_{class} = c\}$
5:         $\vec{\mu}_c \leftarrow \frac{1}{\|\mathbb{D}_c\|} \sum_{m \in \mathbb{M}} \left( \sum_{\vec{d_m} \in \mathbb{D}_c \cap \mathbb{D}_m} \vec{d_m} \cdot \lambda_{m,m_r} \right)$
6:     end for
7:     return $\{\vec{\mu}_c : c \in \mathbb{C}\}$
8: end function
9: function Classify($\{\mu_c : c \in \mathbb{C}\}$, $d'$)
10:    return $\arg\max \cos(\vec{\mu}_c, \vec{d'})$
11: end function
```

4.1.2 *Temporal Adjustment Factor in Scores.* As mentioned before, this strategy weights the scores generated by several classifiers, one per time unit, being necessary to apply the two class domain transformations. The first transformation determines the centroids $\vec{\mu}_{c,m}$ for the subsets of documents $\langle c, m \rangle$. In order to classify a document d', it is necessary to determine the most similar centroid to d'. The similarity is a function of the scores produced by the various classifiers, that is, the score for a class c is the weighted sum of the scores for $\langle c, m \rangle, \forall m \in \mathbb{M}$:

$$\vec{\mu'}_c \leftarrow \sum_{m \in \mathbb{M}} \textsc{Score}(\vec{\mu}_{c,m}) \cdot \lambda_{m,m_r}.$$

where λ_{m,m_r} is the temporal adjustment factor. The class with highest final score is assigned to d'. The pseudo-code of this strategy is presented in Algorithm 2.

Algorithm 2 Temporal Rocchio – Temporal Adjustment in Scores

```
1:  function TRAIN($\mathbb{C}$, $\mathbb{M}$, $\mathbb{D}$)
2:      for each $\langle c, m\rangle \in \mathbb{C} \times \mathbb{M}$ do
3:          $D_{c,m} \leftarrow \{d : \langle d, \langle c, m\rangle\rangle \in \mathbb{D}\}$
4:          $\vec{\mu}_{c,m} \leftarrow \frac{1}{\|D_{c,m}\|} \cdot \sum_{\vec{d} \in D_{c,m}} \vec{d}$
5:      end for
6:      return $\{\vec{\mu}_{c,m} : \langle c, m\rangle \in \mathbb{C} \times \mathbb{M}\}$
7:  end function
8:  function CLASSIFY($\{\mu_{c,m} : \langle c, m\rangle \in \mathbb{C} \times \mathbb{M}\}, d'$)
9:      $m_r \leftarrow d'_{creationMoment}$
10:     for each $c \in \mathbb{C}$ do
11:         $\vec{\mu}'_c \leftarrow \sum_{m \in \mathbb{M}} \text{SCORE}(\vec{\mu}_{c,m}) \cdot \lambda_{m,m_r}$
12:     end for
13:     return $\arg\max \cos(\vec{\mu}_c, \vec{d'})$
14: end function
```

4.2 Temporal KNN

The K Nearest Neighbor (KNN) is a lazy classifier whose decision rule consists of assigning to the test document d' the most popular class among its k nearest neighbors. The implementation that we use as a starting point determines the majority class c through the weighted sum of cosine similarities between each of the k nearest documents $\vec{d} \in \mathbb{D}_c$ and $\vec{d'}$. Next we discuss how we extended KNN using the temporal adjustment factor.

4.2.1 *Temporal Adjustment Factor in Documents.* Our first strategy to extend KNN is similar to the one employed for Rocchio, that is, we weight the cosine similarity between each training document and the test document d' according to its temporal distance to d' ($\vec{d} \leftarrow \vec{d} \cdot \lambda_{m,m_r}$). More specifically, we reduce the contribution of a document proportionally to the disparity between the observed characteristics of the collection at their creation moments and those observed at reference moment m_r. We adapt KNN by embedding the temporal adjustment factor in the similarity calculation, as follows. Let λ_{m,m_r} be the adjustment factor for the moment m and the reference m_r. The similarity between the training document d and the test document d' is given by:

$$sim(d, d') \leftarrow \cos(\vec{d}, \vec{d'}) \cdot \lambda_{m,m_r}.$$

The strategy is listed in Algorithm 3.

4.2.2 *Temporal Adjustment Factor in Scores.* Again, we adjust KNN scores similarly as we did in Rocchio. We start by transforming the training datasets, so that we build a classifier for each subset of documents $\mathbb{D}_m$. We then calculated the weighted sum of the scores produced by the KNN classifiers using the corresponding temporal adjustment factor, as detailed in Algorithm 4.

5. EXPERIMENTAL RESULTS

In this section, we report our experimental results for the evaluation of the proposed temporal classifiers for both strategies (temporal adjustment factor in documents and in scores), contrasting them to the traditional approaches. The achieved results demonstrate the effectiveness of the temporal-aware versions, which outperformed significantly the original versions of the algorithms.

Algorithm 3 Temporal KNN – Temporal Adjustment in Documents

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

Algorithm 4 Temporal KNN – Temporal Adjustment in Scores

```
1: function Preprocess($\mathbb{D}$)
2:     for each $d \in \mathbb{D}$ do
3:         $c \leftarrow d_{class}$
4:         $m \leftarrow d_{creationMoment}$
5:         $d_{class} \leftarrow \langle c, m \rangle$
6:     end for
7: end function
8: function Classify($\mathbb{D}, d', k$)
9:     $m_r \leftarrow d'_{creationMoment}$
10:    $scores \leftarrow$ KNN $(d', k)$
11:    for each $s \in scores$ do
12:        $c \leftarrow s_{class}$
13:        $m \leftarrow s_{moment}$
14:        Score$[c] += s_{score} \cdot \lambda_{m,m_r}$
15:    end for
16:    return $\arg\max_c$ Score$[c]$
17: end function
```

5.1 Reference Collections

The evaluations presented here are experiments on two real and large text collections composed by documents from distinct knowledge areas. The first collection is a subset of the ACM Digital Library, comprising 24.897 computer science articles that span from 1980 to 2001. We consider just the first level of the ACM taxonomy, which consists of 11 distinct classes that were part of the taxonomy during the whole period. The second collection is derived from the MedLine collection, and consists of 861.454 documents from 7 distinct classes, which are associated with medicine papers published between 1970 and 1985. In both collections, each document is assigned to just one class.

5.2 Experiments

Since the document collections contain scientific articles, we adopted an annual time unit, since the events' editions tipically occur yearly. We should remark that, in other scenarios, such as news collections, the temporal unit may be different (e.g., day or week).

In order to evaluate KNN and its temporal-aware versions, it is necessary to determine the parameter k, since such parameter is key to the classification effectiveness. We used a simple heuristic, which varies k for each classifier ($k = \{30, 50, 100, 150, 200\}$) and selects the k values that provided the best classification performance for each strategy. For the original classifier, $k = 30$ provided the best results, whereas for both temporal classifiers $k = 50$ achieved the best results. Intuitively, we expect to employ a smaller k for the original versions of the classifier because it reduces the likelihood of considering documents that are temporally inconsistent with the test document. Similarly, larger k values for the temporal-aware versions maximize their performance, since we tend to consider more consistent documents. In both cases, however, there is a limit for k, since larger values of k introduce a significant bias, which may favor the most frequent classes, being a problem for imbalanced collections.

We employed the precision and recall metrics to evaluate the effectiveness of the proposed classifiers. Precision is the fraction of correctly classified documents assigned to a class. Recall is the fraction of documents from a class that were correctly classified. We usually evaluate classification quality through the F_1 measure, which is the harmonic mean of precision and recall. In practice, since our classification scenarios are multi-class problems, we employ MacroF_1 and Accuracy. MacroF_1 measures the classification effectiveness for each class, considered independently and equally important. Accuracy measures the global effectiveness of all classifier outcomes, and is defined as the fraction of correctly classified documents among all documents.

In order to ensure statistical significance, we adopted a 10-fold cross validation strategy [Breiman and Spector 1992], so that the final results are the average of all 10 achieved results. We present the averaged MacroF_1 and Accuracy in Tables I and II, for Rocchio and KNN, respectively. In each table, the lines entitled "bl.", "sco." and "doc." are the achieved results – MacroF_1 (column "macF_1") and Accuracy (column "acc.") – for the baseline (traditional classifier) and for the application of the temporal adjustment factor in scores and in documents, respectively. The line entitled "gain" is the percentage difference between the values achieved by the temporal classifiers and the traditional versions (baseline). 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. Therefore, ▲ denotes a significant positive variation, ● a non significant variation and ▼ a significant negative variation.

As observed in Table I, the Rocchio classifier that employs the temporal adjustment factor in scores achieved the highest gains: 18.87% and 11.46%, for MacroF_1 and Accuracy, respectively, considering MedLine collection. The same strategy also achieved the highest gains for KNN, as we can observe in Table II: 3.85% and 3.14%, for MacroF_1 and Accuracy, respectively, for ACM-DL collection. In all cases, the temporal classifiers outperformed the original ones for ACM-DL, which is a more dynamic collection, as pointed out by [Mourão et al. 2008].

Furthermore, the original version of the Rocchio classifier was the most affected by the temporal effects, considering the gains achieved by its temporal-aware versions. We explain such results by the fact that the representative vector from Rocchio averages information from all documents, which may be more affected by the temporal-related variability, while KNN takes a local decision, which is naturally less affected by the temporal-related variability in the document characteristics.

Notice that the application of the temporal adjustment factor in scores outperformed the application in documents, which can be explained by the variability of the evolutive patterns per term that are not quantified by the adjustment factor, which gives an overall value for a given time unit. We leave as future work to investigate a finer grain factor.

We then compared the KNN that uses the temporal adjustment factor applied in scores with the state of the art classifier SVM [Joachims 1999], in terms of effectiveness (classification quality) and efficiency (execution time). We used the SVM_Perf [Joachims 2006], a package that implements an efficient version of such classifier that can be trained at a linear cost. We adopted an one-against-all approach [Vapnik 1998] to use it in a multi-class problem, since SVM is a binary classifier. The results are reported in Table III, and the temporal KNN outperformed SVM w.r.t. Accuracy (6.84%), considering ACM-DL, and MacroF$_1$ (5.90%), for MedLine. Considering that SVM is a state of the art classifier and both collections are very imbalanced (which significantly difficulties the classification process), our results demonstrate the quality of the proposed solution.

Collection		ACM-DL		MedLine	
Metric		macF$_1$(%)	acc.(%)	macF$_1$(%)	acc.(%)
Rocchio Temporal Adjustment Factor in Scores	bl.	55.12	66.42	55.36	70.17
	sco.	59.07	71.54	65.81	78.21
	gain	+7.17 ▲	+7.70 ▲	+18.87 ▲	+11.46 ▲
Rocchio Temporal Adjustment Factor in Documents	bl.	55.12	66.42	55.36	70.17
	doc.	57.24	68.64	56.21	70.83
	gain	+3.85 ▲	+3.34 ▲	+1.53 ●	+0.94 ●

Table I. Temporal Rocchio evaluation.

Collection		ACM-DL		MedLine	
Metric		macF$_1$(%)	acc.(%)	macF$_1$(%)	acc.(%)
KNN Temporal Adjustment Factor in Scores	bl.	61.80	72.29	73.33	84.63
	sco.	64.18	74.56	76.82	87.01
	gain	+3.85 ▲	+3.14 ▲	+4.54 ▲	+2.81 ▲
KNN Temporal Adjustment Factor in Documents	bl.	61.80	72.29	73.33	84.63
	doc.	63.40	73.98	73.94	84.07
	gain	+2.59 ▲	+2.34 ▲	0.83 ●	-0.66 ●

Table II. Temporal KNN evaluation.

Coleção	ACM-DL			MedLine		
Metric	macF$_1$(%)	acc.(%)	Runtime (s)	macF$_1$(%)	acc.(%)	Runtime (s)
SVM	60.07	73.03	4200	72.28	83.27	1300000
Temporal KNN	64.18	74.56	80	76.82	87.01	5440
gain	+6.84 ▲	+2.09 ▲	–	+5.90 ▲	+4.49 ▲	–

Table III. Temporal KNN (Temporal Adjustment Factor in Scors) *versus* SVM.

6. CONCLUSION AND FUTURE WORK

This work proposed a solution to handle the temporal effects in ADC, which applies a temporal adjustment factor that aims to capture the evolutive characteristics of the documents. It is a quite general solution and has wide applicability, considering the temporal evolution of documents and its negative impact in ADC. We discussed the issues to be considered in order to define such factor, and proposed two strategies to use it in ADC: temporal adjustment factor in documents and in scores. We extended both Rocchio and KNN (deriving what we call Temporal Rocchio and Temporal KNN) and evaluated them using two real text collections. The temporal classifiers outperformed significantly the traditional ones with a reasonable computational cost.

There is still room for further improvements. The proposed temporal adjustment factor generalizes information from all documents and classes, being not specific for each term. Thus, it does not consider each term's specific evolutive pattern. Since it is known that the terms have distinct evolutionary characteristics, addressing them individually is promising and should be explored by refining the temporal adjustment factor. Such refinement can be achieved, for example, through Natural Computing techniques. Besides the dominance, other variables may have an important role in the definition of this factor. In this case, Genetic Programming methods may be used to find optimized functions that determine the adjustment factor, specifically for the target collection.

Moreover, given the results obtained when comparing SVM to the Temporal KNN, as a future work we may incorporate temporal information to the SVM classifier, by defining kernel functions that are temporally robust. Finally, other application scenarios, such as classification of news articles (which presents a more dynamic behavior) and adaptive document classification, demands investigation, allowing us to explore the potential of the temporal adjustment factor.

REFERENCES

Borko, H. and Bernick, M. Automatic Document Classification. *Journal of the ACM* 10 (2): 151–162, 1963.

Breiman, L. and Spector, P. Submodel Selection and Evaluation in Regression – the X-Random Case. *International Statistical Review* 60 (3): 291–319, 1992.

Caldwell, N. H. M., Clarkson, P. J., Rodgers, P. A., and Huxor, A. P. Web-Based Knowledge Management for Distributed Design. *IEEE Intelligent Systems* 15 (3): 40–47, 2000.

Cohen, W. W. and Singer, Y. Context-Sensitive Learning Methods for Text Categorization. *ACM Transactions on Information Systems* 17 (2): 141–173, 1999.

Dries, A. and Rückert, U. Adaptive Concept Drift Detection. *Statistical Analysis and Data Mining* 2 (5-6): 311–327, 2009.

Folino, G., Pizzuti, C., and Spezzano, G. 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, pp. 183–188, 2007.

Joachims, T. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (Eds.). MIT Press, Cambridge, USA, chapter 11, pp. 169–184, 1999.

Joachims, T. Training Linear SVMs in Linear Time. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Philadelphia, USA, pp. 217–226, 2006.

Kim, Y. S., Park, S. S., Deards, E., and Kang, B. H. Adaptive Web Document Classification with MCRDR. *Information Technology: Coding and Computing* vol. 1, pp. 476, 2004.

Klinkenberg, R. Learning Drifting Concepts: Example Selection vs. Example Weighting. *Intelligent Data Analysis* 8 (3): 281–300, 2004.

Klinkenberg, R. and Joachims, T. Detecting Concept Drift with Support Vector Machines. In *Proceedings of the International Conference on Machine Learning*. Stanford, USA, pp. 487–494, 2000.

Kolter, J. and Maloof, M. Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. Technical report, Department of Computer Science, Georgetown University, Washington, USA. June, 2003.

Lawrence, S. and Giles, C. L. Context and Page Analysis for Improved Web Search. *IEEE Internet Computing* 2 (4): 38–46, 1998.

Lazarescu, M. M., Venkatesh, S., and Bui, H. H. Using Multiple Windows to Track Concept Drift. *Intelligent Data Analysis* 8 (1): 29–59, 2004.

Liu, R.-L. and Lu, Y.-L. Incremental Context Mining for Adaptive Document Classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Canada, pp. 599–604, 2002.

Mourão, F., Rocha, L., Araújo, R., Couto, T., Gonçalves, M., and Meira, Jr., W. Understanding temporal aspects in document classification. In *Proceedings of the International Conference on Web Search and Web Data Mining*. Palo Alto, USA, pp. 159–170, 2008.

Rocha, L., Mourão, F., Pereira, A., Gonçalves, M. A., and Meira, Jr., W. Exploiting temporal contexts in text classification. In *Proceedings of the International Conference on Information and Knowledge Engineering*. Napa Valley, USA, pp. 243–252, 2008.

Scholz, M. and Klinkenberg, R. Boosting Classifiers for Drifting Concepts. *Intelligent Data Analysis* 11 (1): 3–28, 2007.

Tsymbal, A. The problem of concept drift: Definitions and Related Work. Technical report, Department of Computer Science, Trinity College, Dublin, Ireland. December, 2004.

Vapnik, V. N. *Statistical Learning Theory*. Wiley-Interscience, New York, USA, 1998.

Widmer, G. and Kubat, M. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning* 23 (1): 69–101, 1996.

Zaïane, O. R. and Antonie, M.-L. Classifying text documents by associating terms with text categories. In *Proceedings of the Australasian Database Conference*. Melbourne, Australia, pp. 215–222, 2002.