Information Systems 38 (2013) 388–409

Contents lists available at SciVerse ScienceDirect

Information Systems

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

Temporal contexts: Effective text classification in evolving document collections

Leonardo Rocha[a,*], Fernando Mourão[b], Hilton Mota[c], Thiago Salles[b], Marcos André Gonçalves[b], Wagner Meira Jr.[b]

[a] Federal University of São João del-Rei, Computer Science Department—São João del-Rei, Brazil
[b] Federal University of Minas Gerais, Computer Science Department—Belo Horizonte, Brazil
[c] Federal University of Minas Gerais, Electrical Engineering Department—Belo Horizonte, Brazil

ARTICLE INFO

Article history:
Received 21 October 2011
Received in revised form
29 October 2012
Accepted 8 November 2012
Recommended by: T.G.K. Calders
Available online 21 November 2012

Keywords:
Classification
Text mining
Temporal evolution

ABSTRACT

The management of a huge and growing amount of information available nowadays makes Automatic Document Classification (ADC), besides crucial, a very challenging task. Furthermore, the dynamics inherent to classification problems, mainly on the Web, make this task even more challenging. Despite this fact, the actual impact of such temporal evolution on ADC is still poorly understood in the literature. In this context, this work concerns to evaluate, characterize and exploit the temporal evolution to improve ADC techniques. As first contribution we highlight the proposal of a pragmatical methodology for evaluating the temporal evolution in ADC domains. Through this methodology, we can identify measurable factors associated to ADC models degradation over time. Going a step further, based on such analyzes, we propose effective and efficient strategies to make current techniques more robust to natural shifts over time. We present a strategy, named temporal context selection, for selecting portions of the training set that minimize those factors. Our second contribution consists of proposing a general algorithm, called *Chronos*, for determining such contexts. By instantiating Chronos, we are able to reduce uncertainty and improve the overall classification accuracy. Empirical evaluations of heuristic instantiations of the algorithm, named *WindowsChronos* and *FilterChronos*, on two real document collections demonstrate the usefulness of our proposal. Comparing them against state-of-the-art ADC algorithms shows that selecting temporal contexts allows improvements on the classification accuracy up to 10%. Finally, we highlight the applicability and the generality of our proposal in practice, pointing out this study as a promising research direction.

Contents

* Corresponding author. Tel.: +55 32 3373 3985.
E-mail address: lcrocha@ufsj.edu.br (L. Rocha).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.11.001

1. Introduction

The widespread use of the Internet has increased the amount of information being stored and accessed through the Web in a very fast pace. This information is frequently organized as textual documents [1] and is the main target of search engines and other retrieval tools, which perform tasks such as searching and filtering. A common strategy to deal with this information is to associate it with semantically meaningful categories, a technique known as Automatic Document Classification (ADC) [2]. This automatic document class assignment can support and enhance several tasks, such as automated topic tagging [3], digital library creation [4], and improvement of Web searching precision [5].

ADC usually employs a supervised learning strategy, in which a classification model is first built using pre-classified documents, i.e., a training set, and this model is then used to classify unseen documents. Building text classification models consists of finding and weighting a set of features (e.g., terms) that help to identify classes of documents. The concept is illustrated in Fig. 1.

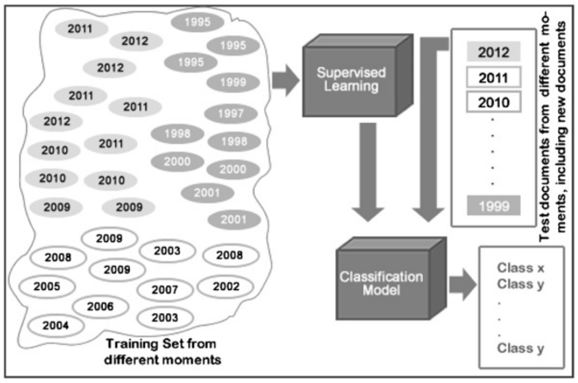
2011
2012
1995
Training Set from different moments
Supervised Learning
Classification Model
2012
2011
2010
1999
Test documents from different moments, including new documents
Class x
Class y
Class y

Fig. 1. Traditional supervised learning strategy. The whole training set, composed of documents from distinct moments, is given to a supervised learning technique. Then, it generates a classification model that infers classes for each test document, disregarding its creation moment.

In [6], it is advocated that time is an important dimension of any information space and may be very useful in information retrieval. Despite the potential impact of the temporal evolution on the quality of classification models, most of the current techniques for ADC do not consider this evolution while building and using the models. Furthermore, there are some relevant questions related to temporal evolution that are still not clear, three of which we address in this paper, namely:

(1) How does the temporal evolution of the information space affect the performance of the classifiers?
(2) What are the temporal-related characteristics that affect the classification's effectiveness?
(3) How to exploit such characteristics to improve the classification's effectiveness?

The first two questions are addressed in the first part of the paper. We distinguish three temporal effects that may affect the performance of automatic classifiers. The first one, called *class distribution*, is related to the impact of temporal evolution on the distribution of class frequency. It is a consequence of the dynamic evolution of knowledge and the ways to express it, which may result in classes appearing, disappearing, splitting or merging. The second effect, called *term distribution*, refers to how the relationship between terms and classes changes over time. These changes may occur as a consequence of terms appearing, disappearing, and presenting variable discriminative power within classes. The third effect, *class similarity*, concerns how the similarities among classes vary over time, as a function of the terms that occur in their documents. For instance, two classes may be very similar at a given moment, and less similar later in the future.

In order to understand and characterize these three factors more easily, we propose a novel and pragmatical methodology for assessing the impact of the temporal evolution in ADC applicable to distinct domains. Our methodology allows us to analyze each factor separately, pointing out the causes that result in the classifier's effectiveness degradation. Evaluations of such methodology on two real collections demonstrate its usefulness. To the best of our knowledge, we are the first to evaluate in depth, quantify, and identify measurable factors related to the temporal evolution in ADC.

After understanding the temporal effects in ADC, the next step is to answer the third question, that is, how to build effective classification models regardless of temporal effects. One strategy is to create classification algorithms that directly deal with the aforementioned temporal effects. A different strategy, which is the focus of this paper, is more data engineering oriented: it selects a portion of the training set that minimizes those effects, considering a specific creation moment m_i, and then uses only this portion to train a classifier for unseen documents belonging to m_i. The adoption of such strategy is justified by two facts. First, this 'data engineering' could be applied to almost all existing ADC techniques, without any sort of change, since it can be seen as a pre-processing step. Second, it could be easily adapted to distinct domains needs and behaviors in terms of dynamics. This strategy is illustrated in Fig. 2.

The problem of selecting temporal contexts is formally defined through the identification of three requirements that should be fulfilled by a good temporal context: reference constrained, characteristic stability and uncertainty reduction. Based on these requirements, we propose a general algorithm for determining such contexts, named *Chronos*, which is able to derive effective and efficient strategies to make current techniques more robust to natural shifts over time. Since Chronos defines an exponential search space, the search for optimal solutions is not feasible in practical scenarios, motivating us to exploit heuristic solutions able to balance effectiveness and performance. For such, we propose *WindowChronos* and *FilterChronos*, two heuristic instantiations that construct continuous time-windows that may grow in both directions, past and future, starting from a given creation moment m_i. These time-windows are defined by

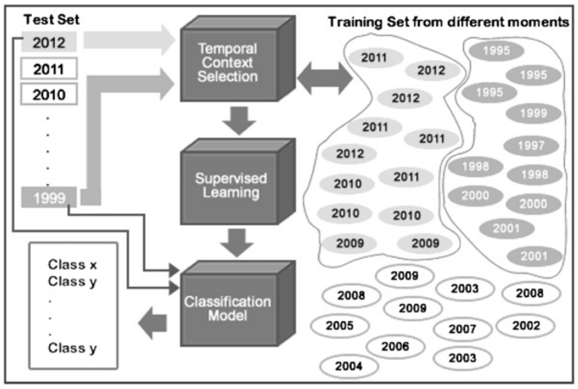

Fig. 2. Supervised learning strategy with temporal contexts. Each distinct moment of a test set is taken into account by the temporal context selection in order to retrieve an appropriate piece of the training set. Only this piece is given to a supervised learning technique for defining a classification model. Test documents belonging to different moments are classified by using the temporally corresponding model.

evaluating the terms' stability regarding our three requirements, as the time distances increase from m_i. The main difference of these two heuristics concerns the information used for composing the contexts. While *WindowChronos* takes into account only features from the test set, an approach referenced as asymmetrical, *FilterChronos* considers both training and test features, defining a symmetrical approach.

In order to verify whether the temporal contexts selected by these heuristics improve the classification performance, we evaluate them in conjunction with four state-of-the-art classification algorithms *k nearest neighbors* (kNN), Naïve Bayes, Rocchio and SVM—using the two collections mentioned above. The results show that *WindowChronos* is able to enhance the performance of algorithms that assume an asymmetric premise, but do not work properly with symmetric ones. The use of *FilterChronos*, on the other hand, resulted in significant improvements in all evaluated scenarios. For instance, by using temporal contexts generated with this heuristic, the algorithms were able to outperform their original versions by up to 8%, 9%, 5% and 3%, respectively. Analyzing these results in the light of our characterizations and the properties of the collections, we also present some conclusions about the potential and limitations of using temporal contexts by each one of the evaluated algorithms. Finally, we highlight the applicability and the generality of the discussed issues in practice, pointing out this study as a promising research direction on distinct real domains.

This paper is organized as follows. Section 2 covers related work. Section 3 characterizes the temporal effects in textual collections. Section 4 formalizes the temporal context selection problem and presents *Chronos*. Section 5 introduces the two heuristics that implement the general algorithm while Section 6 describes our experimental evaluation. Section 7 concludes the paper.

2. Related work

While ADC is a widely studied topic, the analysis of the impact caused by temporal aspects has only started in the last decade. Actually, most of the current efforts aim to alleviate such impacts on classifiers without a proper understanding about its causes. In contrast, our study first aims to characterize and understand this problem, providing insights to the development of temporally robust classifiers. In this sense, in [7], we presented a qualitative analysis regarding the existence of three main temporal effects which are, in fact, manifestations of the drifting patterns. A brief explanation of these results is presented in the next section.

Despite the limited understanding about these temporal aspects, several proposals to *minimize* their impact in ADC exist. Such efforts can be categorized in two broad areas, namely adaptive document classification and concept drift. Adaptive document classification [8] embodies a set of techniques to deal with changes in the underlying data distribution in order to improve the effectiveness of classifiers through incremental and efficient adaptation of the classification models. This line of investigation brings a number of challenges to text mining. The first challenge 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 close to a certain keyword [9,10] and terms that indicate the scope and semantics of the document [11]. The second challenge is creating the models incrementally [12]. The third challenge regards the computational efficiency of the document classifiers. These techniques often adopt a notion of context very different from ours, i.e., they exploit *semantic contexts*, which arise, for instance, from the co-occurrence of terms [13]. A semantic context is not associated with the temporal dimension of the data, and may even present occurrences at different moments. For instance, a semantic context defined by the co-occurrence of the terms 'building' and 'world trade center' may be observed both after 2001, associated with the terrorism topic, as before, more correlated to the financial topic. Despite this clear distinction, for adaptive document classification this context is seen uniquely associated with the most frequent topic. However, we believe that semantic contexts from distinct moments may be associated with different topics. Thus, in this paper we are concerned with performing document classification in the creation moment of test documents, which we call *temporal context*. Note that the traditional definition of semantic contexts can be improved by our temporal contexts, since the former when, restricted to a unique temporal context, tend to be associate with a single topic.

On the other hand, concept (or topic) drift [14] groups techniques in which the classifier is completely retrained according to some instance selection or weighting method. It comprises most of the recent efforts in dealing with temporal aspects and data streams with changing distributions, from a wide domain of applications like textual classification [15], recommendation systems based in collaborative filtering [16] and others. A number of previous studies fall in the instance selection approach category. In [17] the authors employed a sliding window whose size is automatically adjusted in order to minimize the estimated generalization error. The method proposed in [18] builds the classification model using training instances which are close to the test in terms of both time and space. In [19] different approaches are used such as adaptive time windows on the training data and selection or weighting of representative training examples for model construction. In [20] the authors described a set of algorithms that react to concept drift in a flexible way and can take advantage of situations of recurring contexts. The main idea of these algorithms is to keep only a window of currently trusted examples and store concept descriptions in order to reuse them if a previous context reappears. In general, concept drift assumes that the data are sorted chronologically and that test items are always newer than the training ones. Thus, the goal is always to adapt the training information in order to model a new scenario defined by the test, focusing only on the newest training information. This sort of scenarios is well studied and there are various proposals to address efficiently ADC in such cases. Our study focuses on deeply analyzing a scenario disregarded by most of these studies: when data are not ordered chronologically, with test items created at any existing moment in the training. Since it is a broader scenario, obviously, we also can use our proposal to deal with the concept drift, handling new items, although our focus is on old documents.

Regarding the instance weighting approach, a common strategy to deal with concept drift focuses on the combination of various classification models generated by different algorithms [21,22]. 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 is empirically shown to outperform learning algorithms that ignore concept drift. In [22], it is presented a method that builds an ensemble of classifiers using Genetic Programming (GP) to inductively generate decision trees, each trained on different parts of the distributed training set. Despite not being the focus of the present work, these techniques could be adapted to use the temporal contexts selected by our algorithms in the generation process of classification models. Finally, still regarding the instance weighting approach, a study that has shown promising results is presented in [23]. In that work the temporal information is incorporated to document classifiers based on the evolution of term-class relationship over time, which is captured by a metric of *dominance* [24]. A *temporal weighting function* (TWF) is defined for a given dataset by means of a series of statistical tests aimed at determining the TWF's expression and this TWF is then incorporated to ADC algorithms. It is noteworthy that we decided to propose an instance selection based strategy rather than an instance weighting one mainly due to three limitations related to the later. First, instance weighting tends to generate complex ADC models, since they must be robust to temporal shifts on the data. Second, such models usually require a distinct parameter tuning according to different dynamic levels observed in each domain. Finally, underlying assumptions may limit the applicability of such strategies in certain scenarios.

More recently, two approaches related to data stream mining have been receiving a lot of attention: Concept-evolution [25–27] and Topic Models [28–30]. The first group aims to fill the gap left by the traditional data stream classification techniques that are not able to identify novel classes until their manual identification is performed, mainly in scenarios with the presence of concept-drift in which the data distribution changes over time. The second group aims to discover and annotate large archives of documents with thematic information. In these approaches, prior annotations or document labeling are not required since the topics emerge from the analysis of the original texts, as well their changes over time. Topic modeling is an important tool to organize and summarize electronic archives at a scale that would be impossible by human annotation. Our work differs from these studies since we need a training set in which the classes must be manually assigned and we do not consider the appearance or disappearance of classes over time.

Finally, a group of techniques that is interesting to mention is related to the transfer learning paradigm [31]. In [32] a Temporal Inductive Transfer (TIX) method is proposed. TIX follows a transfer learning paradigm and does not assume that the data distributions remain the same as time goes by. It considers the data created at the same test's creation time as the target domain, whereas the remaining documents compose the source domain. TIX transfers knowledge from the source domain to the target domain in order to improve classification effectiveness. In [33], the authors also propose a classification strategy based on the transfer learning paradigm, however considering an online learning task [34]. In this case, transfer learning is used to combine a previously built model with an online learning model, in order to take advantage of data whose distribution may not be the same as the distribution currently observed. The proposed Online Transfer Learning (OTL) algorithm uses this combination in order to learn from data that potentially comes from distinct distributions (for example, due to concept drift) increasing both the initial performance of the online classifier and its asymptotic effectiveness. Efforts on TIX, however, were concerned to cope with concept drift, analyzing only scenarios in which test items are newer than training ones.

Differently from all previously mentioned work, in this paper we introduce the explicit concept of *temporal contexts*, defined as a subset of the collection that minimizes the impact of temporal effects in the performance of classifiers. These temporal contexts are used to sample the training examples for the classification process, discarding instances that are considered to be outside the context. The most distinct advantage of this approach is that it can be applied together with different classification algorithms, since it can be seen as a data engineering pre-processing technique.

3. Characterizing temporal effects

This section presents experiments that clarify how the temporal evolution manifests in document collections. The results demonstrate the existence of three temporal effects that contribute to this evolution: class distribution, terms distribution, and class similarity [7]. We also discuss how the sample size affects the precision of classifiers (the sampling effect), and the trade-off between this and the temporal effects.

The experiments were conducted using the Support Vector Machine (SVM), considered a state-of-the-art classifier [35], applied to two document collections. The first one is a set of 30,000 documents from the ACM Digital Library (ACM-DL) containing papers from 1980 to 2002. The classes of the ACM taxonomy are assigned to documents by their own authors. We used just the first level of the taxonomy, comprising all 11 categories, so that each document is assigned to a unique class. The second collection is MedLine, which consists of 4,112,069 documents from 1970 to 2006, classified into seven distinct classes. Note that this collection is significantly larger (by almost 2 orders of magnitude) and more diverse than OHSUMED,[1] which is a medical collection commonly used for evaluation of document classifiers [36]. Both datasets can be found at http://www.lbd.dcc.ufmg.br/lbd/collections/temporal-contexts-datasets.

A first aspect to be addressed when dealing with temporal information regards the temporal granularity. In fact, we can see time as a discretization of natural changes inherent to any knowledge area, although detectable changes may occur at different time scales (minutes, days, years, etc.). In the case of our experimental collections, which correspond to sets of scientific papers, we adopted yearly intervals for identifying such changes (e.g., scientific conferences are usually annual), but in other scenarios the granularity may be different.

In order to quantify the sampling effect, a document sub-collection was built for each collection, containing documents from just one year. This allowed the separation of sampling from temporal aspects. Using these sub-collections, several classification models were built, varying the size of the training set from 20% to 100% of the sub-collection. As expected, the results demonstrated that, as we increase the size of the training set, the classifier performance gets better.

To assess the temporal effects, each collection was divided into several sub-collections A_i, one per year, each containing the same number of documents (441 for ACM-DL and 34,755 for MedLine[2]). Then, each sub-collection A_i was further divided in three parts with the same size. We used two parts as our training set to generate the classification model. Then this model was evaluated over all the other sub-collections A_j, in which $j \neq i$. Besides, we also tested with the portion of A_i not used as the training set. We repeated this process for each of the three combinations using 3-fold cross-validation. Fig. 3 summarizes the experiments for both ACM and MedLine. The vertical axis shows the relative accuracy (y-axis) and the horizontal the various time distances (x-axis). This is measured as the difference, in years, between the training and test sub-collections. Notice that, in most cases, the best scenario is found when the training and test documents belong to the same year (time distance zero).

These results demonstrate a challenge in dealing with the sampling and temporal effects simultaneously. Increasing the size of the training set may introduce documents that are out of the defined temporal context, reducing the effectiveness of the classifier. On the other hand, by reducing the training set to only documents temporally close to the test document, there is a risk of not having enough information to build a model able to generalize. It can be seen that the best training set would be formed by the documents temporally consistent with the documents being classified and, simultaneously, as large as possible. This is one of the principles for the "temporal context selection" proposed in this work.

3.1. Quantification of temporal effects

In order to establish parameters to define the proper temporal contexts, initially we performed an investigation to

[1] http://ai-nlp.info.uniroma2.it/moschitti/corpora.htm.

[2] The years with the smallest numbers of documents are 1980 (441) for ACM-DL and 1970 (34,755) for Medline.

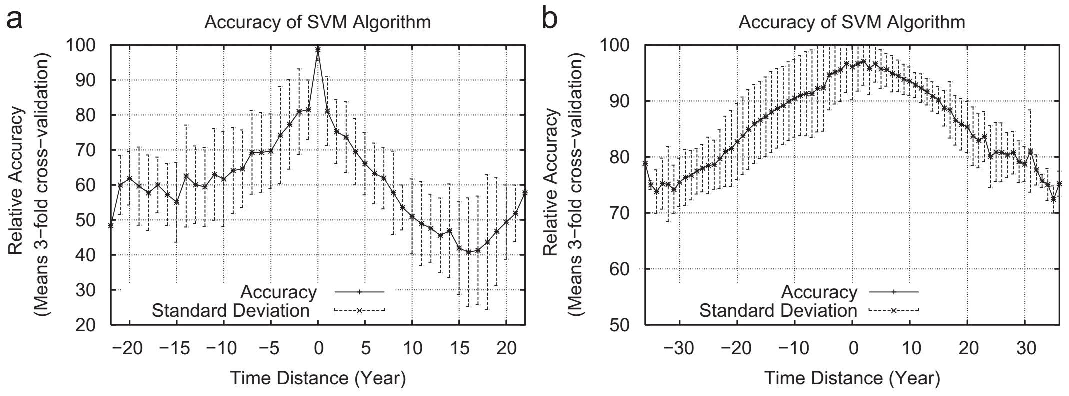

Fig. 3. Temporal locality variation. (a) ACM (b) MedLine.

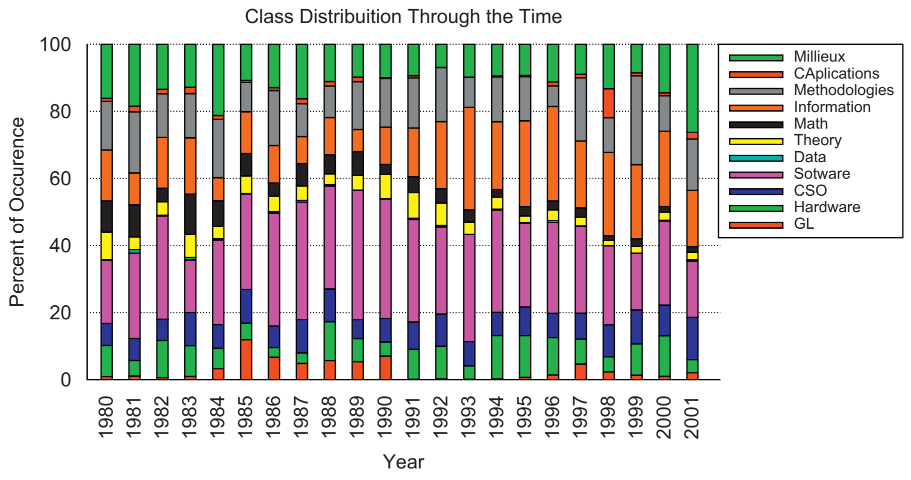

Fig. 4. Class occurrence variation through the time. ACM collection.

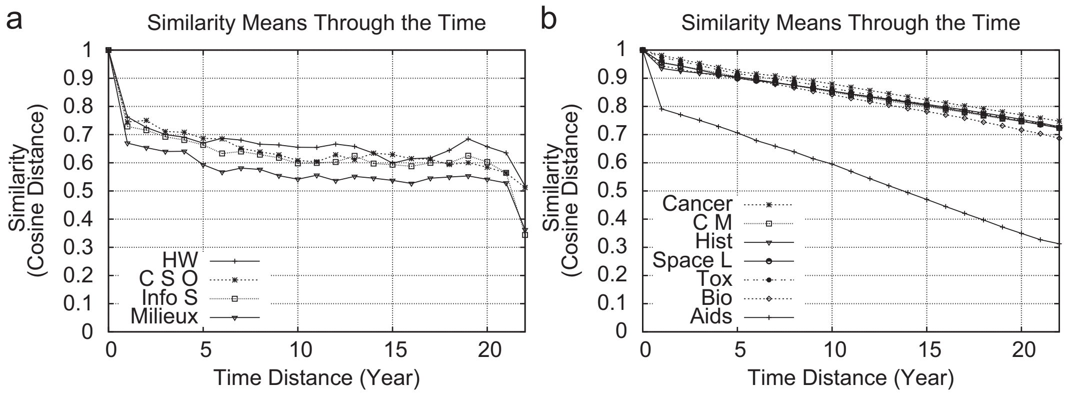

Fig. 5. Terms distribution means. (a) ACM. (b) MedLine.

understand and quantify the aforementioned temporal effects (class distribution, terms distribution, class similarity).

The procedure started by analyzing the variation of the class distributions over time. As an example, the class probability distribution of ACM is plotted in Fig. 4. It illustrates the variation of class representativeness over the years. As can be seen, the class frequency distribution varies significantly (e.g. the class *Math* becomes less frequent as time evolves). Similar results were observed for the MedLine collection. Additionally, in [7] we conducted an experiment that showed that the document distribution among classes influences the performance of the classifiers.

The second step was to characterize the term distribution effect. To this intent, we defined sets of vocabularies composed by the words with the highest values of info-gain [37] to represent the classes. These vocabularies were constructed for each class and each year of occurrence. Then, they were compared against each other for the same class in different years, using the normalized cosine similarity. The results can be seen in Fig. 5 as the evolution of the mean of cosine similarity versus the time distance. Distance zero means a comparison of a vocabulary to itself, which obviously corresponds to the maximum similarity. It can be observed that the larger the time distance, the less similar the vocabularies are. These experiments demonstrate that vocabularies change over time and this evolution must be accounted for when building the classification model. Models generated according to documents of a certain period of time may be less effective when testing documents from another period of time, even if they belong to the same class. The vocabulary may evolve in a way so that the identified premises are no longer true, that is, the discriminative terms may not be the same anymore.

Finally, to verify the effect of the evolution on the class similarity, we calculated the cosine similarity between the vocabularies for each pair of distinct classes for each year. Table 1 shows the results for the MedLine classes. Each entry in the table corresponds to the standard deviation from the mean of the similarities between the associated pairs of classes in all years. As can be observed, for some pairs of classes the similarity variation is very high, such as for *Complementary Medicine* (CM) and *History* (Hist). This means that these classes may have been very similar in some periods, but were loosely related in other periods. Consequently, the difficulty in separating them varies significantly over time. Similar results were obtained for ACM and are not presented here for the sake of brevity.

Based on this brief discussion, we found evidence of the class and term variation over time. Moreover, we could see that these effects may affect significantly the performance of the classifiers. In the next section, we present an analysis to show that the appropriate understanding of the time effects may lead to improvements in the classification process. The idea is not to propose a method for exploring temporal information but to show that there is room for the development of such methods as a way to increase the classifier performance.

3.2. Exploring temporal effects

The previous sections demonstrated that time evolution imposes a trade-off between sampling and time effects. In order to show how temporal information can be used to optimize this trade-off, we performed an experiment consisting of a time-sensitive selection of documents for training. In this approach the training set was composed only by documents closer in time to the document being tested. The proximity is defined by a time window that can grow symmetrically in both directions, past and future, related to the year of the test document. To account for the sampling effect, we varied the size of this window from 0 (that is, the year A_i the tested document belongs to) to N, the number of years before and after A_i. The best N was defined as the value that maximizes the performance of the classifier. We performed this search for the test documents of each year, evaluating all possible window sizes. The results are presented in Fig. 6. The main observation of this experiment is that the window sizes vary significantly for each year. This demonstrates that there is no unique value for the number of years to consider. This parameter must be dealt with dynamically, since it varies with the dynamics of the classes and the terms that characterize them.

Fig. 7 shows the highest accuracy values found for each year in both collections. The mean accuracies obtained in the experiments were 89.76% for ACM and 87.57% for MedLine. These are much better than the results obtained when the whole set was used for training, which were 73.03% and 73.67%, respectively.

This empirical analysis shows that the proper exploitation of temporal aspects may lead to significant improvements in the classification process. Nevertheless, despite this exhaustive search being appropriate to analytical evaluation, it is certainly impracticable in actual scenarios due to computational costs. From this point of view, a need emerges for computationally feasible approaches that properly explore the temporal aspects of document classification.

4. Temporal context selection

4.1. Problem definition

Before dealing with the problem of selecting temporal contexts, we formally define text classification with temporal information.

Let $D=\{d_1,d_2,\ldots,d_K\}$ be the documents that compose a training set of a collection; $C=\{c_1,c_2,\ldots,c_L\}$ be the set of categories that occur in a collection; $T=\{t_1,t_2,\ldots,t_U\}$ be the set of terms associated with the documents of a collection; $M=\{m_1,m_2,\ldots,m_P\}$ be the set of moments

Table 1
Similarity Std_Dev matrix—MedLine.

	Aids	Bio	Cancer	C M	Hist	Space	Tox
Aids	0	0.19	0.16	0.18	0.19	0.18	0.19
Bio	–	0	0.04	0.20	0.17	0.19	0.12
Cancer	–	–	0	0.04	0.03	0.04	0.05
C M	–	–	–	0	0.21	0.08	0.05
Hist	–	–	–	–	0	0.20	0.11
SpaceL	–	–	–	–	–	0	0.05
Tox	–	–	–	–	–	–	0

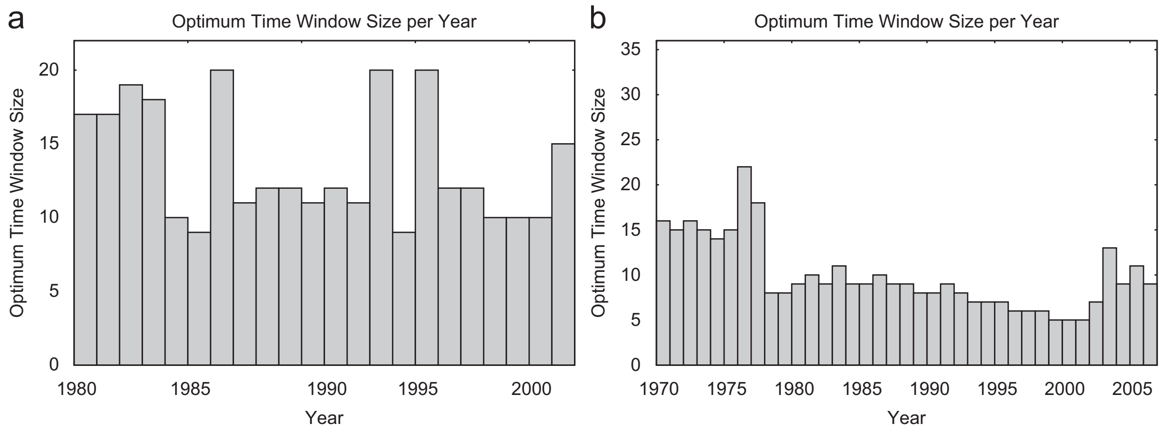

Fig. 6. Analysis of the optimum time window per year. (a) ACM (b) MedLine.

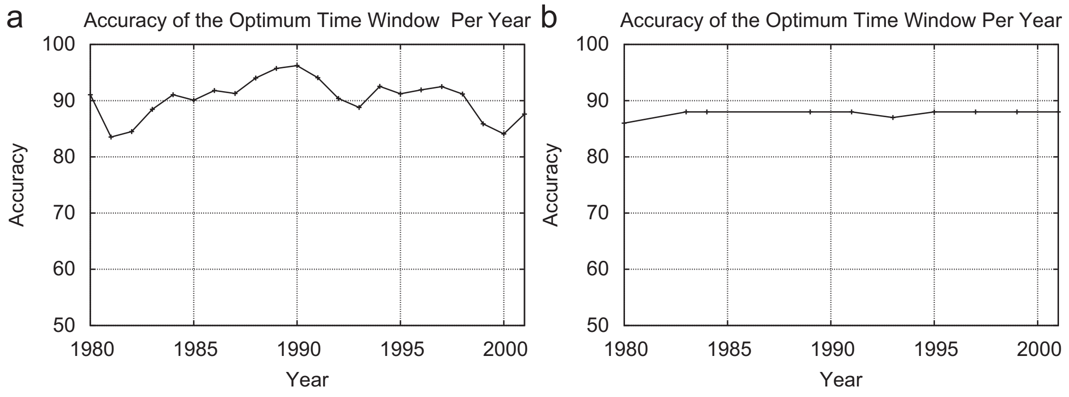

Fig. 7. Accuracy of the optimum time window per year. (a) ACM. (b) MedLine.

Table 2
Example.

Word: Pluto			
Class: astrophysics		**Class: mythology**	
Doc_Id	Year	Doc_Id	Year
d_1	1998	d_9	2005
d_2	1999	d_{10}	2006
d_3	2000	d_{11}	2006
d_4	2001	d_{12}	2008
d_5	2002	d_{13}	2009
d_6	2003	d_{14}	2009
d_7	2004	d_{15}	2011
d_8	2005	d_{16}	2012

on a temporal space when both test and training documents were created (timestamp). We define a training document d_k by the triple $\{Z,m_p,c_l\}$, where $Z \subseteq T$, $m_p \in M$ and $c_l \in C$. Now, given a set $S=\{s_1,s_2,\ldots,s_V\}$ comprising the test documents that must be classified, the task of ADC is to determine the class c_i associated with each test document s_i, where $s_i=\{Y,m_p\}$, $Y \subseteq T$, $m_p \in M$. ADC usually follows a supervised learning strategy, where we first select a subset of documents $X \subseteq D$ which is used to build a classification model. Then we use this model to classify new unseen documents (S). Often X is equal to D or a random sample of D and, consequently, it is usually composed by documents created at different time moments (M). It is important to clarify that this subset of documents X is defined as a context in this work.

As previously discussed, the sampling effect suggests that the size of a context X should be increased, while the temporal effects induce the reverse. Therefore, the *temporal context selection problem* could be formally defined as *determining the biggest subset X of documents belonging to a training set D that simultaneously minimizes the measurable effects related to the temporal evolution of D.* Notice that the number of possible contexts grows exponentially with the number of documents, thus imposing the need for strategies that narrow the search. Further, it is important to distinguish the context selection problem from feature selection [38], which aims to reduce the number of features used in the classification model. Although the reduction of size turns to be a (beneficial) side-effect, the main goal of temporal context selection is to balance the trade-off between sampling and temporal effects for the sake of the classification performance.

To illustrate the problem, consider the training set presented in Table 2. It consists of documents that contain the term *Pluto*. Besides being the god of hell in Roman mythology, *Pluto* was also considered to be a planet until the middle of 2006. As it can be observed, before 2005 the training set consists of documents from only one class, Astrophysics. After 2005, it contains only documents from class Mythology, while in 2005 there is one document from each class. If the whole training set is employed to train a classifier, as it is usually done, it would depend on other features to determine the correct category for a document that contains *Pluto*. However, the problem can be solved by considering the temporal information for delimiting contexts where *Pluto* has an unambiguous meaning, as we propose in this paper.

4.2. The Chronos algorithm

In this section we present a general algorithm for temporal context selection. This algorithm may be used as a template for further implementations aiming to minimize the temporal effects.

Based on the discussion presented in Section 3, we identify three requirements that should be fulfilled by a good temporal context:

(1) *Reference constrained:* as observed in Section 3, the best scenario for document classification is found when the training and test documents belong to the same moment in time, in which the characteristics of both collections (training and test) are similar with respect to the class distribution, terms distribution and class similarity. Therefore, the context X to be selected needs to consider these characteristics, that is, it needs to be temporally constrained to a given reference (in our case, the document to be classified) in order to capture the temporal characteristics of the training set associated with the moment when the test document was created. For example, if the test document is new, the reference constraint will be the current time and the temporal characteristics of the training set will be captured according to the present.

(2) *Characteristic stability:* as the temporal distances increase, we can observe changes in the characteristics of the training set and these changes affect the quality of the classification, as observed in Section 3. Therefore, the selection of context X needs to be stable with relation to the temporal characteristics of the training set (class distribution, terms distribution and class similarity) observed at the moment when the test documents were created (reference constrained), avoiding abrupt changes in the definition of classes and their terms. In other words, the characteristics of the context that will be used to classify a specific test document should be similar to the ones observed in the original training set at the moment when this test document was created. This requirement constraints the time distance among the documents of X and is related to the temporal effects which suggest that long time periods may negatively affect the performance of automatic classifiers.

(3) *Uncertainty reduction:* as discussed before, the terms and the definition of classes evolve over time, since the terms may be associated with different classes across time and the classes may also present different definitions. This evolution changes the relationship between some terms and classes and these changes may cause uncertainty in the information that terms carry about the classes. Consequently, the whole training set D may become very confuse, degrading the effectiveness of classifiers. In sum, a good context X (the goal of our search) should present both: smaller uncertainty and higher information gain when compared to the whole training set. Moreover, these contexts should contain enough features to identify and discriminate the correct classes of the documents. This requirement is related to the sampling effect discussed before and, as the second requirement, also constraints the time distance among the documents of X.

These requirements are the basis for the *Chronos* algorithm (Algorithm 1), which comprises the main steps towards determining contexts while reducing the temporal effects. As we discuss next, the decisions associated with the design and implementation of these steps will determine both the quality of the generated contexts and the computational complexity of the process.

Algorithm 1. Chronos algorithm.

```
1:  function CHRONOS $D,S,\delta$
2:     $min \leftarrow \infty$
3:     $X \leftarrow \emptyset$
4:     $state \leftarrow$ GetReference($D,S$)
5:     $option \leftarrow$ Enumerate(D)
6:     repeat
7:        if (StabilityFunction($option,state$) $< \delta$) then
8:           $uncertainty \leftarrow$ TemporalUncertainty($option$)
9:           if ($uncertainty < min$) then
10:             $X \leftarrow option$
11:             $min \leftarrow uncertainty$
12:          end if
13:       end if
14:       $option \leftarrow$ Enumerate(D)
15:    until ($option \neq \emptyset$)
16:    return($X$)
17: end function
```

Chronos receives as input two document sets, D and S, and a stability threshold δ. Collection D corresponds to the training set, in which the documents labels are known, and S contains the documents to be classified.

The algorithm addresses the first requirement in line 4. The function **GetReference**(D, S) captures the temporal characteristics of the training set D at the moment when the test documents set S was created. These characteristics comprise the class distribution, terms distribution and class similarity. The function **Enumerate**(D) in line 5 lists, one at a time, the possible contexts X that will be analyzed by the algorithm.

In line 7, the second requirement is addressed. The function **StabilityFunction**(*option*,*state*) quantifies the difference between the temporal characteristics of context *option* and those of the training set D at the moment when the test documents set S was created. This difference must be less than the threshold δ. Finally, the third requirement is addressed in line 8. The function **TemporalUncertainty** (*option*) calculates the uncertainty inherent to *option*.

The final result is that, among the possible contexts X that are stable, the algorithm selects the one with the smallest uncertainty.

4.3. Chronos instantiation issues

Ideally, a proper Chronos instantiation must satisfy simultaneously the three requirements discussed above. However, it should also take into account the traditional trade-off between effectiveness and efficiency. Clearly, the search for contexts that adhere to the three requirements generates an exponential search space. The requirement of Reference constrained requires on-demand context for each moment. Thus, for N distinct moments in a test set, we would have N different reference points. At this way, we can say that a proper instantiation demands a lazy solution[3] [39]. Moreover, the requirement for optimal Characteristic Stability context requires to identify the most stable context X regarding characteristics of each reference moment, among all possible contexts associated with that moment. As the problem of enumerating all contexts is equivalent to enumerate the number of subsets within a set, we have 2^D possible contexts for each one of the N reference point. Furthermore, each of these contexts must be evaluated for the inherent uncertainties and information gain in order to fulfill the Uncertainty Reduction requirement. Despite several pruning can be realized, mainly due to a minimum context length in reference to sampling effects, the resulting space size is still huge.

As exhaustive searches for all possibilities is prohibitive, relaxing the adherence to the three requirements is necessary. One possible relaxation strategy would be to ignore some of the aforementioned requirements. For instance, we may adopt a strategy where there is only one reference point, regardless the creation moment of the test documents. Then, the Characteristic Stability and Uncertainty Reduction requirements may be optimized by considering only a single moment. This approach could work in the particular cases where the test sets have several subsets belonging to the same period of time. Another possible relaxation strategy would be to adopt heuristic search methods. That is, assuming some simplifying assumptions, we can adopt methods that traverse the search space, in polynomial time, in order to identify good solutions, but not necessarily optimal. This strategy seems more flexible and suitable for a larger number of real-world scenarios, allowing us to exploit characteristics inherent to the algorithms or observable in many of these scenarios. For these reasons, in this work we adopt heuristic instantiation strategies for Chronos. In the following section, we present two heuristics able to properly adhere to some of the main state-of-the-art algorithms in ADC.

[3] Lazy algorithms are based on the assumption that there is not a universal classification model which is good enough in all situations and, thus, they sample the training set at classification time for constructing the model.

5. WindowChronos and FilterChronos

In this section we present two heuristics for selecting temporal contexts based on the general algorithm *Chronos*: *WindowChronos* and *FilterChronos*. The goal of both heuristics is to select a context from the training set in which the three temporal effects are minimized (reference constrained, feature stability and uncertainty reduction). It is important to mention that the main difference between these two heuristic regards the temporal contexts generated being symmetric or not symmetric. We adapt the concepts of asymmetry and symmetry presented by Faith [40], in which the “asymmetry in binary data arises when one of the two states is interpreted as more informative than the other state.”. In our case, a symmetric context uses all features (test and training terms) to select the documents that compose the context. Otherwise, in an asymmetric context only test term (terms that occur in test documents) are considered to select those documents. More specifically, *WindowChronos* generates asymmetric contexts for each test document. *FilterChronos*, on the other hand, generates symmetric contexts for each set of test documents belonging to the same temporal moment.

5.1. WindowChronos

WindowChronos is executed for each test document, naturally addressing the reference constrained requirement, since it captures the characteristics associated with just one document, more specifically its terms and creation time. The other two requirements (feature stability and uncertainty reduction) are addressed together as we describe next. We define a time-window for each term in the test document, which can grow in both directions, past and future, starting from the creation moment of the test document. The size of each window is set by the period during which the term remains “stable”. The term stability is measured by its degree of exclusivity in any class for a specified time period, which is quantified by the Dominance measure [24].

Formally, let $T = \{t_1, t_2, t_3, \ldots, t_N\}$ be the set of terms associated with a collection; $C = \{c_1, c_2, \ldots, c_K\}$ be the set of categories (classes) that occur in a collection; $df(t_i, c_j)$ be the number of documents in the training set associated with class c_j that contain t_i. We define the Dominance of the class c_j on term t_i as

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

Therefore, the Dominance quantifies both the stability and the reduction of uncertainty associated with a window. The stronger is the relationship between a term and a class, the smaller is the uncertainty.

The time-windows constructed in this way should be contiguous. We justify this choice for two reasons. First, because the relationships between terms and classes tend to change smoothly with the time distance, as discussed in Section 3. Second, because the size of the search space for determining a non-contiguous window is 2^M whereas for a contiguous window is M^2, where M is the set of moments in a temporal space.

The analysis starts at the creation moment of the test document being processed. For each term of the test document the algorithm determines the term Dominance for all classes of the training set with the same creation moment. If the Dominance of one of the classes is higher than a pre-defined threshold, this year is included in the time window. The analysis then re-starts for the adjacent years in the past and future, until the Dominance falls below the threshold. The temporal context, and thus the training set, will be constituted by the union of documents that were created in the time windows of all test terms analyzed and contain at least one occurrence of that terms. The WindowChronos strategy is detailed in Algorithm 2.

Algorithm 2. WindowChronos.

```
function WindowChronosClassify ($\mathbb{D}_{train}$, $\mathbb{D}_{test}$)
  $predictions \leftarrow \emptyset$        $\triangleright$ Predictions storage.
  for all $d' \in \mathbb{D}_{test}$ do
    $\mathbb{D}_{train}^{window} \leftarrow \emptyset$
    for all term $t \in d'$ do
      $\mathbb{P} \leftarrow$ StabilityPeriod($t$, $creationTime(d')$)
      $\mathbb{D}_{train}^{window} \leftarrow \mathbb{D}_{train}^{window} \cup \{d \in \mathbb{D}_{train} | t \in d \text{ and } creationTime(d) \in \mathbb{P}\}$
    end for
    $model \leftarrow$ Learn($\mathbb{D}_{train}^{window}$)    $\triangleright$ Learn the base classifier.
    $predictions \leftarrow predictions \cup$ Classify($model$,$d'$)
  end for
  Evaluate predictions (predictions corresponding to the entire test set $\mathbb{D}_{test}$)
end function
```

To illustrate, let us consider again the example presented in Table 2 and the test set S, consisting of documents s_1,s_2,s_3,s_4,s_5 from 1999, 2006, 2001, 2005 and 2012 in which there is an occurrence of the word *Pluto* in all of them. Consider that the algorithm will use a *Dominance* $> 50\%$ to calculate the time window for the word *Pluto* in each test document. For example, the document s_1 is from 1999 and in this year there is only one document in the training set that belongs to class Astrophysics. Thus, in this year, the term *Pluto* has a Dominance equal to 100%, therefore this year will compose the time window of term *Pluto* for document s_1. Then, the algorithm analyzes the training set in neighbor years. A similar behavior is observed in years 1998, 2000, 2001, 2002, 2003, and 2004, so that these years will also compose the time window. However, in 2005, there are two documents that contain the term *Pluto*, one of them belongs to class Astrophysics and the other one belongs to class Mythology. Therefore, in this year the term *Pluto* does not have a *Dominance* $> 50\%$. Consequently, this year will not compose the time window of term *Pluto* for document s_1 and the algorithm terminates the search for the time window of term *Pluto* for document s_1. The algorithm is executed for all terms of each test document. The time window for the word *Pluto* in each test document from test set S would be: s_1 (1998–2004); s_2 (2006–2007); s_3 (1998–2004); s_4 (empty); and 2012 (2006–2012).

After determining the time window for each test term, the last step is to select the documents that will be in the context and be used as the training set for the test document. For each test term, we select the documents that have the term and belong to its time window. Finally, we make the union of the selected documents for all terms from the test document. In our example, the training set for test documents s_1 and s_3 would be the documents d_1, d_2, d_3, d_4, d_5, d_6, and d_7 and for s_2 would be the documents d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, and d_{16}. Considering just the term Pluto, for the test document s_4 there are no documents in its temporal context. For the new document s_5 from 2012, the temporal context would contain the documents d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15} and d_{16}. This example demonstrates the procedure considering just one term, *Pluto*, for a set of test documents. In a real scenario, each document will contain several terms and each term will define a time-window with a corresponding subset of the training documents. At the end, the training set for that document will be constituted by the union of all these subsets.

It is important to mention that, despite the selection of documents be based on a minimum *Dominance*, in the final document set that will compose the temporal contexts we may observe a slightly lower *Dominance* for some terms. This occurs because the temporal contexts are the union of documents in which each term, individually, have *Dominance* higher than the minimum threshold. However, we do not verify such constraint considering all the terms at the same time. Moreover, since *WindowChronos* evaluates only test terms to define the time-windows, we say it defines *asymmetric contexts*. More formally, let T be the set of terms found in training documents and Y_i be the set of terms found in each test document d_i. We say that a context is asymmetric if it is composed by documents, selected based only in terms of Y_i, which have at least one *test* term stable. A symmetric context, on the other hand, consists of documents, selected based on terms of T, which have at least one term stable (being *test* or *training*).

5.1.1. *WindowChronos memory and time complexity*

Before starting the temporal contexts selection, *WindowChronos* performs a pre-processing in which it evaluates the number of occurrences of each term in each class, considering the entire training set. The result of this processing is a tridimensional indexed table containing the Dominance for each term in each class and year of occurrence. The procedure has an $O(U \cdot P \cdot T_p)$ complexity, where U represents the number of terms, P the number of temporal moments and T_p the number of training documents belonging to moment p. Since $P \cdot T_p$ represents the entire training set, the complexity turns to $O(U \cdot T)$, where T is the number of training documents. The memory cost imposed by this table is also $O(U \cdot T)$. Since the table is previously computed and is indexed, the access cost for any element is of $O(1)$.

After indexing the terms' Dominance, the algorithm determines the time-windows of each term in each test document. This process has a complexity of $O(V \cdot U' \cdot P \cdot K)$, where V is the number of test documents, U' is the mean number of terms per test document, P is the number of temporal moments and K is the number of classes. This complexity is set by the worst case in which the time-window depends on the evaluation of each term Dominance in each temporal moment in each class. It is important to mention that, in most cases, it is not necessary to evaluate all temporal moments since, as mentioned before, the time-windows grow only while the term Dominance is above the threshold.

The last step consists of determining the temporal contexts for each test document. This phase presents a complexity of $O(V \cdot U' \cdot T)$ in the worst case. The cost is related to the task of selecting all documents that contain the test term. In the worst case, the time-window would encompass all the temporal moments of the training base, what would imply evaluate all its documents.

Therefore, the total complexity for *WindowChronos* is $\max\{O(U \cdot T), O(V \cdot U' \cdot P \cdot K), O(V \cdot U' \cdot T)\}$. Considering that the number of terms and documents are much higher than the other parameters, this equation can be simplified to $O(U \cdot T)$. For the worst case in which the number of training terms is equal to the training documents, the complexity turns to be a quadratic function of the number of terms. This is also the memory complexity, since the indexed table is previously computed and stored in memory. As previously mentioned, in practice this cost is lower since the time-windows tend to be smaller.

5.2. FilterChronos

Like *WindowChronos*, *FilterChronos* defines temporal contexts by performing a time-sensitive selection of documents for training. The procedure also relies on the construction of time-windows that may grow in both directions, past and future. But, differently from *WindowChronos*, *FilterChronos* constructs its time-windows based on a *set* of reference documents, not just one. In fact, the *FilterChronos* is an evolution of the *WindowChronos* introducing two fundamental differences with the goal of improving the performance compared to the previous approach. First, each temporal context is now related to several documents created at the same moment, which improves the generalization capability. Second, the time-windows are defined by documents from both the training and test sets, making the process symmetric, which is a premise of several ADC algorithms.

The definition of temporal contexts is based on the evaluation of the terms' stability as the time distances increases. The procedure starts by grouping all documents, for both training and test sets, according to their respective creation moments. This strategy addresses the reference constrained requirement since all test documents from m_i have the same reference point. In this heuristic, each temporal context is also determined by a time-sensitive selection of the documents for training, that is, it selects as training set, for the set of test documents belonging to each distinct temporal moment, all documents that are closer in time to it. The proximity is defined by a time-window that may grow in both directions, past and future, based on the temporal moment of the set of test documents.

To determine which temporal moments will compose the time-window of the set of test documents, the stability of the characteristics on these moments, related to the characteristics of the base moment m_i, need to be calculated. Therefore, after separating the documents according to their creation moment, for both test and training sets, for each base moment m_i, we obtain a list of all terms that occur in it. For each term, starting from the base moment m_i, we calculate the period during which they remain stable, where the term stability is measured, as in *WindowChronos*, by Dominance (Eq. (1)). At the end, each term has a list of temporal moments during which it remains stable. Then, we calculate the percentage of occurrence of the temporal moments in the time-window of all terms. These moments are ranked according to this percentage and the N temporal moments with the highest percentage of occurrence will be the moments that will compose the time-window of the set of test documents of m_i. Thus, this heuristic considers all terms that occur in each temporal moment, not only the test terms as in *WindowChronos*. Moreover, using this strategy, we address the characteristics stability requirement, since the stability of all terms is evaluated in order to determine the temporal moments. This heuristic is detailed in Algorithm 3.

Algorithm 3. FilterChronos.

```
function FilterChronosClassify ($\mathbb{D}_{train}$, $\mathbb{D}_{test}$)
  $\{\mathbb{D}^p_{test}\} \leftarrow$ PartitionPerTimepoint($\mathbb{D}_{test}$)
  $predictions \leftarrow \emptyset$        ▷ Predictions storage.
  for all $\mathbb{D}^p_{test}$ do
    $\mathbb{P}_K \leftarrow$ top-K best ranked timepoints according to the reference point $p$
    $\mathbb{D}^{filter}_{train} \leftarrow \{d \in \mathbb{D}_{train};|;creationTime(d) \in \mathbb{P}_K\}$
    $model \leftarrow$ Learn($\mathbb{D}^{filter}_{train}$)        ▷ Learn the base classifier.
    $predictions \leftarrow predictions \cup$ Classify($model, \mathbb{D}^p_{test}$)
  end for
  Evaluate predictions (predictions corresponding to the entire test set $\mathbb{D}_{test}$)
end function
```

To illustrate how the heuristic works, consider a collection consisting of documents from different years (the temporal moments in this example are discretized in years). Suppose that in this collection there is the year 1995 and their documents contain the terms $(t_1, t_2, \ldots, t_{10})$. As described, the first step consists of determining the time window of each term, starting from the base year 1995. Table 3 presents one possible configuration for the temporal windows of the terms. While the time window of the term t_3 is 1993–1998, the time window of t_8 is empty. This means that t_8 was not stable enough in any period considering the assumed value of Dominance. Table 4 shows the result of the second phase, which determines the percentage of occurrence of each year in the time windows of Table 3. For example, the year 1996 occurred in 50% of the time windows, while 1990 occurred in only 10%.

Table 3
Set of terms of a base year.

Base Year: 1995			
Term_Id	Window	Term_Id	Window
t_1	1995–1997	t_6	1994–1995
t_2	1993–1996	t_7	1995–1995
t_3	1993–1998	t_8	–
t_4	1990–1999	t_9	–
t_5	1994–1996	t_{10}	1994–1997

Table 4
% of occurrence of years in temporal window of terms.

Base Year: 1995			
Year	% of occurrence	Year	% of occurrence
1995	80	1998	20
1996	50	1999	10
1994	70	1992	10
1997	40	1991	10
1993	30	1990	10

In order to find the best value of N and to address the uncertainty reduction requirement, we propose to split the training set into training and validation, on which the documents of the validation set are also discretized. Then, for the documents of each base moment m_i in the validation set, the value of N need to be varied from 1, that is, the most frequent temporal moments in the time-windows of all terms (top of the rank), to Y, where Y represents the total number of moments that occur in the stability period of all terms. The value of N is defined as the value that maximizes the effectiveness of the classifier in the validation documents that belong to moment m_i. Finally, a temporal context for each base moment m_i is determined by the set of documents from the whole training set that belong to its time-window, defined by the value N. These temporal contexts, which are symmetric, are used as training set for the test documents of each m_i in order to create its classification models.

Returning to the previous example, the first value of N to be evaluated is 1, as previously presented. From Table 1 we have that the most frequent year in the time windows of terms of 1995 is the year 1995 itself. Therefore, using only the documents from 1995 in the training set we generate a classification model and we evaluate it using the documents of the validation subset from 1995. The next value to be evaluated is 2, that is, the two years that occurs more often in the time windows of terms of 1995 (1995 and 1996). Then, using only the documents from 1995 to 1996 in the training set we generate a classification model and we evaluate it using the documents of the validation subset from 1995. This process is repeated for all possible values of N (10 in our case) and the value of N that presents the best results in the classification of the validation set will be used to classify the test set. In our example, assuming that the value of N which achieved the best results in the validation set was 4, we have that the temporal context of the test documents of 1995 will be composed by training documents that belong to the years 1995, 1996, 1994 and 1997, the four most frequent years in Table 4.

5.2.1. *FilterChronos memory and time complexity*

Like *WindowChronos*, *FilterChronos* starts by constructing an indexed tridimensional table containing the Dominance of each term in each year and class. The computational and memory costs are the same, thus of $O(U \cdot T)$.

Based on this table, the algorithm determines the time-windows for all the terms of training and test documents. The procedure has a cost of $O((U+U') \cdot P \cdot K)$, where $U+U'$ is the number of terms in the collection, P is the number of temporal moments and K the number of classes. The result of this processing is a second indexed table containing the frequency of occurrence of all the temporal moments in the time windows of all terms P_i and for each temporal moment P_i. The memory cost is of $O(P^2)$. The last step consists of calculating the temporal context for all the test sets of each moment P_i. This step has a complexity of $O(P^2 \cdot Class)$, where *Class* is the complexity of the classification algorithm employed.

Therefore, the total complexity of *FilterChronos* is given by $\max\{O(U \cdot T), O((U+U') \cdot P \cdot K), O(P^2 \cdot Class)\}$. Since the complexity of a classification algorithm is usually much higher than the other terms, the equation can be simplified to $O(P^2 \cdot Class)$. The memory cost is defined by the two indexed tables and is of $O(U \cdot T + P^2)$. Despite both costs being superior than for *WindowChronos*, *FilterChronos* presents benefits which justify its adoption, as will be shown next. Moreover, in Section 6.4.1 we present a simplification in the computation of temporal contexts aiming to reduce this cost without decreasing the classification accuracy.

6. Experimental setups and results

6.1. *Experimental setup*

This section presents the results obtained with four ADC algorithms using the temporal contexts selected by *WindowChronos* and *FilterChronos*. The goal is to evaluate how the traditional classification techniques can benefit from temporal context selection. To this end, we chose the most traditional and referred ADC algorithms in the literature, based on different approaches. More specifically, the algorithms were kNN (a lazy approach), Rocchio (a vector space approach), and Naïve Bayes (a probabilistic approach), implemented in the Libbow classification software [41]. We also employed SVM-Perf [42], a package that implements a linear SVM method (a maximum-margin approach) which can be trained in linear time. We used the one-against-all approach [43] in order to adapt the binary SVM classifier to our collections, which contain more than two classes. Although there are other methods to adapt the SVM, like a pairwise classification approach, we chose the one-against-all since it is the earliest and one of the most widely cited approaches [44,13] and has a shorter runtime. In all cases, we explored the information about intra-document term frequency (TF).

The experiments were performed considering the two collections presented in Section 1. However, in order to reduce the complexity, for MedLine we considered only documents from 1970 to 1985, forming a base with 861,454 samples.

The effectiveness was evaluated using standard information retrieval measures [45]: accuracy and macro average F1 (MacroF1). While accuracy measures the classification effectiveness globally over all decisions, the MacroF1 measures the effectiveness for each individual class and averages them.

6.2. *Evaluating the dominance metric*

Dominance is a fundamental metric and the basis for the algorithms proposed here. Therefore, before evaluating *WindowChronos* and *FilterChronos*, we performed a set of experiments to evaluate its potential. The experiments consist of determining the temporal context (and then the training set) associated with each test document, when the Dominance varies from 0% (no temporal information is considered) to 90%. The results for Dominance greater than 90% will not be presented since the resulting contexts are very sparse (most of them empty). In all experiments we employed a 10-fold cross-validation [46] and the final result of each one is the average of the 10 runs.

We started the evaluation by analyzing the reduction in the size of the selected contexts as a function of Dominance. The metric in this case is called *total terms*, which is the sum of the number of terms in the selected contexts for all test documents. This metric quantifies the effort of a classifier to build a model. The results for 0% Dominance were used as a baseline and all other results were relative to them. The results are shown in Fig. 8.

As can be seen, in both collections, the use of temporal contexts reduces significantly the number of terms in the training sets. For example, by using a Dominance of 90%, the total number of terms for ACM-DL is less than 10% of the baseline, while for MedLine it is less than 3%. It is interesting to note that there are some critical points from which increasing Dominance drastically reduces the number of terms. This is approximately 20% for ACM-DL and 40% for MedLine. This difference can be explained by the smaller number of classes in MedLine (7) when compared to ACM-DL (11).

We also evaluate the number of distinct terms, that is, the union of the sets of terms of the training sets for each Dominance value, which is presented in Fig. 9. There is a small reduction on the number of distinct terms in ACM-DL as we increase the Dominance, while in MedLine this reduction is very significant. Analyzing these last graphs together with the graphs in Figs. 8 and 5, we can conclude that the training sets for the test documents overlap less in the ACM-DL than in MedLine, since the terms in ACM-DL have a stability period usually smaller than in MedLine, that is, the ACM-DL is more affected by the time effects as presented.

Thus, Dominance presents a beneficial effect related to the reduction of total terms in the training set, which leads to a reduction on the effort to construct a classification model. Nevertheless, it is also necessary to evaluate if this benefit is accompanied by a reduction in the uncertainty for the sake of classification. To this end, we evaluated the entropy [47] of each training set selected by *WindowChronos* while increasing the Dominance. Entropy, in our case, is understood as the necessary information to build a classification model. This metric quantifies the terms dispersion among classes, measuring its disorder. The average of the results is presented in Fig. 10. It can be seen that there is a significant decrease in entropy as Dominance is increased. Thus, this parameter reduces both the uncertainty and the effort to build a classifier.

We confirmed this last observation by analyzing the number of distinct classes that are associated with each test term in its respective training set. The results of this analysis for each collection is presented in Fig. 11. It can be seen that an increase in Dominance causes a decrease in the number of classes associated with a term. For example, when using 0% Dominance, each test term occurs, on average, in 8 distinct classes in the training sets for ACM-DL and 5 for MedLine. When Dominance is increased to 90% this number reduces to less than 2 for both collections. This result shows that the temporal contexts determined by the Dominance reduces the uncertainty of the training sets. However, it is still

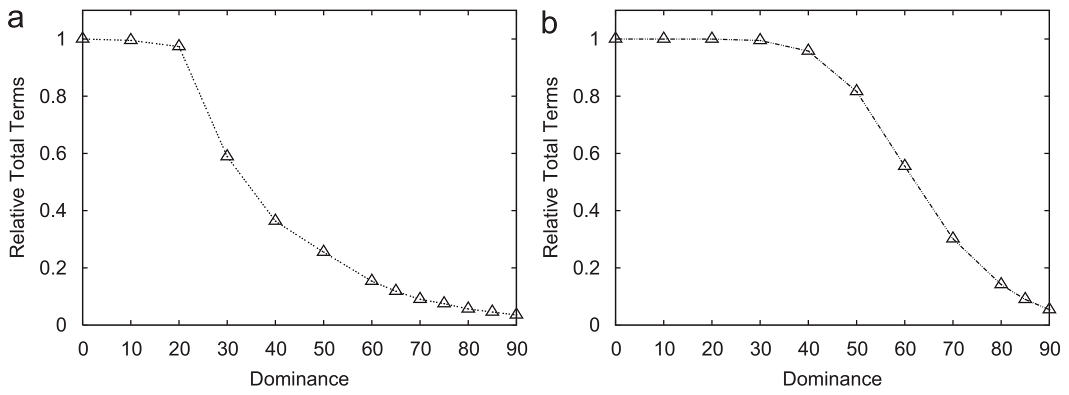

Fig. 8. Effort reduction. (a) ACM collection (b) MedLine collection.

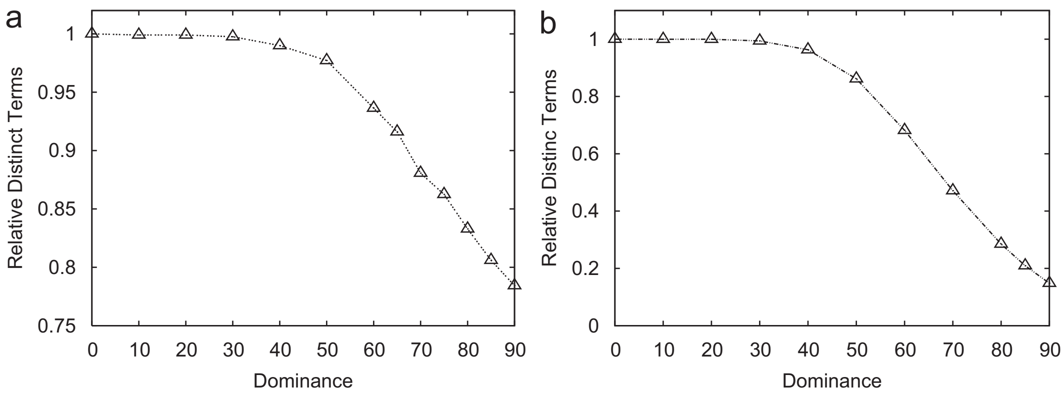

Fig. 9. Distinct terms. (a) ACM collection. (b) MedLine collection.

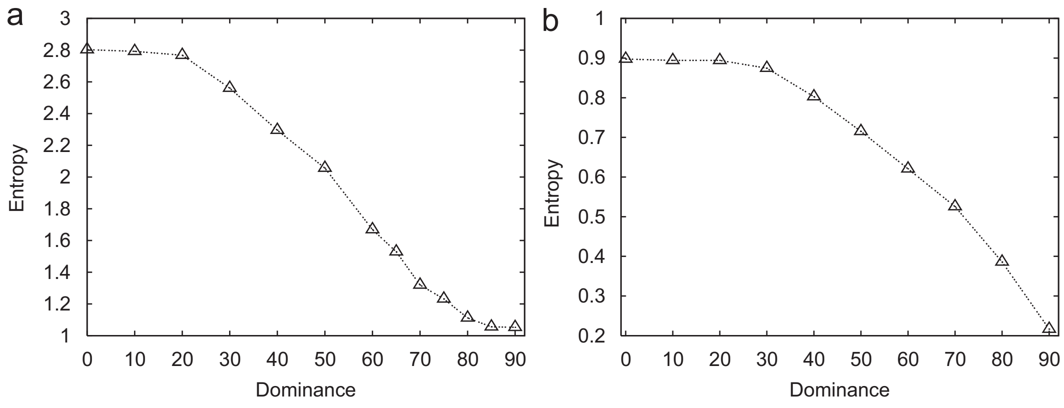

Fig. 10. Uncertainty reduction. (a) ACM collection (b) MedLine collection.

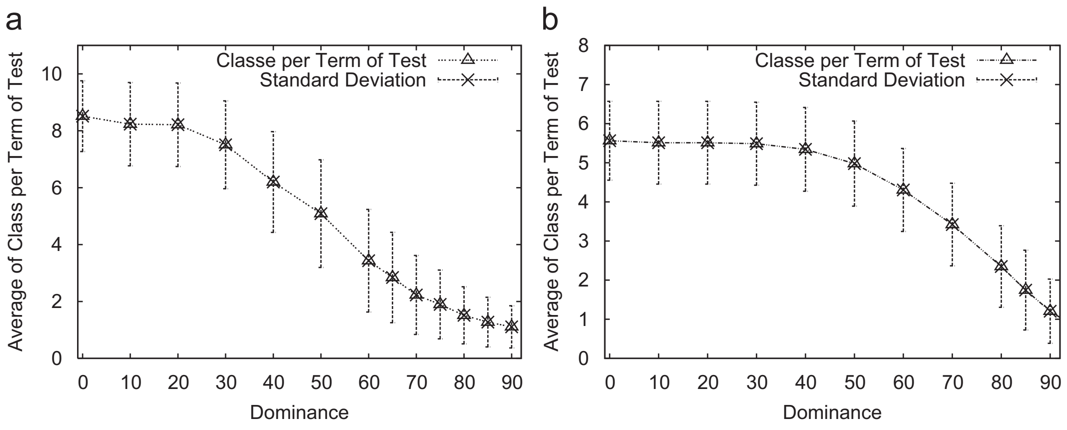

Fig. 11. Occurrence of test terms in distinct classes. (a) ACM collection. (b) MedLine collection.

necessary to analyze whether this reduction contributes to determine the correct class of the test document when applied with different ADC algorithms.

6.3. *Evaluating WindowChronos*

First we performed a set of experiments that determine the temporal context associated with each test document and evaluate them with several ADC algorithms. In order to do this, we split the training set into training and validation. Then, we looked for Dominance values (with folded cross-validation) that provide the best results in the validation set for each combination of algorithm and collection, since specific characteristics of each collection and algorithm may affect the choice of Dominance. We applied the *WindowChronos* heuristic to produce the temporal contexts for the test documents using the best Dominance values found in each case. In all experiments we employed a 10-fold cross-validation and the final results is the average of the 10 runs.

The results achieved using temporal contexts selected by *WindowChronos* are presented in Table 5. The lines with heading “bl.” (base line) present values for executions using the complete training set without taking into account temporal issues, while the lines with heading “tc.” present values for classifications using the temporal contexts. The lines with the heading “gain” provide the percentage difference between the classification with the temporal contexts and the baselines for each metric. Finally, the lines with “t-t.” describe whether the variations produced by our methodology (“tc.”) with respect to the baselines (“bl.”) represent a statistically significant difference, given a 99% confidence in a 2-tailed *t*-test. ▲ denotes a significant positive variation, • a non-significant variation and ▼ a significant negative variation.

As we can see, the temporal contexts selected for *WindowChronos* are a good alternative for improving kNN. In the temporal contexts the discriminative power of the terms from test documents is increased, given that their ambiguity with respect to classes is reduced. Consequently, these improvements can be explained since this algorithm, being naturally lazy, uses only the test terms in the classification process, based on asymmetric premises.

However, the results also showed that this technique may not be ideal to be applied along ADC algorithms based on symmetric premises and, consequently, use directly (e.g. SVM) or indirectly (e.g. Naïve Bayes, Rocchio) all terms. Despite the fact that the confusion regarding the ambiguity of test terms among the classes is reduced in the temporal contexts selected by *WindowChronos* heuristic, the same cannot be guaranteed for the remaining terms that are included in the temporal contexts. Consequently, the premises of the algorithms that consider these terms can be affected (as we will see in the next section, this fact is not observed in *FilterChronos* that generate symmetric temporal contexts). We discuss this issue further for each selected algorithm.

Naïve Bayes [48] is a probabilistic classifier, which computes the score of a class c_i as the probability of a document d_j be assigned to c_i. This probability is measured using the Bayes Theorem and is defined by Eq. (2). We can observe that the key point in calculating $P(c_i|d_j)$ is how to estimate $P(t|c_i)$. In our experiments we used the Multinomial [48] (more traditional in statistical language modeling for text classification) in which $P(c_i|d_j)$ is defined as presented in Eq. (3), where $Tc_i(t)$ is the number of occurrences of term t in class c_i and $\sum_{t'} Tc_i(t')$ is the sum of the number of occurrences of all terms that occur in document of class c_i.

$$P(c_i|d_j) = \frac{P(c_i)\prod_{t\in d_j} P(t|c_i)}{P(d_i)} \tag{2}$$

$$P(t|c_i) = \frac{Tc_i(t)}{\sum_{t'} Tc_i(t')} \tag{3}$$

It can be concluded that $P(t|c_i)$ defines the representativity of a term t in a class c_i as the ratio between the frequency of term t in class c_i and the total frequency of all terms in class c_i. Thus, Naïve Bayes is based on symmetric premise that the representativity of test terms in a given class is based on all terms that occur in that classes, that is, the probability of test terms in a given class is affected by the remaining terms that occur in temporal contexts. Consequently this premise is seriously affected.

Rocchio is based in the symmetric premise that all features are used to create the prototype vector for each class. Each training document is represented as a weighted vector usually created using the tf-idf weighting scheme [49]. Class prototypes are generated by a vector summation of the vector representations of all documents of a given class in the training set. To classify a test document, the distance between the vector that represents the test document and the prototype vectors that represent each class is calculated. Since the temporal contexts selected by *WindowChronos* are asymmetric, the idf values of test terms will be lower than those of the remaining terms in the temporal contexts. While we can say that there exists at least one test term in all documents that compose the temporal context of each test document, this statement is not true regarding the occurrence of other terms and consequently they tend to be less frequent in temporal contexts. Consequently, in the prototype vectors that represent each class, the weight of the test terms is lower than the remaining terms. When the vector distance between the test documents and these prototype vectors is calculated, the test terms have less influence than they really should have, which is exactly the opposite effect we want.

Support Vector Machines are based on the Structural Risk Minimization principle from computational learning theory [43]. According to [50], the main idea of structural risk minimization is to find a hypothesis *h* for which we

Table 5
Impact of temporal contexts (*WindowChronos*).

Collection	ACM-DL		MedLine	
Metric	macF$_1$ (%)	acc. (%)	macF$_1$ (%)	acc. (%)
kNN				
bl.	56.78	69.82	66.57	79.82
tc.	60.10	72.31	69.48	82.46
gain	+5.84	+3.47	+4.37	+3.31
t-t.	▲	▲	▲	▲
Naïve Bayes				
bl.	56.87	74.00	65.64	79.65
tc.	58.90	73.30	66.68	80.36
gain	+3.47	−0.96	+1.75	+0.44
t-t.	▲	•	▲	•
Rocchio				
bl.	56.97	67.95	54.14	69.36
tc.	53.25	61.65	52.21	65.54
gain	−6.53	−9.28	−3.56	−5.51
t-t.	▼	▼	▼	▼
SVM				
bl.	60.07	73.03	72.28	83.27
tc.	56.60	71.19	67.15	80.13
gain	−5.78	−2.52	−7.10	−3.77
t-t.	▼	▼	▼	▼

can guarantee the lowest true error. The true error of h is the probability that h will not classify correctly an unseen and randomly selected test example. An upper bound can be used to connect the true error of a hypothesis h with the error of h in the training set and the complexity of H (i.e. the hypothesis space), as measured by the VC-Dimension. Support vector machines find the hypothesis h, which minimizes this bound on the true error by effectively and efficiently controlling the VC-Dimension of h. This optimization process corresponds to the learning process. At the end of the learning process, the SVM gives weights to each feature in each class. High positive weights indicate that documents with these features are probably related to the associated class, whereas negative weights mean that they are not. All features and their weights, positive or negative, are used by SVM to classify unlabeled documents, which is clearly a symmetric premise. Therefore, the information that is provided by the terms that occur in temporal contexts, but are not test terms, may be distorted if compared with the results using the entire collection and this fact contributes to degrade the results of SVM.

6.4. *Evaluating FilterChronos*

In *FilterChronos* the first step to select the temporal contexts is to determine the stability of all terms for each set of test documents from each year A_i, which is calculated using the Dominance metric. In order to determine the best value of Dominance, we performed an experiment in which we varied its levels from 50%[4] to 90%. We generated the histograms H_i for each A_i with the number of occurrences of the years in the stability period of the terms (with folded cross-validation). By analyzing these histograms for each collection, we observed that their probability distributions were very similar, that is, the years with the highest percentage of occurrence in H_i were almost the same for the various Dominance levels. Only the magnitudes of the percentages of occurrence vary significantly.

In Fig. 12 we summarize this analysis for both ACM and MedLine collections, using the Dominance metric equal to 50% and 80%, respectively. Each bar is the average of the popularity for each time distance, considering all pairs of years to which that distance is applicable. The popularity is the percentage of all occurrences that compose the stability period of the terms that occur in given year A_i.

Observing Fig. 12 it can be seen that, for both collections, the years that are closer to the base year are the most frequent ones with respect to the stability period of the terms. As the time distance increases (for the past and future), the frequency decreases, that is, there are few terms that remain stable for a long period. However, we can note that this decay is smoother in MedLine than in ACM-DL. Also, observing the difference between the percentage of occurrence of the most and least frequent years in each collection, it can be seen that this value is much higher in ACM-DL. These findings can be explained by the fact that the ACM-DL collection is much more affected by temporal effects than MedLine, as discussed before. Moreover, observing the decay of the popularity as the time distance increases, it can be seen that in the ACM-DL this decay is more symmetric than in MedLine (in which the future years decay slower than past years). This is a consequence of more new terms being introduced over the years in MedLine and, consequently, the stability of these new terms being larger for years after they are introduced.

Comparing Fig. 12 with Fig. 3 (Temporal Locality Variation) and Fig. 5 (Terms Distribution Means), presented in Section 3, we can see again that Dominance is a very good metric to determine the stability period of the terms. Moreover, the time distance with the highest percentage of occurrence in the graphs related to Dominance 50 and 80 were almost the same in both collections. Therefore, any Dominance value above or equal to 50% can be used in order to determine which years will compose the temporal contexts for the set of test documents of each base year A_i. Intuitively, this is the expected behavior for other document's collections, since the quality of the classifiers is directly related to the stability of the characteristics in the collections, which is ensured by a Dominance greater than or equal to 50%.

To evaluate *FilterChronos* we performed a set of experiments to determine the temporal contexts associated with the set of test documents for each year and for each ADC algorithm. As previously described, we split the training set into training and validation. Then, using a 50% Dominance value with a 10-fold cross-validation, we determined the number N that performs better in the validation set for each combination of algorithm and collection. We did this because specific characteristics of each collection and algorithm may affect the choice of N. Then, we used the best N to produce the temporal contexts for the test documents. In these first experiments, the values of N were found by a complete search of alternatives, varying it from 1 to Y (e.g., in ACM there are 22 base years, and, consequently, 484 alternatives were evaluated). The results obtained in each scenario are presented in Table 6.

It can be observed that the temporal contexts selected by *FilterChronos* improved the results of all algorithms. It can also be seen that the algorithms kNN, Rocchio and Naïve Bayes approached SVM's effectiveness, considered the state-of-the-art. In particular, the large improvements in kNN can be explained by the fact that, as before, in these temporal contexts the discriminative power of terms is enhanced. The improvement over the other algorithms can be explained by the fact that *FilterChronos* considers all terms that occur in each year, thus generating symmetric temporal contexts.

All four algorithms create, for each test document, a ranking of classes based on the assigned scores and classifies the test document to the best ranked class. In order to analyze in more details the impact of temporal contexts selected by *FilterChronos*, we inspected how they changed these ranks by analyzing how the rank of the

[4] This value ensures that, regardless of the number of classes of a collection, a term will have a degree of exclusivity in one class.

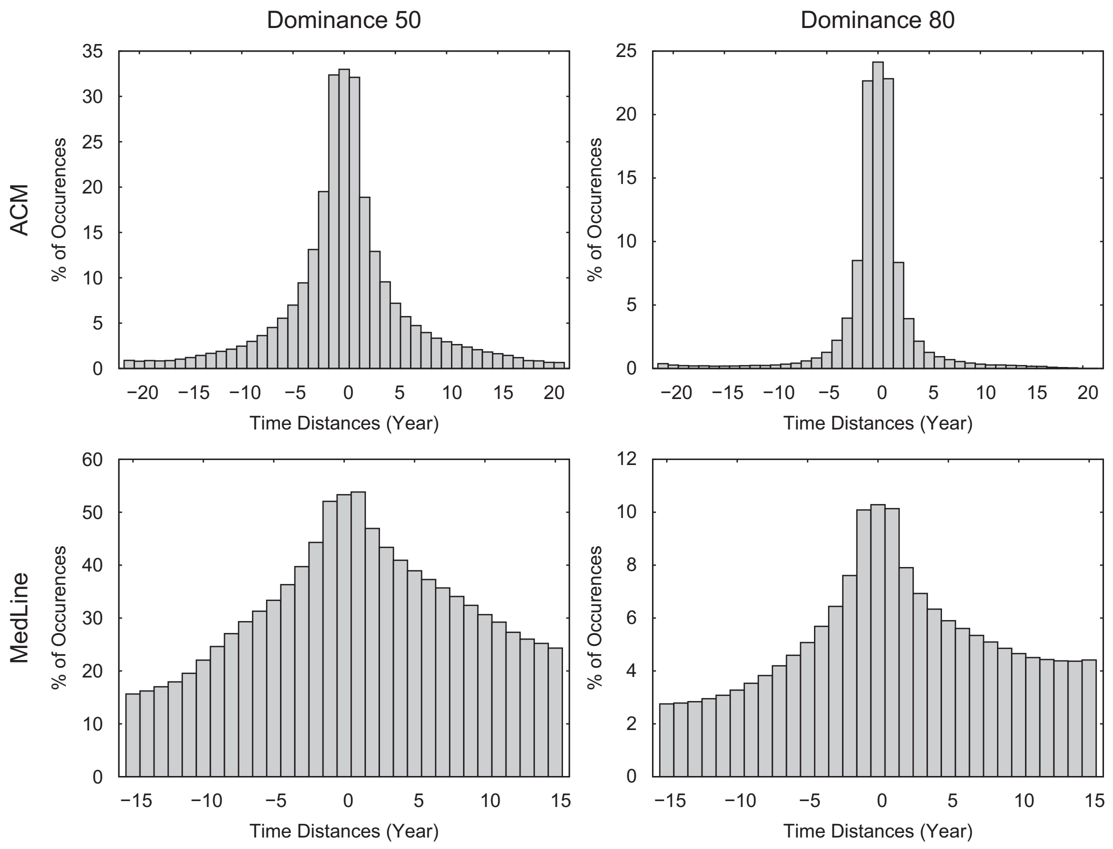

Fig. 12. Popularity of years in temporal window of terms.

true class of the document is affected. We defined a variable *Z*, which represents the number of positions in the rank gained or lost by the true class when using the temporal contexts. For evaluating *Z*, documents correctly classified were not taken into account. Thus, more formally, we have $X(d_t) = Pos_{t.c.}(d_t) - Pos_{bl.}(d_t)$, where $Post_{bl.}(d_t)$ is the rank of the true class for the test document d_t by using all training set and $Pos_{t.c.}(d_t)$ is the analog as before, but using temporal contexts.

We evaluated the variable *Z* for both collections and for all algorithms. We quantified the skewness of the distributions using the Third Standardized Moment's Mode Skewness [51]. The measured skewness was positive in both collections, as shown in Table 7. This fact demonstrates that, for all the algorithms in both collections, the contexts selected by *FilterChronos* generate a positive shifting which is strong enough to improve the rank of true classes, so that the correct class is predicted in more cases.

6.4.1. *Optimizing FilterChronos*

A complete search for the best value of *N* demands the evaluation of a significant number of values. A strategy to reduce the computational cost is to prune alternatives. Towards this end, we performed the following set of experiments. As before, we split the training set into training and validation, and using a 50% Dominance value with a folded cross-validation, we searched for the value of *N* that produces the best results in the validation set for each combination of algorithm and collection. However, in the optimized version, we stopped the search when the performance of the classifier does not improve after a number of pre-defined attempts (5 attempts for ACM and 3 attempts for MedLine).

The results achieved in our experiments for each scenario are presented in Table 8. The lines with heading "tc." present values for classifications without pruning, and the lines with heading "Ptc." present values for executions applying pruning. The lines with the heading "diff." provide the percentage of difference between the classification with and without pruning. Finally, the lines with the "t-t." describe whether the variations are statistically significant, given a 99% confidence in a 2-tailed *t*-test.

The use of pruning in the search for the best value of *N* provided an average reduction of 32% in the total number of evaluated alternatives. Moreover, as can be observed, for all algorithms in both collections, the pruning strategy has a good approximation to the solution that assesses all possibilities. This can be explained by the fact that the relationship between terms and classes tends to change smoothly, as discussed in Section 3. Consequently, the years that compose the best value of *N* tend to be contiguous and temporally closer to the set of test documents.

Finally, we have performed a last set of experiments in order to demonstrate the effectiveness of employing the *Dominance* of the terms in order to search for the best

Table 6
Impact of temporal contexts (*FilterChronos*).

Collection	ACM-DL		MedLine	
Metric	macF$_1$ (%)	acc. (%)	macF$_1$ (%)	acc. (%)
kNN				
bl.	56.78	69.82	66.57	79.82
tc.	61.90	74.44	68.97	81.87
gain	+9.01	+6.53	+3.61	+2.57
t-t.	▲	▲	▲	▲
Naïve Bayes				
bl.	56.87	74.00	65.64	79.65
tc.	59.76	76.56	68.95	81.88
gain	+5.09	+3.45	+5.04	+2.80
t-t.	▲	▲	▲	▲
Rocchio				
bl.	56.97	67.95	54.14	69.36
tc.	61.66	72.68	56.99	71.71
gain	+8.24	+6.95	+5.26	+3.39
t-t.	▲	▲	▲	▲
SVM				
bl.	60.07	73.03	72.28	83.27
tc.	62.32	76.50	73.98	84.90
gain	+3.75	+3.80	+2.35	+1.96
t-t.	▲	▲	▲	▲

Table 7
Skewness values.

Algorithm		kNN	Rocchio	Naïve Bayes	SVM
Collection	ACM-DL	+0.09	+0.30	+0.10	+0.20
	MedLine	+0.20	+0.10	+0.09	+0.10

window size. To this end, we implemented and evaluated a technique, which we call the Simple Temporal Context Selection (st.), that selects temporal contexts without considering the *Dominance*. More specifically, it selects training documents according solely to the temporal distance to a reference moment (i.e., the creation time of the test document). In this case, the time window grows one temporal unit per time, to the past or to the future starting from the year of the test document. The window size is optimized through cross-validation over the training set, varying the time window size from 0 (which selects training documents created at the same year of the validation documents) to *Y* (which, in turn, symmetrically selects training documents created at *Y* time units away from creation moment of the validation documents). The optimized window size is thus the value that minimizes the cross-validation error. A complete search for the best window size demands the evaluation of a significant number of values. Therefore, as performed in the optimized version of FilterChronos, we also stop the search when the performance of the classifier does not improve after a number of pre-defined attempts (5 attempts for ACM and 3 attempts for MedLine). A temporal context for each reference moment is determined by the set of documents from the whole training set that belong to its optimized time-window.

The results achieved in our experiments for each scenario are presented in Table 9. The lines with heading "st." present values for executions using the Simple Temporal Contexts Selection, and the lines with heading "Ptc." present values for classifications using the optimized version of FilterChronos. The lines with the heading "gain" provide the percentage of difference between the classification using Simple Temporal Selection Contexts and optimized FilterChronos. Finally, the lines with the "t-t." describe whether the variations are statistically significant, given a 99% confidence in a 2-tailed *t*-test.

As we can observe in Table 9, the temporal contexts selected by Simple Temporal Contexts Selection also

Table 8
Impact of using prune in *FilterChronos*.

Collection	ACM-DL		MedLine	
Metric	macF$_1$ (%)	acc. (%)	macF$_1$ (%)	acc. (%)
kNN				
tc.	61.90	74.44	68.97	81.87
Ptc.	61.56	74.15	68.94	81.81
diff	−0.54	−0.39	−0.04	−0.07
t-t.	•	•	•	•
Naïve Bayes				
tc.	59.76	76.56	68.95	81.88
Ptc.	59.21	76.23	68.75	81.78
diff	−0.92	−0.43	−0.29	−0.12
t-t.	•	•	•	•
Rocchio				
tc.	61.66	72.68	56.99	71.71
Ptc.	61.44	72.35	56.90	71.68
diff	−0.36	−0.45	−0.16	−0.04
t-t.	•	•	•	•
SVM				
tc.	62.32	76.50	73.98	84.90
Ptc.	62.03	76.01	73.75	84.40
diff	−0.64	−0.47	−0.31	−0.59
t-t.	•	•	•	•

Table 9
The importance of *Dominance* in the search for temporal contexts.

Collection	ACM-DL		MedLine	
Metric	macF$_1$ (%)	acc. (%)	macF$_1$ (%)	acc. (%)
kNN				
st.	58.56	72.79	69.78	81.63
Ptc.	61.56	74.15	68.94	81.81
gain	+5.12	+1.87	−1.20	+0.22
t-t.	▲	▲	▼	•
Naïve Bayes				
st.	57.88	74.09	67.05	81.60
Ptc.	59.21	76.23	68.75	81.78
gain	+2.30	+2.89	+2.54	+0.22
t-t.	▲	▲	▲	•
Rocchio				
st.	58.28	70.02	55.00	70.04
Ptc.	61.44	72.35	56.90	71.68
gain	+5.42	+3.33	+3.45	+2.34
t-t.	▲	▲	▲	▲
SVM				
st.	62.73	74.62	73,06	83,41
Ptc.	62.03	76.01	73.75	84.40
gain	−0.97	+1.83	+0.94	+1.19
t-t.	•	▲	•	▲

improved the results of almost all algorithms. This result shows that even a simpler strategy for selecting temporal contexts is a good alternative to improve the effectiveness of classifiers. However, the results achieved by the optimized version of *FilterChronos* are better in most of the cases, demonstrating that a heuristic in which the search for temporal contexts is based on the *Dominance* of the terms is more effective.

6.4.2. Application of FilterChronos in concept drift scenarios

As previously mentioned, despite not being the focus of the present work, the techniques described here can naturally be used in Concept Drift scenarios. In this section we present a possible use of the *FilterChronos* technique, which achieved the best results in the previous experiments, in such a scenario. We compare it with a widely used strategy to cope with Concept Drift problems.

Unlike in batch classification, in a typical concept drift scenario, the data comes in batches of documents, chronologically ordered. Let $d_{(i,j)} = (\overrightarrow{x}, y)$ be the *i*th document of batch *j*, where $\overrightarrow{x}$ denotes its bag of words vector representation and y its class. Furthermore, let n_j be the number of documents in batch *j*. We have the following configuration:

$$\underbrace{d_{(1,1)},\ldots,d_{(n_1,1)}}_{Batch\ 1};\underbrace{d_{(1,2)},\ldots,d_{(n_2,2)}}_{Batch\ 2};\cdots;\underbrace{d_{(1,t)},\ldots,d_{(n_t,t)}}_{Batch\ t}$$

The goal is to select a subset of batches $\{t' | t' < t\}$ to be used as training data, optimizing the effectiveness when classifying data from batch *t*.

We assume that inside a batch, the documents are independently and identically distributed (i.i.d.) with the same underlying probability distribution. The challenge here is that, across distinct batches, such probability distributions may not be the same (e.g., due to the temporal effects) and, if not carefully considered, it may degrade the classification effectiveness.

Several strategies were already proposed to deal with drifting concepts in automatic classification. Perhaps the most adopted ones are the window-based strategies. When classifying documents of batch *t*, these strategies determine a window size which selects the batches $t' < t$ to be used as training data (with the hope that outdated documents, which may hurt effectiveness, will be filtered out). Here we consider the Adaptive Time Window approach, proposed in [19]: for each batch *t*, this strategy finds an optimized window size $sz_t > 0$ to select the batches $\{t' < t | |t-t'| \le sz_t\}$ to compose the training set. Then, a classifier is learned to classify data from batch *t*, while being robust to drifting concepts. As argued in [19], unlike fixed window strategies, the adaptive window strategy can cope with varying speeds and amounts of drift, since it can adjust the window size according to the current extent of drift.

In order to assess the effectiveness of *FilterChronos*, we simulated the concept drift scenario using both reference datasets (ACM-DL and MedLine). In this case, each batch corresponds to a single year (again, assuming that documents created in the same year follow the same underlying distribution). We predefined a split time point p_s denoting the year the test batches arrive to the classifier. For each time point $p > p_s$, we learned a traditional SVM classifier (using as training data all batches $p' < p$), as well as the adaptive time window and *FilterChronos* classifiers (using SVM as the base learner). All three strategies were calibrated through cross validation over the training set. This process was repeated $r = 10$ times, in order to achieve statistical significance. For each test batch, we evaluated the classification effectiveness via accuracy and MacroF1. The experimental setup is summarized in Algorithm 4.

Algorithm 4. Concept drift: simulation design.

```
function DRIFTINGEVALUATIONdataset $\mathbb{D}$, split_timepoint $p_s$, replications $r$
  $p_f \leftarrow$ LASTTIMEPOINT($\mathbb{D}$)
  for all $p = p_s$ to $p_f$ do
    for all $i = 1$ to $r$ do
      $\mathbb{D}_s \leftarrow$ RANDOMSAMPLE($\mathbb{D}$,70%)
      $\mathbb{D}^i_{trn} \leftarrow \{d \in \mathbb{D}_s | creationTime(d) < p\}$
      $\mathbb{D}^i_{tst} \leftarrow \{d \in \mathbb{D}_s | creationTime(d) = p\}$

      Strategy 1: Traditional SVM Classifier
      SVM ($\mathbb{D}^i_{trn}$,$\mathbb{D}^i_{tst}$)

      Strategy 2: Adaptive Window SVM Classifier
      TUNEWINDOWBASEDSVM ($\mathbb{D}^i_{trn}$)    ▷ Find optimized
        window size.
      WINDOWBASEDSVM ($\mathbb{D}^i_{trn}$,$\mathbb{D}^i_{tst}$)

      Strategy 3: FilterChronos Classifier
      TUNEFILTERCHRONOS($\mathbb{D}^i_{trn}$)    ▷ Find number of top-K
        timepoints.
      FILTERCHRONOS ($\mathbb{D}^i_{trn}$,$\mathbb{D}^i_{tst}$)
    end for
  end for
end function
```

As we can observe in Fig. 13, the "drift-aware" strategies Adaptive Window and *FilterChronos* outperformed the traditional SVM classifier in all but one case (the exception being the first batch 1995, for ACM-DL). Furthermore, we see that the *FilterChronos* strategy is not worse than the Adaptive Window approach, with the former outperforming the latter in two batches (namely, 1998 and 2001), considering the ACM collection. Considering the MedLine collection, we can observe that the *FilterChronos* was not worse than the Adaptive Window strategy, being superior in two batches (namely, 1981 and 1982). It is interesting to note that in this specific collection, the traditional SVM classifier was statistically tied with our approach, in all batches. This is due to the more stable behavior of the MedLine collection, as characterized in Section 3. In fact, the SVM classifier was able to deal with such a mild data variation. On the other hand, considering batches 1981 and 1982, the Adaptive Window approach degraded classification effectiveness. We attribute this to a sub-optimal sampling of data: the chosen window size discarded useful stable information (laying outside the window), which was captured by the temporal contexts. Moreover, notice that, despite the fact that the *FilterChronos* was statistically tied with the traditional SVM classifier, the training set size was substantially reduced, which contributes to a better efficiency. This highlights the quality of the learned temporal contexts which are comparable to the optimized adaptive

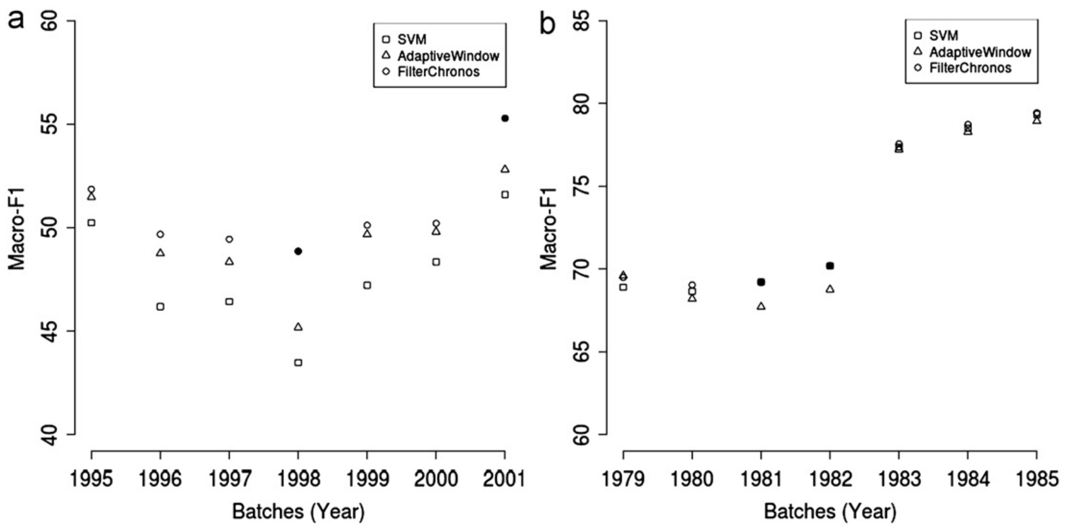

Fig. 13. MacroF1 results achieved for each batch (year). The filled circles • represent the cases where *FilterChronos* achieved statistically significant gains over the Adaptive Window strategy, with 95% confidence. (a) ACM collection. (B) MedLine collection.

windows, or even superior. We stress here that, despite not being the main application of the proposed *FilterChronos*, it can still be effectively applied to drifting scenarios, due to its general formulation.

7. Conclusion and future work

In this paper we provided evidence that time is indeed an important factor and that must be considered in classification algorithms. First we demonstrated that the temporal evolution may negatively affect the performance of text classification models, since models based on the whole training data may not perform at their full potential. Accounting for this evolution is key for an effective classification. We distinguished three different temporal effects (class distribution, terms distribution and class similarity) that may affect the performance of automatic classifiers. Moreover, we proposed and applied a novel methodology for assessing the impact of the temporal evolution on the performance.

In the second part of this work, we proposed a temporal context selection strategy for building classification models. Our strategy consisted of selecting contexts, a set of pre-classified documents that minimizes the three aforementioned temporal effects, so that the resulting learned classifiers are less susceptible to temporal related changes. Then we proposed *Chronos*, an algorithm that may be used as a template by techniques that aim to generate temporal contexts.

In the third part of this work, we proposed and evaluated two computationally feasible heuristics for temporal context selection. We evaluated them using four ADC algorithms (Naïve Bayes, kNN, Rocchio and SVM), using two real collections. The results showed that the *WindowChronos* heuristic works well with algorithms that are based on asymmetric premises, i.e., algorithms that use only the test terms in order to determine the correct class of the documents, but may not work properly with algorithms that are based on symmetric premises. Despite the fact that the confusion regarding the ambiguity of test terms across classes may be reduced in temporal contexts selected by the *WindowChronos* heuristic, the same cannot be guaranteed for the other terms that are included in the temporal contexts. Consequently, the (symmetric) premises of the algorithms that consider all terms may be affected. However, using the temporal contexts selected by the *FilterChronos* heuristic, an evolution of the *WindowChronos* on which these premises are preserved, all algorithms presented improvements.

Finally, we demonstrated the applicability and the generality of the discussed temporal contexts in practice, pointing out this study as a promising research direction on distinct real domains. More specifically, we evaluated the use of the *FilterChronos* technique in a Concept Drift scenario, on which we compared it with a traditional approach, the Adaptive Time Window [19]. The results showed that, despite not being the main application of the proposed *FilterChronos*, the quality of the learned temporal contexts selected by it are comparable to the optimized adaptive windows, or even superior.

As future work we propose to apply the methodology to different multilabel Web collections in order to investigate how the temporal evolution affects them. Moreover, we intend to apply both techniques presented in this paper on these Web collections, using different ADC algorithms in order to show the potential of temporal contexts in several scenarios. We also want to extend our techniques to consider other types of evidence, such as inlinks, outlinks and compound-feature [13] within a temporal context.

Acknowledgments

This work is partially supported by InWeb, CNPq, CAPES, FINEP and FAPEMIG.

References

[1] A. Hwee Tan, Text mining: the state of the art and the challenges, in: Proceedings of the PAKDD, 1999, pp. 65–70.

[2] H. Borko, M. Bernick, Automatic document classification, Journal of the ACM 10 (2) (1963) 151–162.

[3] R. Feldman, I. Dagan, Knowledge discovery in textual databases (KDT), in: Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), 1995, pp. 112–117.

[4] E.A. Fox, R.M. Akscyn, R.K. Furuta, J.J. Leggett, Digital libraries, Communications of the ACM 38 (4) (1995) 22–28.

[5] C. Chekuri, M. Goldwasser, P. Raghavan, E. Upfal, Web search using automatic classification, in: Proceedings of the Sixth International Conference on the World Wide Web, Citeseer, 1997.

[6] O. Alonso, M. Gertz, R. Baeza-Yates, On the value of temporal information in information retrieval, SIGIR Forum 41 (2) (2007) 35–41.

[7] F. Mourao, L. Rocha, R. Araújo, T. Couto, M. Gonçalves, J. Wagner Meira, Understanding temporal aspects in document classification, in: Proceedings of ACM WSDM Conference, ACM, New York, NY, USA, 2008, pp. 159–170.

[8] W.W. Cohen, Y. Singer, Context-sensitive learning methods for text categorization, ACM Transactions on Information Systems 17 (2) (1999) 141–173.

[9] S. Lawrence, C.L. Giles, Context and page analysis for improved web search, IEEE Internet Computing 2 (4) (1998) 38–46.

[10] P. Turney, The identification of context-sensitive features: a formal definition of context for concept learning, in: Proceedings of the Workshop on Learning in Context-Sensitive Domains, 1996, pp. 53–59.

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

[12] Y.S. Kim, S.S. Park, E. Deards, B.H. Kang, Adaptive web document classification with MCRDR, in: Proceedings of ITCC Conference, vol. 2, IEEE Computer Society, Washington, DC, USA, 2004, p. 476.

[13] F. Figueiredo, L. Rocha, T. Couto, T. Salles, M.A. Gonçalves, W. Meira Jr., Word co-occurrence features for text classification, Information Systems 36 (5) (2011) 843–858.

[14] A. Tsymbal, The Problem of Concept Drift: Definitions and Related Work, Technical Report, Department of Computer Science, Trinity College, Dublin, Ireland, 2004.

[15] H. Wang, P.S. Yu, J. Han, Mining concept-drifting data streams, in: Data Mining and Knowledge Discovery Handbook, 2010, pp. 789–802.

[16] Y. Koren, Collaborative filtering with temporal dynamics, Communications of the ACM 53 (2010) 89–97.

[17] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: P. Langley (Ed.), Proceedings of ICML Conference, Morgan Kaufmann Publishers, San Francisco, US, Stanford, US, 2000, pp. 487–494.

[18] I. Žliobaitė, Combining time and space similarity for small size learning under concept drift, in: ISMIS, Prague, Czech Republic, 2009, pp. 412–421.

[19] R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Journal of Intelligent Data Analysis 8 (3) (2004) 281–300.

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

[21] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intelligent Data Analysis 11 (1) (2007) 3–28.

[22] G. Folino, C. Pizzuti, G. Spezzano, An adaptive distributed ensemble approach to mine concept-drifting data streams, in: Proceedings of the 19th ICTAI, Washington, DC, USA, 2007, pp. 183–188.

[23] T. Salles, L. Rocha, G.L. Pappa, F. Mourão, M.A. Gonçalves, W. Meira Jr., Temporally-aware algorithms for document classification, in: Proceedings of ACM SIGIR Conference, ACM Press, Genebra, Switzerland, 2010, pp. 307–314.

[24] L. Rocha, F. Mourao, A. Pereira, M. Gonçalves, W. Meira, Exploiting temporal contexts in text classification, in: Proceedings of ACM CIKM Conference, ACM Press, Napa Valley, CA, USA, 2008, pp. 243–252.

[25] M.M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, Integrating novel class detection with classification for concept-drifting data streams, in: Proceedings of ECML PKDD Part II, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 79–94.

[26] M.M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han, B. Thuraisingham, Addressing concept-evolution in concept-drifting data streams, in: Proceedings of IEEE ICDM Conference, IEEE Computer Society, Washington, DC, USA, 2010, pp. 929–934.

[27] M.M. Masud, T.M. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, B. Thuraisingham, Detecting recurring and novel classes in concept-drifting data streams, in: Proceedings of IEEE ICDM Conference, 2011, pp. 1176–1181.

[28] C. Wang, D.M. Blei, D. Heckerman, Continuous time dynamic topic models, in: Proc. of UAI Conference, Helsinki, Finland, 2008, AUAI Press, pp. 579–586.

[29] J. Boyd-Graber, D. Blei, Multilingual topic models for unaligned text, in: Proceedings of UAI Conference, Corvallis, Oregon, 2009, pp. 75–82.

[30] D. Mimno, D. Blei, Bayesian checking for topic models, in: Proceedings of EMNLP Conference, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011, pp. 227–237.

[31] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359.

[32] G. Forman, Tackling concept drift by temporal inductive transfer, Washington, USA, 2006, pp. 252–259.

[33] P. Zhao, S.C.H. Hoi, OTL: a framework of online transfer learning, in: Proceedings of ICML Conference, 2010, pp. 1231–1238.

[34] K. Crammer, O. Dekel, J. Keshet, S. Shalev-shwartz, Y. Singer, Online passive–aggressive algorithms, Journal of Machine Learning Research 7 (2006) 551–585.

[35] T. Joachims, Making large-scale support vector machine learning practical, in: A.S.B. Schölkopf, C. Burges (Eds.), Advances in Kernel Methods, MIT Press, Cambridge, MA, 1998, pp. 169–184.

[36] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, H.-W. Hon, Adapting ranking SVM to document retrieval, in: Proceedings of the 29th ACM SIGIR Conference, ACM, New York, NY, USA, 2006, pp. 186–193.

[37] G. Forman, An extensive empirical study of feature selection metrics for text classification, Journal of Machine Learning Research 3 (2003) 1289–1305.

[38] L. Molina, L.C, Belanche, A. Nebot, Feature selection algorithms: a survey and experimental evaluation, in: Proceedings of the IEEE ICDM Conference, 2002, pp. 306–313.

[39] Y. Yang, Expert network: effective and efficient learning from human decisions in text categorization and retrieval, in: Proceedings of the 17th ACM SIGIR Conference, ACM Press, 1994, pp. 13–22.

[40] D.P. Faith, Asymmetric binary similarity measures, Oecologia (Berlin) 57 (1983) 287–290.

[41] A.K. McCallum, Bow: a toolkit for statistical language modeling, text retrieval, classification and clustering ⟨http://www.cs.cmu.edu/~mccallum/bow/⟩, 1996.

[42] T. Joachims, Training linear SVMs in linear time, in: Proceedings of the 12th ACM SIGKDD Conference, ACM Press, 2006, pp. 217–226.

[43] V. Vapnik, Statistical learning theory, 1998.

[44] Y.F. Zheng, One-against-all multi-class SVM classification using reliability measures, in: Proceedings 2005 IEEE International Joint Conference on Neural Networks, vol. 2, 2005, pp. 849–854.

[45] D.D. Lewis, Evaluating and optimizing autonomous text classification systems, in: Proceedings of ACM SIGIR Conference, Seattle, Washington, 1995, pp. 246–254.

[46] L. Brieman, P. Spector, Submodel selection and evaluation in regression: the x-random case, International Statistical Review 60 (1992) 291–319.

[47] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.

[48] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[49] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval* 1, Information Processing & Management 24 (5) (1988) 513–523.

[50] T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, Springer Verlag, 1998, pp. 137–142.

[51] R. Groeneveld, An influence function approach to describing the skewness of a distribution, in: The American Statistician, vol. 45, 1991, pp. 97–102.