**X-Classifier**  
A fleXible Classifier for the common user

[Extended Abstract]

Jorge Fernandes  
Instituto Superior Técnico  
Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal  
jorge.figueiredo.fernandes@gmail.com

**ABSTRACT**

Automatically managing collections of texts of considerable size is an important task because it allows the consultation of texts by topic or category, facilitating the individual search for a text, something that would be quite difficult to achieve without any help.

A popular way of managing text collections is through the task of text classification, which allows the grouping of texts by category. However, currently there is still no solution to the classification of any collection without the need of minor adjustments, which requires specific skills.

In this paper we propose a solution for this problem, introducing a new approach to text classification that adapts to any given problem without the need of any human intervention.

Experimental results show that our solution can achieve a very similar performance to that of a supervised approach, without requiring the user to select or configure the algorithms to use in order to make the classification.

**Keywords**  
Text classification, Bootstrapping, Supervised learning, Unsupervised learning

**1. INTRODUCTION**

In the midst of a digital era, there have been an increasing use of digital media, which creates collections of information with a considerable size. This is a problem because it turns the task of consulting any particular subject very hard to achieve. Therefore, there is a clear need of a system capable of managing/cataloging such collections.

One way of achieving this is through the task of text classification, where through a content analysis of the texts one can assign a theme or category, thus allowing us to better manage our collections of documents.

However, existing approaches to text classification require some effort and a high level of knowledge on this subject by users, making them inaccessible to the common user. Another problem of current approaches is that they are optimized for a specific problem, so it’s necessary to make a number of modifications for it to be used in another context, which is limiting and further strengthens the need for prior knowledge.

To solve these problems we propose a new approach for text classification, where the main focus is to make it flexible to different problems and easily used by any user. To achieve this, we designed a solution whose main characteristic is it's adaptive bootstrapping, that uses two algorithms of different characteristics, LSA and a modified version of TFIDF, making an automatic selection (and if necessary their parameterization) of which one is most appropriate to the problem in question, thus becoming fully transparent to the user.

With this paper our main contribution is a novel approach to text classification, that is independent of the problem domain and is accessible to any user. We also contributed with an extensive study of the performance of LSA and various algorithms based on keywords, leaving a clear picture of their behaviors, as well as strengths and weaknesses. As a direct conclusion from these studies we suggest an empirical formula to estimate the number of dimensions to use in LSA and introduce a new variant of the approaches based on keywords that presents a better performance.

Experimental results show that the bootstrapping process we developed is able to automatically adapt to different problems, with a performance between 80-100%, which is a good outcome. The classifier trained with the outcome of our bootstrapping also obtained good results. By analyzing the experimental results we see that the obtained values and their standard deviations intersect with those of a supervised approach under the same conditions, even under adverse conditions.

So, with our solution we are able to train a text classifier that obtains results similar to its equivalent supervised solution, but withouth its costs. Therefore we can conclude that our solution not only is adaptive to various problems
and accessible to all types of users but that it achieves it without affecting the performance.

The paper is organized as follows. Section 2 presents the related work, where we discuss the different approaches and how they influenced our solution. In Section 3 we describe in detail our approach and the ideas that are behind it. In Section 4 we evaluate our solution and discuss the obtained results. Section 5 concludes.

2. RELATED WORK

Most of the existing text classification techniques can be grouped into two groups, supervised learning and unsupervised learning. In this section we investigate both approaches in order to realize the current state of the art of the task of text classification and identify aspects that could be improved.

2.1 Supervised Learning

Supervised Learning requires a manual classification of a group of texts into a predefined set of categories. The results of the previous classification will later be used to train and build an automatic classifier able to categorize any text into the predefined set.

According to Huang [4], the key factors for success of supervised learning are the feature extraction, which should accurately represent the contents of text in a compact and efficient manner, and classifier design, which should take maximum advantage of the properties inherent to the texts in order to achieve the best possible results. Huang studied several algorithms for both factors, and in the end concluded that for the feature extraction the LSA algorithm would be the most appropriate as it offers clear advantages over the others due to its representation in a Latent Semantic Space. On the design of the classifier, Huang believes that the SVM classifier represents the best choice, because it’s strong performance is already documented in cases where the texts have the following properties: high dimension and sparseness.

Ishii et al. [5] also believe that one of the factors that determines the success of supervised learning is the feature extraction, agreeing that the most appropriate choice to solve it is the LSA. However the authors find LSA to be insufficient and introduce the concept of data grouping to further improve the results.

Debole and Sebastiani [2] agree with the importance of feature extraction but believe that the information on the membership of training documents to categories should also be used to determine term weights that will influence the training of the classifier. To accomplish that, the authors propose a number of “supervised variants” of TFIDF weighting, obtained by replacing the IDF function with functions typically used for term selection, such as \( \chi^2 \), information gain and gain ratio. As for classifier design, the authors also chose the SVM classifier.

In [6] Joachims proposes a different way of extracting the features, which consists in a combination of three distinct methods: pruning of infrequent words, pruning of high frequency words and choosing words with high mutual information.

Although this type of approach can obtain good results, they require a large number of texts (literature values vary between 500 and 14000) to train the final classifier, which is a problem.

2.2 Unsupervised Learning

Unsupervised Learning tries to solve the problem of the supervised approaches by replacing the need of a manual classification of a high number of texts with an automatic classification (often called bootstrapping) based on various techniques. In doing so it can greatly reduce the costs and the need for human intervention.

Unfortunately the automatic classification of texts proposed by unsupervised learning causes various misclassifications, which introduce some noise in the training of the classifier, affecting it’s final performance. For this reason, these approaches traditionally obtain results slightly lower than supervised learning.

In [3] Gliozzo et al. try to minimize the number of misclassifications from the process of bootstrapping by adding principled unsupervised mechanisms within the initial classification step. The approach of the authors begin by preprocessing the text, removing all the words that are not nouns, verbs, adjectives and adverbs, representing them later in a latent semantic space through the use of LSA. The authors then propose to use an algorithm based on unsupervised estimation of Gaussian Mixtures, with the purpose of differentiating relevant and nonrelevant category information using statistics from unclassified texts. Finally a SVM classifier is trained with the results of the bootstrapping proposed by the authors, and they conclude that their solution can achieve results comparable to a supervised solution where nearly 70-160 training texts by category were used.

Ko and Seo [8] propose a different way of approaching unsupervised learning. Instead of doing a automatic classification of the training texts, their approach consists of dividing the texts into sentences, classifying them and finally using the outcome to train the final classifier. By doing that, the authors argue that the proposed method achieves a similar degree of performance when compared with the traditional supervised learning methods.

In most unsupervised approaches, the user is asked to provide some representative keywords for each category in order to improve results. Liu et al. [9] propose a novel approach to assist the user in this matter, which consists of using k-means to cluster the unlabeled documents and then performing feature selection on the resulting clusters to identify those important words of each cluster. The end result is a list of all words ranked according to their discriminative power of all the categories which allows the user to inspect the ranked list and select a small set of representative keywords for each category in order to improve the bootstrapping process.

Barak et al. [1] also try to solve the problem of the users being required to provide representative keywords for each
category. However, unlike the approach suggested by Liu et al., that only helps the user with the selection, the proposed approach aims to eliminate this need by completely automating the process. To achieve this, the authors propose an approach that attempts to automatically extract possible keywords using only the category name as starting point. Barak et al. introduce a novel scheme that models both lexical references, based on certain relations present in WordNet and Wikipedia, and contextual references, using the model of the LSA. From the resulting model we can extract the necessary keywords.

Ko and Seo [7] also tackle the problem of keyword requirement by proposing a new text classification method that launches text classification tasks with only unlabeled documents and the name of each category for learning, and then it automatically learns the text classifier by using bootstrapping and feature projection techniques. Simply put, the authors claim to be able to extract keywords by using co-occurrence information between category names and the other words. To do so, they introduced the concept of a context (a sequence of 60 words in a document) and define it as a unit of meaning for the bootstrapping process. Contexts that contain at least one category name are called centroid-contexts and are used to extract keywords based on first-order co-occurrence. However, since these extracted keywords cannot sufficiently describe the meaning of the category, they collect more contexts by measuring similarities between centroid-contexts and the remaining contexts, using the end result to train the classifier.

However, such approaches require a great knowledge of the problem domain in order to be able to choose the most appropriate techniques. And if later we change the problem, we need to make a minimum of changes to maintain good performance.

2.3 Discussion

As we mentioned earlier, our main objective is to create a novel approach to text classification, one that is both accessible to any user and adaptive to any problem. Analyzing in detail the current approaches we concluded that these are still insufficient to achieve this goal.

Supervised learning requires the user to make a manual classification of a large number of texts, which is an arduous and monotonous task, and represents an enormous cost.

Unsupervised learning solves this problem by replacing the manual classification for an automatic one, but it does so by introducing bootstrapping techniques that require specific knowledge that a common user simply does not have and in some cases it is domain dependent.

To minimize direct intervention and user effort, it was necessary to avoid the manual text classification required by supervised learning, therefore we had to choose a unsupervised approach. However, since we want our solution to be flexible and adaptive we have to introduce major changes in the bootstrapping process in order to solve the identified problems of these approaches.

3. OUR APPROACH

The solution we propose introduces several new automatisations in the bootstrapping process, with the purpose of becoming completely transparent to the user and making the best choice of which bootstrapping algorithm to use.

3.1 Architecture Overview

To reduce the size of vocabulary and remove useless words that only introduce noise in categorization, our solution includes a module for text pre-processing that allows us to remove both infrequent words and stopwords, and manage the vocabulary.

In order to avoid a high degree of human intervention, we chose to follow an unsupervised approach that does not need a manual classification of texts. To correct the problems of this type of approach, we have chosen to incorporate several bootstrapping algorithms with distinct properties, and we introduce a system that is able to automatically choose the algorithm that best fits the problem, ending problem dependency. However, since some of these algorithms need to be
Since the bootstrapping techniques perform differently when used on problems with different characteristics, it is necessary to choose between various techniques the one that best fits the problem at hand in order to ensure the quality of the results in a more general context. To achieve this, we first need to know exactly how the individual performance of the various bootstrapping techniques varies when applied on problems with various characteristics and contexts. As such, we set out to study various bootstrapping algorithms, among which we emphasize the LSA and TFIDF.

### 3.3 Adaptive Bootstrapping Module

Since the bootstrapping techniques perform differently when used on problems with different characteristics, it is necessary to choose between various techniques the one that best fits the problem at hand in order to ensure the quality of the results in a more general context. To achieve this, we first need to know exactly how the individual performance of the various bootstrapping techniques varies when applied on problems with various characteristics and contexts. As such, we set out to study various bootstrapping algorithms, among which we emphasize the LSA and TFIDF.

#### 3.3.1 LSA

We opted to choose the LSA as one of the techniques for the study because through the analysis of the related work we concluded that the use of this algorithm was extremely beneficial when framed in the text classification problem.

In fact, LSA has several qualities that enhance the quality of the results, such as dimensionality reduction, non sparse representation and the discovery of high-level relations between groups of words.

The LSA in our solution was implemented as follows. The word-document matrix was built by grouping the individual document representations provided by the pre-processing module. TFIDF was then applied to the resulting frequency matrix, followed by SVD\(^1\) in order to obtain the reduced dimensionality matrices. The resulting latent semantic space was then used to classify the documents according to this approach:

\[ k = \sqrt{nt - {\frac{nd}{10}}} \]

Where \( nt \) represents the total number of texts, and \( nd \) the number of digits of the total number of texts.

#### 3.3.2 Keyword based algorithms

We decided to study some keyword based approaches because in ideal circumstances they are able to achieve good results and because they only depend on the number and quality of the keywords used to describe the categories.

We implemented both the traditional keyword classification and the TFIDF variant, which consist on using the cosine similarity measure to compare the word vector representation (weighted by TFIDF in the second case) between documents and categories.

By analysing their results, we quickly discovered that this kind of approach is unable to classify texts that do not contain the keywords, resulting in a significant amount of unclassified texts. To solve this problem, we devised an additional step, which we called Iterative, and that can be used in any of the above techniques. The basic idea of this step \(\footnote{We used the SmartMathLibrary provided in http://smartmathlibrary.codeplex.com/} \)

As we know, it is necessary to choose the appropriate number of dimensions to use accordingly to the problem in order to get satisfactory results. However, we cannot delegate this responsibility to the user, as this would imply that he had to be skilled in this subject and would violate our goal of making the solution transparent to the user. Therefore we needed to find a way to properly estimate the number of dimensions.

Reviewing the literature, we found that the LSA’s performance increases with the number of dimensions used until it reaches a maximum. Thereafter any further increase in the number of dimensions will only worsen the performance. Given this peak of performance, if no irregularities arise, the detection of a range where the function has a maximum value corresponds to the range where the optimum value lies.

With this information in mind we conducted a significant number of experimental tests where we analysed the performance of the LSA, using various values for the number of dimensions, for problems with different characteristics. After analyzing the results, we were able to obtain an empirical formula that we feel will be able to estimate the number of dimensions satisfactorily:

\[ k = \sqrt{nt - {\frac{nd}{10}}} \]

Where \( nt \) represents the total number of texts, and \( nd \) the number of digits of the total number of texts.

1. All categories and corresponding keywords were represented as pseudo-documents in the latent semantic space
2. Each document in the reduced matrixes was queried with each of the categories
3. After the queries, the category that obtained the higher similarity value\(^2\) was assigned to the document in question

As we know, it is necessary to choose the appropriate number of dimensions to use accordingly to the problem in order to get satisfactory results. However, we cannot delegate this responsibility to the user, as this would imply that he had to be skilled in this subject and would violate our goal of making the solution transparent to the user. Therefore we needed to find a way to properly estimate the number of dimensions.

Reviewing the literature, we found that the LSA’s performance increases with the number of dimensions used until it reaches a maximum. Thereafter any further increase in the number of dimensions will only worsen the performance. Given this peak of performance, if no irregularities arise, the detection of a range where the function has a maximum value corresponds to the range where the optimum value lies.

With this information in mind we conducted a significant number of experimental tests where we analysed the performance of the LSA, using various values for the number of dimensions, for problems with different characteristics. After analyzing the results, we were able to obtain an empirical formula that we feel will be able to estimate the number of dimensions satisfactorily:

\[ k = \sqrt{nt - {\frac{nd}{10}}} \]

Where \( nt \) represents the total number of texts, and \( nd \) the number of digits of the total number of texts.

1. All categories and corresponding keywords were represented as pseudo-documents in the latent semantic space
2. Each document in the reduced matrixes was queried with each of the categories
3. After the queries, the category that obtained the higher similarity value\(^2\) was assigned to the document in question

As we know, it is necessary to choose the appropriate number of dimensions to use accordingly to the problem in order to get satisfactory results. However, we cannot delegate this responsibility to the user, as this would imply that he had to be skilled in this subject and would violate our goal of making the solution transparent to the user. Therefore we needed to find a way to properly estimate the number of dimensions.

Reviewing the literature, we found that the LSA’s performance increases with the number of dimensions used until it reaches a maximum. Thereafter any further increase in the number of dimensions will only worsen the performance. Given this peak of performance, if no irregularities arise, the detection of a range where the function has a maximum value corresponds to the range where the optimum value lies.

With this information in mind we conducted a significant number of experimental tests where we analysed the performance of the LSA, using various values for the number of dimensions, for problems with different characteristics. After analyzing the results, we were able to obtain an empirical formula that we feel will be able to estimate the number of dimensions satisfactorily:

\[ k = \sqrt{nt - {\frac{nd}{10}}} \]

Where \( nt \) represents the total number of texts, and \( nd \) the number of digits of the total number of texts.
is to use the information of the texts already categorized to classify those who were left out. In detail what we do is replace the word vector representation of each category by a new one that extracts all the information contained in the texts that were classified in that specific category. We then use the new word vectors in order to classify the remaining texts, using the Tanimoto coefficient as the similarity metric.

3.3.3 Performance Study
To succeed in making the process of bootstrapping adaptive to any problem we needed to clearly distinguish which algorithms to use and under which circumstances we should be using them. To do so we did several experimental tests to define their strengths and weaknesses.

By analyzing the results we can draw several conclusions. The first conclusion we can draw is that the LSA algorithm (see Figure 2) produces unsatisfactory results when faced with problems with few texts available. On the other hand, for problems with a significant number of texts, it manages to make the best use of it’s properties and returns the best results among all algorithms.

![Figure 2: LSA Performance](image)

Among the keyword based approaches we found that the Iterative versions can get better results than their original versions, which proves that the step that we introduced added value to the typical approaches. As expected, this type of approach suffers when we use few keywords to describe the categories, but produces good results otherwise. They also seem to be unaffected by the number of texts, which is an important property to take into consideration. By analyzing all the results, we conclude that among these approaches, the one with better performance is the Iterative TFIDF (see Figure 3).

![Figure 3: IterativeTFIDF Performance](image)

In conclusion, this study allowed us to confirm that the use of a single bootstrapping technique is insufficient to guarantee the achievement of quality results when faced with changing problems. To solve this problem, we studied in detail the behavior of some techniques and found in what circumstances they achieve the best results. And more importantly, we discovered when to choose between each of them in order to adapt to the problem at hand.

3.3.4 Adaptive Bootstrapping
By analyzing the results of the individual performance of bootstrapping techniques (of different types) we found that they had very different behaviors. The LSA algorithm obtained poor results when applied in a problem with a small number of texts, but it is only slightly affected by a reduction of keywords. On the other hand, the Iterative TFIDF is severely affected by a reduction of keywords but remains unaffected by varying the number of texts. Complementing the two we can manage to get a good performance for both the variations of keywords and text.

Analyzing the subtle differences between the variations in performance of these algorithms, and knowing that getting the same results Iterative TFIDF presents itself as more advantageous due to represent a less complex time, our bootstrapping module adapts to the characteristics of the problem as follows:

<table>
<thead>
<tr>
<th>Total of texts</th>
<th>&lt;320</th>
<th>320-1000</th>
<th>≥1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 30 keywords</td>
<td>IterativeTFIDF</td>
<td>LSA</td>
<td>LSA</td>
</tr>
<tr>
<td>≥ 30 keywords</td>
<td>IterativeTFIDF</td>
<td>IterativeTFIDF</td>
<td>LSA</td>
</tr>
</tbody>
</table>

![Table 1: Informed choice between the techniques of bootstrapping](image)

3.4 Classifier Design Module
The classifier design module is responsible for building and training the SVM classifier, which was chosen due to it’s specific properties (high dimension and sparseness) that fit well in the text classification task. Specifically we choose the SVM implementation provided by the library LIBSVM.

This module provides the necessary functionalities to convert the text into the LIBSVM format and to automatically select the necessary parameters in order to optimize the training of the classifier.

As a direct result of the training, a classifier model is generated, which can then be used to classify any given text.
Again, it is necessary to convert the texts we want to classify into the LIBSVM format, which this module does automatically.

4. EVALUATION

In order to check whether our objectives have been met and that the addition of a higher degree of automation and transparency does not affect the final performance of our solution, we held an evaluation of the results produced by our new bootstrapping system, and the results produced by the trained classifier.

The evaluation methodology that we followed consisted of two phases. In the first phase we compared the results obtained by the bootstrapping based on the use of individual techniques, and our adaptive bootstrapping. In the second phase we compared the results obtained by the classifier trained by our solution and a classifier trained by an analogous supervised solution, where the texts were classified manually. By doing this, we intend to determine if our solution, which automates most of the classification process, is able to achieve results comparable to the supervised solutions where there is a strong component of human intervention.

4.1 Adaptive Bootstrapping Evaluation

Comparing the results obtained by our adaptive bootstrapping (See Figure 4) with the results of each individual approach (See Figures 2 and 3), we can easily verify that our solution provides a more steady performance, obtaining results almost exclusively in the range of 80% to 100%, which is a good outcome.

In fact, if we analyse in detail the results obtained by our adaptive bootstrapping while considering their standard deviations, we find out that there is only one value that does not intersects with the threshold of 80%, peaking at 79.74%.

Given this minimal difference, these results show that statistically the bootstrapping process introduced by our solution is able to obtain a performance equal to or greater than 80% over variable problems, something that has not yet been achieved (to the best of our knowledge) by the solutions present in the literature.

4.2 Classifier Evaluation

To prove the quality of the classifier created and trained by our solution it is necessary to compare its results with those obtained by using the same training texts but classified manually. This allows us to compare the practical results of the text classification done by our adaptive bootstrapping versus a human classification.

For this purpose several tests were devised in order to compare the performance achieved by our solution, both in favorable and unfavorable situations, with the performance of a classifier trained using the same texts classified manually. Two distinct situations were tested, one where we trained the classifiers with 500 texts and the task of classifying 1000 (See Figure 5), and another where the classifiers were trained with 1000 and had to classify 500 texts (See Figure 6).
Analyzing the results obtained by the classifier trained by our solution compared with the same classifier trained manually classified texts (supervised approach), we found out that in both studied situations we have obtained very similar results.

In fact, by doing a simple statistical comparison, we found out that the obtained values pondered by their standard deviations, in favorable and unfavorable scenarios, intersect with those obtained by the supervised approach. This allows us to state that the samples are statistically identical, and as such, we proved that our solution can achieve comparable results to supervised approaches, but without any of their costs.

<table>
<thead>
<tr>
<th>Unfavorable Scenario</th>
<th>Supervised Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.77487458286</td>
</tr>
<tr>
<td></td>
<td>0.80744108</td>
</tr>
<tr>
<td>Variance</td>
<td>0.036679916</td>
</tr>
<tr>
<td></td>
<td>0.050541812</td>
</tr>
<tr>
<td>Observations</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>T Stat</td>
<td>-3.321127793</td>
</tr>
<tr>
<td></td>
<td>0.069911401</td>
</tr>
<tr>
<td>T Critical two-tail</td>
<td>2.737945106</td>
</tr>
</tbody>
</table>

Table 2: Student t Test: Classifier trained with 1000 texts, Unfavorable Scenario

In order to further compare the results obtained by our solution to its analogous supervised, we undertook a deeper statistical study that was comprised of Student t tests for each situation (See Tables 2 to 5).

<table>
<thead>
<tr>
<th>Favorable Scenario</th>
<th>Supervised Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.83945268</td>
</tr>
<tr>
<td></td>
<td>0.85746108</td>
</tr>
<tr>
<td>Variance</td>
<td>0.019882896</td>
</tr>
<tr>
<td></td>
<td>0.0293412</td>
</tr>
<tr>
<td>Observations</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>T Stat</td>
<td>-2.203688762</td>
</tr>
<tr>
<td></td>
<td>0.069911401</td>
</tr>
<tr>
<td>T Critical two-tail</td>
<td>2.737945106</td>
</tr>
</tbody>
</table>

Table 3: Student t Test: Classifier trained with 1000 texts, Favorable Scenario

Finally, we have proved that a classifier trained with our solution is able to obtain results that are comparable, in most situations, to the supervised solutions, that typically obtained with state-of-the-art results.

5. CONCLUSIONS

In conclusion, we contributed with a new way of looking at the problem of text classification, not trying to maximize the results for a problem already well known, but instead featuring a more flexible solution that adapts to variable problems, that does not require user expertise or intervention and that is able to obtain a good performance.

We introduced a new variant on the keyword based techniques that takes advantage of the information present in the texts that couldn’t be classified by normal means, in order to significantly improve the end results. We also contributed with a formula capable of properly estimate the number of dimensions to use in the LSA algorithm.

With our studies and experimental testes we have confirmed the necessity of using more than one technique in the bootstrap-strapping process in order to maintain a satisfactory performance when faced with changing problems. To solve this, we introduced a process of adaptive bootstrapping, transparent to the user, which is capable of results at or above 80% (F1 score), even for problems with different characteristics.

Finally, we have proved that a classifier trained with our solution is able to obtain results that are comparable, in most situations, to the supervised solutions, that typically obtain state-of-the-art results.

6. REFERENCES


