it using two distinct collections: ACM-DL and MedLine. We in
order to demonstrate the effectiveness of our strategy, we e
heuristic that is able to determine temporal contexts efficien
tially evaluated whether these observed reductions translate int
better classifier and the entropy associated with each context. Fur
further, it is shown that using just portions of the pre-classified
documents, which we call contexts, for building the classifiers re
result in better performance, as a consequence of the minimization of
the aforementioned effects.

In this paper we define the concept of temporal contexts as be
ning the portions of documents that minimize the aforementioned
effects. We then propose a general algorithm for determining such
contexts, discuss its implementation-related issues, and propose a
heuristic that is able to determine temporal contexts efficiently. In
order to demonstrate the effectiveness of our strategy, we evaluated
it using two distinct collections: ACM-DL and MedLine. We ini
tially evaluated the reduction in terms of both the effort to build a
classifier and the entropy associated with each context. Further, we
evaluated whether these observed reductions translate into better
classification performance by employing a very simple classifier,
majority voting. The results show that we achieved precision gains
of up to 30% compared to a version that is not temporally contextu
alized, and the same accuracy of a state-of-the-art classifier (SVM),
while presenting an execution time up to hundreds times faster.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Information Search and Retrieval; I.5.4 [Applications]: Text processing;

General Terms
Algorithms, Classification, Experimentation

Permission to make digital or hard copies of all or part of this work for
personal or classroom use is granted without fee provided that copies are
not made or distributed for profit or commercial advantage and that copies
bear this notice and the full citation on the first page. To copy otherwise, to
reproduce, to post on servers or to redistribute to lists, requires prior specific
permission and/or a fee.

Permission to make digital or hard copies of all or part of this work for
personal or classroom use is granted without fee provided that copies are
not made or distributed for profit or commercial advantage and that copies
bear this notice and the full citation on the first page. To copy otherwise, to
reproduce, to post on servers or to redistribute to lists, requires prior specific
permission and/or a fee.

Copyright 2008 ACM 978-1-59593-927-9/08/0002 ...$5.00.

Exploiting Temporal Context in Text Classification

Leonardo Rocha, Fernando Mourão, Adriano Pereira
Marcos André Gonçalves, and Wagner Meira Jr.
Federal University of Minas Gerais
Computer Science Department
Av. Antônio Carlos 6627 - ICEx - 31270-010
Belo Horizonte, Brazil
{lrcrocha, fhmourao, adrianoc, mgoncalv, meira}@dcc.ufmg.br

ABSTRACT
Due to the increasing amount of information being stored and ac
cessible through the Web, Automatic Document Classification (ADC)
has become an important research topic. ADC usually employs a
supervised learning strategy, where we first build a classification
model using pre-classified documents and then use it to classify
unseen documents. One major challenge in building classifiers is
dealing with the temporal evolution of the characteristics of the
documents and the classes to which they belong. However, most of
the current techniques for ADC do not consider this evolution while
building and using the models. Previous results show that the per
performance of classifiers may be affected by three different temporal
effects (class distribution, term distribution and class similarity).
Further, it is shown that using just portions of the pre-classified
documents, which we call contexts, for building the classifiers re
result in better performance, as a consequence of the minimization of
the aforementioned effects.

In this paper we define the concept of temporal contexts as be
ning the portions of documents that minimize the aforementioned
effects. We then propose a general algorithm for determining such
contexts, discuss its implementation-related issues, and propose a
heuristic that is able to determine temporal contexts efficiently. In
order to demonstrate the effectiveness of our strategy, we evaluated
it using two distinct collections: ACM-DL and MedLine. We ini
tially evaluated the reduction in terms of both the effort to build a
classifier and the entropy associated with each context. Further, we
evaluated whether these observed reductions translate into better
classification performance by employing a very simple classifier,
majority voting. The results show that we achieved precision gains
of up to 30% compared to a version that is not temporally contextu
alized, and the same accuracy of a state-of-the-art classifier (SVM),
while presenting an execution time up to hundreds times faster.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Information Search and Retrieval; I.5.4 [Applications]: Text processing;

General Terms
Algorithms, Classification, Experimentation

Keywords
Text Classification, Temporal Analysis, Digital Libraries

1. INTRODUCTION
The widespread use of the Internet has increased the amount of
information being stored and accessed through the Web. This infor
mation is frequently organized as textual documents 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 meaning
ful categories, a technique known as Automatic Document Classi
fication (ADC) [4]. The class assignment may support and enhance
several tasks such as automated topic tagging (i.e., assigning labels
to documents), building topic directories, identifying documents
writing style, creating digital libraries, improving the precision of
Web searching, or even helping users to interact with search en
gines. Other useful application scenarios include spam filtering,
detection of adult content and plagiarism.

ADC usually employs a supervised learning strategy, where we
first build a classification model using pre-classified documents, i.e.
a training set, and then use the model to classify unseen documents.
Building text classification models usually consists of finding and
weighting a set of characteristics (e.g., terms) that better identify
classes of documents. One major challenge in building classifiers is
dealing with the temporal evolution of the characteristics of the
documents and the classes to which they belong, as a consequence
of the creation of new documents, the introduction of new terms,
the rise of new fields, and the division of large fields into more
specialized sub-fields. Thus, the differences, in terms of document
characteristics, that once were useful for distinguishing classes may
either increase or reduce, affecting the precision of the classifier
that exploits them.

In [1], Baeza-Yates et al. advocate that time is an important di
mension of any information space and may be very useful in infor
mation retrieval. Despite the potential impact of the temporal
 evolution on the quality of classification models and its recogni
tion as a relevant factor, most of the current techniques for ADC,
such as nearest-neighbor, Bayesian, support vector machines, and
association-based classification [30] do not consider this evolution
while building and using the models. It is also not clear which tem
poral characteristics affect the quality of the model. In our recent
work [24] we distinguish three different temporal effects that may
affect the performance of automatic classifiers. The first effect,
called class distribution, is related to the impact of the temporal
evolution on the class frequency and how it changes across time.
Classes may appear and disappear, which may happen as a con
sequence of splits and joins, respectively, of existing classes. The
second effect, called term distribution, is how the relationship be-
between terms and classes changes over time, as a consequence of terms appearing, disappearing, and having variable discriminative power across classes. The third effect, class similarity, is how the similarity among classes, as a function of the terms that occur in their documents, varies over time. For instance, two classes may be similar at a given moment, and not similar later in the future. We demonstrated the existence of these three factors and how each of them affects the classifier’s performance.

After understanding the temporal effects in ADC, we need to build classification models that are effective regardless of these effects. In order to do this, we should be able to both classify an unseen document in context, that is, at the moment of its creation, and consider just characteristics that are stable in that context. One strategy is to create classification models that deal with the aforementioned temporal effects. A different strategy, which is the focus of this paper, is to select a portion of the training set that minimizes those effects, which we call temporal context selection. We propose a general algorithm for temporal context selection and derive a heuristic that feasibly instantiates the general algorithm. In order to demonstrate the effectiveness of our strategy, we applied it to two distinct collections derived from the digital library of the ACM (ACM-DL), containing documents about Computer Science, and MedLine, a digital library related to Medicine. We used the resulting selected documents for building a very simple majority voting classifier. This simple classifier was able to achieve gains of up to 30% against a version that did not consider time and the same accuracy of a state-of-the-art classifier (SVM), while presenting an execution time up to hundreds times faster.

The remainder of this paper is organized as follows. Section 2 briefly describes the characterization of the temporal effects presented in [24], necessary to understand the remainder of the paper. Section 3 formally defines the problem that we deal in this paper, presents a general algorithm for temporal context selection and some possible solutions to implement it. In Section 4, we present the technique we design for this problem. In Section 5, we evaluate this technique. Section 6 discusses the related work and, finally, in Section 7 we conclude and discuss future work.

2. CHARACTERIZING TIME EFFECTS

Before we discuss how we can handle the effects of the temporal evolution of the collections, it is necessary to understand and quantify how this evolution manifests, which is the goal of this section. We distinguish and quantify three temporal effects that contribute to that evolution: class distribution, terms distribution, and class similarity. We assess those effects in the ACM digital library and MedLine collections (more details about these collections are given in Section 5) when building a classifier based on Support Vector Machines (SVM) [13]. We also discuss how the size of the sample affects the precision of the classifiers (the sampling effect), and the trade-off between the size of the sample and how much the classifiers are affected by the temporal effects. We have published these results in [24] and summarized them in this section for a better comprehension of the remainder of this paper.

In order to demonstrate the sampling effect, a document sub-collection was built for each collection, containing documents from just one year, in order to isolate this effect from the temporal effects. Using these sub-collections, several classification models were built, varying the size of the training set from 20% to 100% of the sub-collection. The result of this analysis is presented in Figure 1, where we can see that as we increase the size of the training set, the performance of the classifier gets better, as expected.

For the experiments that assess the temporal effects, each collection was divided into several sub-collections in a per year basis, each containing the same number of documents per collection (441 for ACM-DL\(^1\) and 34,755 for MedLine\(^2\)). Then, for each sub-collection, a classification model is built using it as training set with a cross-validation strategy and the resulting model was used to classify the documents of the other sub-collections from the same original collection. Figure 2 summarizes the experiments for both ACM and MedLine showing the relative accuracy (y axis) for the various time distances (x axis), which represents the difference, in years, between the test and the training sub-collections. Notice that, in most cases, the best scenario is found when the training and the test documents belong to the same year (interval zero).

![Figure 2: Temporal Locality Variation](image)

As we can observe in these results, there is a challenge in dealing with the sampling and temporal effects simultaneously, because they indicate opposite strategies in terms of setting a training set. Increasing the size of the training set may introduce documents that are out of the temporal context, which would reduce the precision of the classifier. On the other hand, reducing the training set to just the documents close to the test document would ease the construction of a classifier, but there is a risk of not having enough data to build the model or the resulting model being overfitted. In order to determine the proper temporal context, that is, a portion of the pre-classified documents that is as large as possible and is temporally consistent, we need to understand and quantify the three aforementioned temporal effects.

First we analyze the variation of the class distribution over time. Figure 3, where the class probability distribution was plot for each year, illustrates the variation in terms of the representativeness of the classes across time. As can be seen, the oscillation of the occurrence of the classes is frequent across the years. For example, the class Math becomes less frequent as time progresses. Similar results can be observed in the MedLine collection. These results

---

\(^1\)The year with the smallest number of documents is 1980 (441).
\(^2\)The minimum number of documents per year is 34,755 in 1970.
show that our temporal contexts should capture the existing characteristics when the document to be classified was created, otherwise the model may not be accurate.

Figure 3: Class Occurrence Variation Through Time - ACM

In order to characterize the term distribution effect, for each class and each year, a vocabulary is defined as the words that have the highest values of info-gain [11] to identify the class. The vocabularies for the same class in different years are compared using the cosine similarity [27] between them. Figure 4 shows the cosine similarity as we vary time distance. Distance zero means we are comparing a vocabulary to itself, which obviously corresponds to the maximum similarity. We can observe that the larger the time distance between the vocabularies, the less similar they are, which demonstrates that vocabularies change over time. The cosine similarity between the vocabularies for the same class with time distance 20 is less than 50% for ACM-DL (for all classes) and less than 80% for MedLine (for almost all classes). An interesting example is the class Aids of Medline collection. Different from the other classes, the difference in similarity among its vocabulary vectors declines very fast, showing the class Aids is very dynamic.

Figure 4: Terms Distribution Means

Finally, to verify the effect of the evolution on the class similarity, we calculate the cosine similarity between the vocabularies for each pair of distinct classes for each year. Tables 1 and 2 show the variation of similarity between each pair of classes over time for MedLine and ACM-DL, respectively. Each entry in these tables is the standard deviation from the mean of the similarities between the associated pairs of classes in all years. As we can observe, for some pairs of classes the similarity variation is very high, such as for classes Complementary Medicine (CM) and History (Hist). It means these two classes may have been very similar in some periods, but also loosely related in others. Consequently, the difficulty in separating these two classes varies significantly across time.

Table 1: Similarity Std_Dev Matrix - MedLine

<table>
<thead>
<tr>
<th></th>
<th>Aids</th>
<th>Bio</th>
<th>Cancer</th>
<th>C, M</th>
<th>Hist</th>
<th>Space</th>
<th>Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aids</td>
<td>0</td>
<td>0.19</td>
<td>0.16</td>
<td>0.16</td>
<td>0.18</td>
<td>0.18</td>
<td>0.19</td>
</tr>
<tr>
<td>Bio</td>
<td>0</td>
<td>0.04</td>
<td>0.04</td>
<td>0.20</td>
<td>0.17</td>
<td>0.19</td>
<td>0.12</td>
</tr>
<tr>
<td>Cancer</td>
<td>0</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.08</td>
<td>0.08</td>
<td>0.05</td>
</tr>
<tr>
<td>C, M</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0.21</td>
<td>0.08</td>
<td>0.11</td>
<td>0.05</td>
</tr>
<tr>
<td>Hist</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0.30</td>
<td>0.11</td>
<td>0.05</td>
</tr>
</tbody>
</table>

3. TEMPORAL CONTEXT SELECTION

In this section we formalize the problem addressed in this work. Moreover, we present a general algorithm to solve this problem and discuss some possible implementations based on it.

3.1 Problem Definition

Before we formalize 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; \( T = \{t_1, t_2, \ldots, t_N\} \) be the set of terms associated with the training set of a collection; \( M = \{m_1, m_2, \ldots, m_P\} \) be the set of moments on a temporal space when both test and training documents are created (timestamp). We define a training document \( d_i \) by the triple \( x, m_p, c_i \), where \( x \subseteq T, m_p \in M \) and \( c_i \in C \). Given a set \( S = \{s_1, s_2, \ldots, s_{V}\} \) comprising the documents that we want to classify and \( E = \{e_1, e_2, \ldots, e_{Z}\} \) be the set of terms associated with \( S \), the task of ADC is to determine the class \( c_i \) associated with the test document \( s_i \), where each \( s_i = \{y, m_p\}, y \subseteq E, m_p \in M \).

ADC usually follows a supervised learning strategy, where we first need to select a context \( X \subseteq D \) that is used to build a classification model. Often \( X \) is equal to \( D \) or a random sample of \( D \). Then we use this model to classify new unseen documents \( S \).

As expected, the context \( X \) is composed by documents which were created at different time moments \( (M) \). As previously discussed in the last section, the sampling effect suggests that we should increase the size of \( X \), while the time effects induce the reverse, since using long time periods may generate conflicting evidence that would harden the task of building the classifier. Therefore, the problem we address in this paper is to determine contexts that optimize the trade-off between the sampling effect and the time effects. Notice that the number of possible contexts grows exponentially with the number of documents, thus demanding strategies that narrow the search to better contexts efficiently.

It is important to distinguish the context selection problem from feature selection [23], which are used to reduce the number of features that will be used in the classification model, usually for efficiency purposes, but maintaining its accuracy and robustness. Although the reduction of the size of the training set is a side-effect, the main goal of temporal context selection is to determine largest portion of the training set in which the time effects (class distribution, class similarity and terms distribution), and thus the uncertainty that a classifier has to deal with, are minimized.

In order to illustrate our problem, consider the training set presented in Table 3. 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 middle of 2006. Thus, as
we can observe in Table 3, before 2005 our training set consists of documents from only the class Astrophysics. After 2005, it contains only documents from class Mythology. While in 2005 there is one document from each class. If we just employ the whole training set for building a classifier, we would depend on other features to determine the correct category for a document, since the term Pluto could confuse the classifier. However, considering the temporal information for delimiting contexts where Pluto has a single meaning could solve the problem.

<table>
<thead>
<tr>
<th>Word: Pluto</th>
<th>Class: Astrophysics</th>
<th>Class: Mythology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc_Id</td>
<td>Year</td>
<td>Doc_Id</td>
</tr>
<tr>
<td>d1</td>
<td>1996</td>
<td>d5</td>
</tr>
<tr>
<td>d2</td>
<td>1998</td>
<td>d6</td>
</tr>
<tr>
<td>d3</td>
<td>2000</td>
<td>d7</td>
</tr>
<tr>
<td>d4</td>
<td>2002</td>
<td>d8</td>
</tr>
<tr>
<td>d5</td>
<td>2004</td>
<td>d9</td>
</tr>
</tbody>
</table>

Table 3: Example

3.2 The Chronos Algorithm

In this section we propose an algorithm for temporal context selection that addresses three fundamental requirements for the contexts. This algorithm can be used as a template for implementations that select contexts from the training set \( X \) in which the temporal effects are minimized. Based on the characterization of time effects presented in [24] and briefly discussed in Section 2, we identify three requirements that should be fulfilled by a good temporal context:

1. **Reference constrained**: as observed in Section 2, the best scenario for document classification is found when the training and test documents belong to the same moment in time. Therefore, the context \( X \) needs to be temporally constrained to a reference, in our case the document to be classified, in order to capture the temporal characteristics of the training set (class distribution, terms distribution and class similarity) associated with the moment when the test document was created. For example, if the test document is new, the reference constrained will be the current time and the temporal characteristics of the training set will be captured according to the present.

2. **Characteristic stability**: as observed in the previous section, as distance increases, the characteristics of documents tend to vary and, consequently, the characteristics of the training set also change. Thus, the selection of context \( X \) needs to be stable w.r.t. 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. 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 restricts the time distance among the documents of \( X \) and is related to temporal effects that suggest that long time periods may generate conflicting evidence.

3. **Uncertainty reduction**: as explained in Section 2, 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 tends make the relationship between terms and classes very uncertain and, consequently, the whole training set \( D \) very confusing, degrading the performance of classifiers. Therefore, the context \( X \) that we are looking for presents both a smaller uncertainty and higher information gain than the whole training set. Moreover, the contexts should have enough characteristics to identify and discriminate the classes. This requirement is related to the sampling effect.

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

**Algorithm 1 Chronos Algorithm**

1. \textbf{function} Chronos(\( D, S, \delta \))
2. \textbf{min} \leftarrow \infty
3. \( X \leftarrow \emptyset \)
4. \textbf{state} \leftarrow GetReference(\( D, S \))
5. \textbf{option} \leftarrow Enumerate(\( D \))
6. \textbf{repeat}
7. \textbf{if} (\textbf{StabilityFunction}(\textbf{option}, \textbf{state}) < \delta) \textbf{then}
8. \textbf{uncertainty} \leftarrow TemporalUncertainty(\textbf{option})
9. \textbf{if} (\textbf{uncertainty} < \textbf{min}) \textbf{then}
10. \( X \leftarrow \textbf{option} \)
11. \textbf{min} \leftarrow \textbf{uncertainty}
12. \textbf{option} \leftarrow Enumerate(\( D \))
13. \textbf{until} (\textbf{option} \neq \emptyset)
14. \textbf{return} (\( X \))

Chronos receives as input two document collections \( D \) (training) and \( S \) (test), and a stability threshold \( \delta \). We address the first requirement.
requirement in line 4. The function GetReference(D, S) captures the temporal characteristics (class distribution, terms distribution and class similarity) of training set D at the moment when the test document S was created. 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 the characteristics of the training set D at the moment when the test document S was created. This difference needs to 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. Then, among the possible contexts X that are stable, the algorithm selects the one with the smallest uncertainty.

In the next section we discuss some implementation strategies that stem from the general algorithm presented in this section.

### 3.3 Implementation Strategies

In this section we discuss and analyze some implementation strategies to temporal context selection, based on Algorithm 1. One possible strategy is to perform the selection of X only once for the whole test set and use it for classifying all test documents. To illustrate this solution, we can take the example presented in Table 3. Considering the test set S composed by documents s1, s2, s3, s4 from 1999, 2006, 2001, and 2005 respectively, select a portion of the training set that considers the three requirements presented in last section is not possible, since our test set consists of documents from different moments. How can we contextualize temporally the training set with the date of test documents if each test document is from a different moment? The function GetReference of Algorithm 1 will capture the temporal characteristics of different moments and the uncertainty will not be reduced, emphasizing the need for a on-demand solution.

In [24] we presented an experiment that illustrates the problem related to this first strategy for two real collections: ACM-DL and MedLine. Their experiment consists of using a time-sensitive selection of the documents for training. That is, they select as their training set, for each test document, just the documents that are closer in time to it. The proximity is defined by a time-window that may grow symmetrically in both directions, past and future, based on the year of the test document. They varied the size of this window from 0 (that is, the year Ai of the tested document belongs to) to N, where N represents the number of years before and after Ai. The value of N is defined as the value that maximizes the performance of the classifier to the test documents that belong to the year Ai. Figure 5 presents the optimum window size for test documents from all years in the ACM collection (a) and the MedLine collection (b) using a SVM classifier. For instance, in the ACM collection, for the documents from 1990, the optimum size window is 10 years. For the documents from 1989, however, the optimum size is 20 years. As we can see, there is not a single optimum size for the window for the entire database, in both collections, but an optimum window size may be found for each specific year. Thus, a unique selection for all test documents is not effective.

Another strategy is to perform an exhaustive search for the best context X for each test document. In order to do this, it is necessary that the function Enumerate(D) lists all possible contexts X from set D. Then, for each test document Sx, according to its temporal characteristics, the strategy chooses the best context X. Since we need to exploit characteristics of the test document, we can say that this alternative is clearly a lazy solution [34].

Lazy algorithms are based on the assumption that there is not a universal classification model and they sample the training set at classification time. However, similarly to other lazy solutions, the implementation must be computationally feasible. Considering that there are 2i possible X contexts, where D is the number of training documents, a brute-force solution is not feasible, since it requires the evaluation of each of these contexts in order to find the best one for each test document. In our example, the function Enumerate(D) would list 210 possible sets and evaluate whether each of them is acceptable. However, when we deal with a real collection like ACM-DL, that contains 30,000 documents, to evaluate 230,000 contexts is not usually feasible. One strategy to reduce the computational costs is to employ heuristics that address the aforementioned requirements while keeping the computational costs low. In the next section we propose such heuristic.

### 4. GREEDYCHRONOS

In this section we present GreedyChronos, a heuristic for selecting temporal contexts based on the general algorithm Chronos. The strategy of GreedyChronos is to select a context from the training set where the three time effects are minimized, enabling more effective classification models.

GreedyChronos 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 (characteristics 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 on any class for a determined period, and is quantified by Dominance [36]. Formally, let \( T = \{t_1, t_2, t_3, \ldots, t_M\} \) be the set of terms associated with a collection; \( C = \{c_1, c_2, \ldots, c_K\} \) be the set of categories that occur in a collection; \( df(t_i, c_j) \) be the number of training documents associated with class \( c_j \) that contain \( t_i \). We define the Dominance of the class \( c_j \) on term \( t_i \) as:

\[
\text{Dominance}(t_i, c_j) = \frac{df(t_i, c_j)}{\sum_{l \neq j} df(t_i, c_l)} \quad (1)
\]

In our approach, the time-window of each test term is contiguous for a couple of reasons. First, because the relationships between terms and classes tend to change smoothly as we increase the distance between the test document and a document in the context, as we could see in Section 2. Second, because the search space for determining a window that not is contiguous is \( 2^{ij} \), while the search space for determining a contiguous window is \( D^j \). Therefore, the dominance quantifies both the stability and the reduction in terms...
of uncertainty associated with a window, since the stronger is the relationship between a term and a class, the smaller is the uncertainty.

Let’s come back to the example presented in Table 3 and the test set S, consisting of documents s₁, s₂, s₃, s₄ from 1999, 2006, 2001, and 2005, on 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 on each test document. For example, the document s₁ is from 1999 and in this year there is only one document in training set that belongs to class Astrophysics. Thus, in this year, the term Pluto has a dominance equal to 100% and this year will compose the time window of term Pluto for document s₁. Then, the algorithm analyzes the training set in neighbor years. A similar behavior can be observed in years 1998, 2000, 2001, 2002, 2003, 2004 and 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 s₁ and the algorithm terminates the search for the time window of term Pluto for document s₁. The algorithm is executed for all terms of each test document. The time window for the word Pluto on each test document from test set S would be: s₁ (1998 to 2001); s₂ (2006 to 2007); s₃ (1998 to 2001); s₄ (empty).

After determining the time window for each term, the last step is to select the documents that will be in the context and be used as 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 s₁ and s₃ would be the documents d₁, d₂, d₃, d₄, d₅, d₆, d₁₀, d₁₁, d₁₂, d₁₃, d₁₄, d₁₅, d₁₆. For the test document s₄ there are no documents, what may be addressed using other terms in the document. It should be noticed that our heuristic can be easily adapted for situations where we only have past information, such as Adaptive Document Classification [8] and Concept Drift [28].

5. EVALUATING GREEDYCHRONOS

In this section we describe the results of experiments conducted for evaluating the GreedyChronos heuristic proposed in this paper. This evaluation shows how the heuristic can reduce the time effects in each training set and demonstrates that it enables us to find a set of documents that effectively reduces the time effects.

5.1 Collections

In all experiments we use two collections. The first one is a set of 30,000 documents from the ACM-DL containing articles from 1980 to 2002. In this collection, classes of the ACM taxonomy are assigned to documents by their own authors. We used the first level of the taxonomy, comprising all 11 categories, which is a collection much harder to classify than ACM8, used in works such as [32], in which the 3 least frequent classes are excluded. All documents are assigned to a single class. The second collection is derived from MedLine and consists of 861,454 documents, containing articles from 1970 to 1985, classified into 7 distinct classes. Note that this collection is significantly larger (in our case almost 2 orders of magnitude) and more diverse than OHSUMED⁴, which is a medical collection commonly used for evaluating document classifiers [7]. It is important to mention that these collections present different characteristics. While ACM-DL presents a very imbalanced class frequency (some classes are much more frequent than others), in MedLine this imbalance is not so high. Moreover, the amount of documents in MedLine is larger than in ACM-DL collection.

5.2 Experiments

We can see time as a discretization of natural changes inherent to any knowledge area, although detectable changes may occur at different time scales, depending on the area characteristics. In the case of our experimental collections, which consist of sets of scientific articles, we adopted yearly intervals for identifying such changes (e.g., conferences, which are usually annual), but in other scenarios the granularity may be distinct.

We performed a set of experiments that consist of determining the temporal context (and then the training set) associated with each test document, and varied the dominance parameter from 0% (no temporal information is considered) to 90% (we will not present the results for dominance greater than 90%, since the resulting contexts are very sparse, most of than are empty). In all experiments we employed a 10-fold cross-validation [5] and the final results of each experiment is the average of the ten runs.

We start our evaluation by analyzing the reduction in terms of the size of the contexts selected as a function of the dominance. Our metric in this case is called total terms, which is the sum of the number of terms in the contexts selected for all test documents, quantifying the effort of a classifier to build a model for each test document. The results for 0% dominance are used as a baseline and all other results are relativized to them. The results are shown in the graphics of Figure 6. As we can see, in both collections, temporal contexts reduce significantly the number of terms in the training sets, in particular for larger dominance values. Employing a 90% dominance, the total number of terms used for classifying documents from ACM-DL is less than 10% of the baseline, while in MedLine collection is less than 5%. Notice that, for the ACM-DL collection, adopting a dominance lower than 20%, the reduction in total number of distinct terms is not so significant while for MedLine the same observation is valid for dominances up to 40%. This may be explained by the smaller number of classes in MedLine (7) than in the ACM-DL (11).

<table>
<thead>
<tr>
<th>Dominance</th>
<th>ACM Collection</th>
<th>MedLine Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>10%</td>
<td>90%</td>
<td>90%</td>
</tr>
<tr>
<td>20%</td>
<td>80%</td>
<td>70%</td>
</tr>
<tr>
<td>30%</td>
<td>70%</td>
<td>50%</td>
</tr>
<tr>
<td>40%</td>
<td>60%</td>
<td>40%</td>
</tr>
<tr>
<td>50%</td>
<td>50%</td>
<td>30%</td>
</tr>
<tr>
<td>60%</td>
<td>40%</td>
<td>20%</td>
</tr>
<tr>
<td>70%</td>
<td>30%</td>
<td>10%</td>
</tr>
<tr>
<td>80%</td>
<td>20%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Figure 6: Effort Reduction

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 Figure 7. 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 graphics together with the graphics in Figures 6 and 4, 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 in [24].

**Figure 7: Distinct Terms**

Despite the significant reduction on the number of total terms used when we increase the dominance, we also need to evaluate whether this reduction in effort is associated with a reduction in the uncertainty for sake of classification. We then calculated the entropy [26] for each training set selected by GreedyChronos while varying the dominance. Entropy, in our case, is understood as the necessary information for building a classification model. This metric quantifies the terms dispersion among classes of a collection, measuring the disorder of it. The average of these results is presented in Figure 8, where we can see that there is a significant decrease on the entropy as we increase the dominance, thus reducing the uncertainty and the effort to build a classifier.

**Figure 8: Uncertainty Reduction**

We confirm this last observation by analyzing the number of distinct classes that are associated with each test term in its respective training set. The result of this analysis for each collection is presented in Figure 9, which shows that an increase in dominance causes a decrease in the number of classes associated with a term. For example, using 0% dominance each test term occur, on average, in 8 distinct classes in the training sets for ACM-DL and 5 for MedLine. When we increase the dominance to 90% this number reduces to less than 2 for both collections. This result shows that, besides reducing the amount of data to be processed, GreedyChronos also reduces the uncertainty of the training sets. However, it is still necessary to analyze whether this reduction in the confusion contributes to determine the correct class of the test document.

One key question is whether the reduction in uncertainty also enables the construction of good classification models. In order to assess such question, we performed the following experiment. For each dominance value, we applied an extremely simple classifier to classify the test documents: a majority voting. This classifier consists of selecting the class with highest score in the training set induced by the temporal context and assign it as the class of correspondent test document. It should be noticed that better classifiers could and will be used in the future, and we believe that they will outperform our majority voting classifier. But our goal here is not to produce the best classifier that exploits the temporal information, but to show that our temporal contexts enable, in fact, uncertainty reduction and, consequently, correct classifications most of the time, even using such simple technique.

We employed three different criteria to calculate the score of each class. The first one (Class Frequency) simply selects the class that occurs most frequently in the training set. Let $df_t(C_i)$ be the number of training documents associated with class $C_i$, the score function can be defined by Eq. (2). The second criterion (Term Occurrence) selects the class that has the largest number of occurrences of test terms in the training set, based on the intuition that this class has the greatest probability of being the correct class.

Let $df_t(t_j, C_i)$ be the number of training documents associated with class $C_i$ that contains $t_j$ and $T$ be the total of distinct terms in test documents, the score function is defined by Eq. (3). The third criterion (Weighted Term Occurrence) is derived from the second one. It also consists of counting the occurrences of the test terms in each class in the training set. We smooth this value by applying the log function to it and by weighting this total number of occurrences by the ratio between the number of distinct test terms in the class ($|T_{C_i}|$) and the number of distinct test terms ($T$). The weight-based smoothing minimizes the impact of a class containing just few distinct test terms with high number of occurrences, which might present a higher score than a class containing more unique terms that occur less, towards a balance between the number of unique terms and their occurrences. The resulting score function is given by Eq. (4).

\[
\text{Score}(C_i) = df_t(C_i)
\]

\[
\text{Score}(C_i) = \sum_{j=1}^{T} df_t(t_j, C_i)
\]

\[
\text{Score}(C_i) = \log \left( \sum_{j=1}^{T} df_t(t_j, C_i) \right) \left( \frac{|T_{C_i}|}{T} \right)
\]

Figures 10 and 11 show the MicroF1 and MacroF1, respectively, for all majority voting classifiers for both collections. As we can see, as we increase the dominance, the MicroF1 and MacroF1 also increase in both collections. Therefore, as we increase the dominance, the uncertainty of the training sets reduces, while the convergence for the correct class of test document increases. For instance, for ACM-DL, the majority voting using the Weighted Terms Occurrence achieved a MicroF1 of 67.77% and a MacroF1 of 53.99% for 65% dominance. In the MedLine collection, using the Weighted Terms Occurrence, the majority voting achieved a MicroF1 of 75.38%
and a MacroF1 of 58.96% for 85% dominance.

We present in Table 4 a summary, for each collection of the best non-temporal (dominance = 0) version of the majority voting classifier, the best temporal majority voting classifier, and SVM. The table also includes the execution time of the algorithms. For the SVM classifier, the execution time represents the time for training and for classifying the test documents. For the temporal versions of majority voting, this time represents the time for calculating the training set for each test document (execution time of our heuristic) and the time to classify the test documents. Notice that, despite being somehow unfair, since we are comparing lazy and non-lazy algorithms, this comparison has two goals. First, it shows that the temporal context selection does not have a large impact in execution time. In fact, it is able to reduce the overall classification time of the majority voting classifiers. And second, to show that our method is a viable alternative in cases where the training time of SVM makes it an unfeasible choice due to the size of the collections, for example, the entire MedLine collection. Moreover, the classification model of SVM often needs to be rebuilt, according to the frequency of changes in the collections, worsening this problem.

We also tried a new SVM implementation [14], which creates the classification model in linear time but it did not produce good results in terms of effectiveness.

To compare these results, we applied a two-tailed paired t-test. Table 4 shows that the majority voting classifier with our temporal selections achieved gains of up to 30% against a version that did not consider temporal information. Adopting a 99% confidence level, we can say that the MicroF1 for the best majority voting classifier for ACM-DL collection and the SVM one are statistically equivalent and that MacroF1 for SVM are higher than the best majority voting classifier. Moreover, in MedLine collection, adopting this confidence level, we can say that the MicroF1 and MacroF1 for the best majority voting is better than SVM one.

We performed experiments using all dominances values since our goal was to characterize and evaluate our heuristic for selecting temporal contexts. As we can see, besides reducing the amount of data to be processed and the uncertainty, the training sets induced by the temporal context also present a good convergence to the correct class of the test documents. This indicates the potential of the temporal contexts, since, even using very simple classifiers like majority voting, we achieved equivalent results with the state-of-the-art classifier (SVM). In an actual scenario, it would be necessary to split the training set into training and validation, looking for dominance values (with folded cross-validation) that best perform in the validation set, in order to find good dominances, since specific characteristics of each collection affect the choice of the dominance. Moreover, we can get even better results if we use these temporal contexts with more effective techniques for ADC, such as nearest-neighbor, Bayesian, association-based classification or the SVM itself, as we propose as future work. In the case of SVM, though, this may be too expensive, since we would have to train it for each test document with its temporal context. Moreover, it is important to point out that the characteristics of the temporal contexts may change in relation to the original sampling, therefore some premises assumed by these algorithms may also change. Thus, it will be necessary to perform a more detailed analysis of how we can apply the techniques presented with others ADC algorithms.
computational efficiency of the document classifiers. These techniques often adopt a notion of context that is different from ours in the sense that they exploit semantic contexts, which arise, for instance, from the co-occurrence of terms. The temporal contexts we propose here may also be applied to these semantic contexts, exploiting their stability. Unlike those efforts to identify new contexts and to deal with the aforementioned challenges, in this paper we are concerned about performing document classification in its original temporal context, that is, categorize the document according to the period of time to which it belongs.

Concept or topic drift [31] comprises another relevant set of efforts. To deal with concept drift, one common approach is to retrain completely the classifier according to a sliding window of m examples [28, 18, 17, 33, 21]. In [28] the authors keep a window containing the most relevant documents. The method presented in [18] also maintain a window with the most representative documents and automatically adjust the window size so that the estimated generalization error is minimized. In [17], the methods presented either maintain an adaptive time window on the training data, select representative training examples, or weight the training examples. In [33] the authors describe a set of algorithms that flexibly react to concept drift and can take advantage of situations where contexts reappear. The main idea of these algorithms consists of keeping only a window of currently trusted examples and hypotheses; and storing concept descriptions and re-using them when a previous context reappears. Unlike previous works which use a single window to determine the drift in the data, in [21] the authors present a method that uses three windows of different sizes to estimate the change in the data.

Other common approach to deal with concept drift focuses on the combination of various models of classification generated from different algorithms for classification [29, 19, 10]. In [29], 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. Kolter et al. [19] present a technique that maintains an ensemble of base learners, predicts using a weighted-majority vote of these "experts", and dynamically creates and deletes experts in response to changes in performance. In [10] 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. In both approaches, the authors consider the existence of temporal effects without exactly understanding what these effects are. They discuss that, despite the fact that some classes present popularity explosion in some periods, their true identity does not modify. We, on the other hand, based on detailed empirical results that explain the temporal aspects [24], consider that there are other two effects beyond the explosion of popularity of certain classes (class distribution): class similarity and terms distribution. Moreover, although our proposal can be applied to deal with concept drift and classify new documents (data streams), we may also apply it to classify old documents.

Another field in which there is a concern about temporal issues is Adaptive Information Filtering [12]. Information Filtering [2] is described as a binary classification problem in which documents are classified as relevant or not with regard to the user's interest. According to [9], building classifiers in static domains is a sufficiently well controlled problem, and many applications assume that the training data distribution and the new data distribution are similar. This may be true for applications that last for a short period of time, but it becomes invalid to applications lasting long periods. In this case, it is unavoidable to adapt the classifier to new scenarios in order to maintain the quality of the classification.

There are some works that are somehow similar to ours in the sense that they investigate how the temporal aspect can be used to improve the quality of a certain process [15, 3, 35]. In [15] the authors show that taking temporal aspects into consideration when generating results for queries is very important and may improve the quality of the results. By analyzing the timeline of a query result set, it is possible to characterize how temporally dependent the topic is, and how relevant the results are likely to be. In [3] the authors propose an algorithm, named T-Rank, which extends PageRank to improve ranking by exploring the temporal proximity between Web pages and in [35] the PageRank algorithm is adapted by weighting each citation according to the citation date. Besides these, there are other efforts that exploit temporal evidence in Web Information Retrieval; a broad survey of them is presented in [25]. Although some of the premises of these effort are similar to ours, none of them address the problem of building classifiers that are robust w.r.t. temporal evolution.

7. CONCLUSION AND FUTURE WORK

In this work we propose a context selection strategy that is used for building classification models. Our strategy consists of selecting contexts which are a set of pre-classified documents which minimizes the three temporal effects (class distribution, terms distribution and class similarity), so that the resulting learned classifiers are less susceptible to temporal related changes. We started by identifying three requirements for the contexts to be selected: reference constrained, characteristic stability, and uncertainty reduction. Then we propose Chronos, an algorithm that can be used as a template by techniques that aim to generate temporal contexts.

Based on this algorithm we also propose GreedyChronos, a heuristic that effectively selects good temporal contexts. We evaluated GreedyChronos using two distinct collections: the ACM digital library (ACM-DL) containing documents about Computer Science, and MedLine, a digital library related to Medicine. We also evaluated both the size of the contexts generated and the uncertainty reduction they provide. Further, we used the resulting contexts for building a simple majority voting classifiers. Some of these simple classifiers were able to outperform in more than 30% compared to versions of them that are not temporally contextualized and produced similar accuracy than a state-of-the-art classifier (SVM), while presenting an execution time up to hundreds times faster.

As future work, we plan to apply the techniques presented in this paper along with several ADC algorithms in order to evaluate whether our technique is able to improve them. However, it is important to point out that the characteristics of the temporal contexts may change in relation to the original sampling, therefore some premises assumed by these algorithms may also change. For example, the rarity of features among classes may changes, impacting in important metrics like IDF (Inverse Document Frequency) and, consequently, in the algorithms that consider these metrics. Thus, it will be necessary to perform a more detailed analysis of how we can apply the techniques presented with others ADC algorithms.

We also want to adopt this technique on different Web collections to verify whether the same temporal evolution effects occur and can be investigated. Moreover, we intend to extend our techniques to consider, for example, other types of evidence, such as inlinks and outlinks within a temporal context.

8. ACKNOWLEDGMENTS

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


Morgan Kaufmann Publishers, San Francisco, US.


[31] A. Tsymbal. The problem of concept drift: Definitions and related work.


