Journal of Universal Computer Science, vol. 25, no. 4 (2019), 334-360
submitted: 26/6/18, accepted: 16/4/19, appeared: 28/4/19

Data-driven Feature Selection Methods for Text Classification: an Empirical Evaluation

Rogério C. P. Fragoso
(Universidade Federal de Pernambuco, Recife, Brazil
rcpf@cin.ufpe.br)

Roberto H. W. Pinheiro
(Universidade Federal do Cariri, Juazeiro do Norte, Brazil
roberto.hugo@ufca.edu.br)

George D. C. Cavalcanti
(Universidade Federal de Pernambuco, Recife, Brazil
gdcc@cin.ufpe.br)

Abstract: Dimensionality reduction is a crucial task in text classification. The most adopted strategy is feature selection using filter methods. This approach presents a difficulty in determining the best size for the final feature vector. At Least One FeaTure (ALOFT), Maximum f Features per Document (MFD), Maximum f Features per Document-Reduced (MFDR) and Class-dependent Maximum f Features per Document-Reduced (cMFDR) are feature selection methods that define automatically the number of features per Corpus. However, MFD, MFDR, and cMFDR require a parameter that defines the number of features to be selected per document. Automatic Feature Subsets Analyzer (AFSA) is an auxiliary method that automates such configuration. In this paper, we evaluate dimensionality reduction, classification performance and execution time of this family of methods: ALOFT, MFD, MFDR, cMFDR and AFSA. The experiments are conducted using three feature evaluation functions and twenty databases. MFD obtained the best results among the feature selection methods. In addition, the experiments showed that the use of AFSA does not significantly affect the classification performances or the dimensionality reduction rates of the feature selection methods, but considerably reduces their execution times.
Key Words: text classification, feature selection
Category: I.5.4, I.5.2

1 Introduction

Text Classification (TC) is the task of labeling documents into pre-defined categories [Sebastiani 2002]. This is usually achieved using machine learning algorithms, more specifically supervised learning algorithms. In the majority of machine learning approaches, an instance is represented as a vector composed of feature and value pairs. So, to represent textual documents as feature vectors it is necessary to carry out some procedure to extract features. A common approach for representing texts in the form of feature vectors is the technique known as Bag of Words (BoW) [Guyon, Elisseeff 2003].

BoW treats the text as a set of words, disregarding grammar or order of occurrence of the words in the text. Each word in the database vocabulary is considered a feature and is associated with the frequency of this word in the document. In other words, the size of the database vocabulary defines the dimensionality of the vectors. For instance, it is common for a medium-sized database to reach tens of thousands of dimensions [Gabrilovich, Markovitch 2004]. Such high dimensionality can make text classification too costly in terms of memory and execution time - or even impracticable for some approaches to machine learning [Yu, Liu 2003]. In addition, this large number of features may negatively impact classification performance, especially in databases with a small number of instances in relation to the number of features [Guyon, Elisseeff 2003]. Given that most features are irrelevant to classification, these problems can be addressed by restricting the amount of features of the database. This approach is known as dimensionality reduction (DR). By simplifying the feature space, DR seeks to increase the efficiency and accuracy of text classification.

An important methodology for DR is Feature Selection (FS). In this approach, a subset of the original features is selected to compose the final feature vector. Filtering methods are considered the more adequate FS technique for text classification problems because the computational cost is much lower than that of other techniques, such as wrapper methods [Yu, Liu 2003].

The classical approach for filtering FS is to sort the features using statistical metrics, known as Feature Evaluation Functions (FEFs). Feature Evaluation Functions are functions that calculate how important is a given feature giving its distribution among the classes of the database. The majority of FEFs are statistical measures such as Odds Ratio and Chi Squared, but many other are proposed to deal with text problems [Lewis, Ringuette 1994, Yang, Pedersen 1997, Sebastiani 2002]. After sorting, a number m (input parameter) of features is selected to compose the new subset. Since this method is very simple, it was commonly called by the name of the FEF used, for instance: Chi Squared Feature Selection. However, in this work, we use the nomenclature defined in [Guyon, Elisseeff 2003] that names this method as Variable Ranking (VR). Variable Ranking has been used with many FEF, such as Information Gain [Lewis, Ringuette 1994], Mutual Information [Yang, Pedersen 1997, Guyon, Elisseeff 2003], Chi-Squared [Yang, Pedersen 1997, Sebastiani 2002], Bi-Normal Separation [Lewis, Ringuette 1994], and Class Discriminating Measure [Chen et al. 2009].

Many other FS methods were proposed approaching unbalanced databases [Ogura et al. 2011], groups of interdependent features [Sun et al. 2013], combination of FS methods [Uğuz 2011, Ghareb et al. 2016], computation of feature importance by class [Uysal 2016, Tang et al. 2016] and feature ranking by class [Guru et al. 2018].

One difficulty faced by filtering methods is the determination of the best value for the m parameter. An exhaustive search to find the optimal value of m leads to an increase of the computational effort, since a classifier is required to assess each value for m. Hence, usually, only a few values for m are empirically evaluated [Yang, Pedersen 1997]. The following FS methods define automatically the number of feature per Corpus: At Least One FeaTure (ALOFT) [Pinheiro et al. 2012], Maximum Features per Document (MFD) [Pinheiro et al. 2015], Maximum Features per Document-Reduced (MFDR) [Pinheiro et al. 2015], and Category-dependent Maximum Features per Document - Reduced (cMFDR) [Fragoso et al. 2016b]. In contrast with VR that selects a number of features for the whole database, these methods select a number of features per document. This strategy leads to an easier parameter configuration concerning to the number of selected features. This paper evaluates this family of methods that were designed to alleviate the definition of the parameter m: ALOFT, MFD, MFDR, and cMFDR. However, these methods (except ALOFT) require a parameter that defines how many features are extracted per document. To automatically choose this parameter, an auxiliary method named Automatic Feature Subsets Analyzer (AFSA) [Fragoso et al. 2016b] is used. This evaluation is carried out with experiments using 3 FEFs, in each method, and 20 public datasets. The source-code of the methods are available online[1].

In a nutshell, the following research questions are investigated in this paper: i) which method of the family (ALOFT, MFD, MFDR, and cMFDR) that automatically choose the number of features per Corpus presents the best classification performance? ii) is it worth using the AFSA method? Which are the methods that take advantage of using AFSA? iii) in which conditions is it better to apply the best family method when compared with literature methods, Multinomial Naïve Bayes, SVM or Random Forest?

This document is organized as follows. Section 2 describes four FS methods and the auxiliary, AFSA, that can be used together with these methods. In Section 3, a toy example is presented to show how the methods work, their advantages and shortcomings. Section 4 exhibits the experimental configurations and the results obtained. Finally, Section 5 concludes the work.

2 Feature Selection Methods

The following sections describe four feature selection methods: At Least One FeaTure (ALOFT), Maximum f Features per Document (MFD), Maximum f Features per Document - Reduced (MFDR), and Category-dependent Maximum f Features per Document - Reduced (cMFDR). Additionally, an auxiliary method, Automatic Feature Subsets Analyzer (AFSA) is also presented.

[1] https://github.com/rcpf/featselection

For the explanation of the methods, we adopt the following notation: $\mathcal{D}_{tr}$, $\mathcal{D}_{tt}$ and $\mathcal{D}_{val}$ are a training set, a test set and a validation set, respectively. V is the total of features in $\mathcal{D}_{tr}$ and d is a document. The i-th document is represented by d_i and c_j is the j-th category. The h-th feature in $\mathcal{D}_{tr}$ is represented by w_h and w_{hi} represents the h-th feature of the i-th document.

2.1 At Least One FeaTure (ALOFT)

The classical Feature Selection (FS) method, Variable Ranking, selects the top m features to compose the final feature vector based on ranks calculated with statistical functions called FEF. This approach has two problems: it is hard to find the best value for m, normally m is determined in a try and error fashion and some documents may not contain any of the top m features, leading to empty feature vectors. At Least One FeaTure [Pinheiro et al. 2012] is a feature selection method designed to deal with these two problems.

ALOFT computes the discriminatory power for each feature w_h in $\mathcal{D}_{tr}$ using the given feature evaluation function (FEF). After, for each document, ALOFT selects the top ranked valued feature (weight greater than zero in that document). The selected feature is stored in the feature vector. If the selected feature is already present in the feature vector, ALOFT ignores this document and iterates to the next one. At the end of the process, each document in the training set is represented by at least one feature in the feature vector. Hence, the number m of features is automatically determined, in a data driven way.

The method assures a high dimensionality reduction, as the number of selected features is at most equals to the number of documents in the database. Furthermore, ALOFT's approach avoids empty feature vectors, since at least one feature is selected per document.

2.2 Maximum f Features per Document (MFD)

Maximum f Features per Document, as well as ALOFT, builds a global feature ranking to determine the most important features. However, in order to overcome ALOFT's shortcoming concerning its low number of selected features [Pinheiro et al. 2012], MFD introduces a parameter, f, which describes the number of features to be selected per document. Thus, instead of selecting only the top ranked feature for each document, MFD allows the user to configure the number of features to be selected per documents using the f parameter.

MFD computes the FEF value for each feature w_h in $\mathcal{D}_{tr}$. Then, for each document $d_i \in \mathcal{D}_{tr}$, f valued features with higher FEF values are evaluated in descending order. If the feature is not present in the FS vector, it is added to the vector. After MFD selects f top ranked features, the algorithm goes to the next document. At the end of this process, FS is the final feature set.

2.3 Maximum *f* Features per Document - Reduced (MFDR)

The best subsets produced by MFD are considerably larger than those generated with ALOFT. To address this drawback, MFDR [Pinheiro et al. 2015] method defines a threshold to determine whether a document will be considered in the feature selection process or whether it will be discarded. The hypothesis is that it is possible to reduce the size of the final feature vector by removing from the feature selection process some documents that do not contribute to the discrimination of categories.

MFDR computes the FEF value for each term in $\mathcal{D}_{tr}$. Then, the method computes the document relevance DR_i for each document d_i. This measure is calculated as

$$DR_i = \sum_{h=1}^{V} \big(S_h \times \text{valued}(w_{hi}, d_i)\big), \tag{1}$$

where the function $valued(\cdot)$ returns 1 if the w_{hi} is present in d_i, otherwise, returns 0. After, MFDR computes mean the DR using all documents in the database. This measure defines the threshold T, that is computed as

$$T = \frac{\sum_{i=1}^{|\mathcal{D}_{tr}|} \big(DR_i\big)}{|\mathcal{D}_{tr}|}. \tag{2}$$

Subsequently, MFDR discards every document d_i with DR_i less than T. From this point, MFDR behaves similarly to MFD. For each remaining document, the f features with higher FEF value are evaluated in descending order. If the feature is not present in the FS vector, the algorithm adds it to this vector. After MFDR analyzes f top ranked features, the algorithm iterates to the next document. At the end of this process, FS vector represents the final feature vector.

2.4 Category-dependent Maximum *f* Features per Document - Reduced (cMFDR)

cMFDR [Fragoso et al. 2016a] aims to improve classification performance by solving two problems identified in MFDR: threshold definition and DR calculation [Pinheiro et al. 2015]. cMFDR method introduces an improvement in the DR computation, shown in Eq. 3. Hence, from now on it will be called cDR. cMFDR intends to avoid the influence of the document size. To ensure a better comparison between document relevancies, the value of cDR of a document d_i is given by the average of the FEF values of the terms present in d_i.

Additionally, cMFDR defines different thresholds for each category, based on the relevancies cDR. The threshold CT for a given category is calculated as the average of the relevancies of the documents in this category. For a category c_j, only documents with cDR greater than $CT(c_j)$ participate in feature selection.

cMFDR computes the FEF value for each term w_h in $\mathcal{D}_{tr}$. The FEF values are stored in S vector. Later, cDR vector, which stores the relevance of each document, is computed as follows:

$$cDR_i = \frac{\sum_{h=1}^{V}\big(S_h \times \text{valued}(w_h, d_i)\big)}{\sum_{h=1}^{V} \text{valued}(w_h, d_i)}, \tag{3}$$

where S is the vector that stores the FEF value of the term w_h in its hth position. The function $valued(\cdot)$ returns 1 if a given term is valued in a given document, otherwise, it returns 0. The next step is to compute CT vector, which stores the thresholds for each category in $\mathcal{D}_{tr}$. CT computation is given by

$$\text{CT}(c_j) = \frac{\sum_{i=1}^{|\mathcal{D}_{tr}|}\big(cDR_i \times \text{belongs}(d_i, c_j)\big)}{\sum_{i=1}^{|\mathcal{D}_{tr}|} \text{belongs}(d_i, c_j)}, \tag{4}$$

where the function $belongs(\cdot)$ returns 1 if a given document belongs to a given category, otherwise, returns 0. The final feature vector is computed in the next step. A document d_i belonging to c_j is ignored if cDR_i is less than the threshold $CT(c_j)$. From this point, cMFDR method works similarly to MFD and MFDR. For each remaining document, the f features with higher FEF value are evaluated in descending order. If the feature is not present in FS vector, the feature is added to the vector. After cMFDR analyzes the f top ranked features for this document, the algorithm iterates to the next document. At the end of this step, FS vector represents the final feature vector.

2.5 Discussions

In this section we introduce a discussion about the family of feature selection methods presented in the previous sections (Section 2.1 to Section 2.4), their strengths and weaknesses.

ALOFT is the first method of this family. It was designed to overcome a problem concerning classical filtering methods: documents not represented in the final feature vector, leading to empty feature vectors. ALOFT introduces the FS per document, instead of FS per database. With this approach, ALOFT assures that every document is represented in the final feature vector by at least one feature, avoiding empty feature vectors. In addition, ALOFT determines the size of the final feature vector in a data driven way. Experiments carried out by [Pinheiro et al. 2012] show that ALOFT produces smaller features vectors and presents similar or better performance than the classical Variable Ranking. However, for some combinations of FEF and database, the number of selected features may be not sufficient to discriminate all categories. To improve the performance in such situations, MFD was proposed [Pinheiro et al. 2015].

Unlike ALOFT, MFD requires a parameter configuration (f) that guides the definition of the feature vector size. However, the choice of the optimal value for f is much easier than the configuration of m parameter in classical FS methods [Pinheiro et al. 2015]. This parameter configuration allows MFD to select a sufficient number of features to discriminate the categories. Experimental results [Pinheiro et al. 2015] show that MFD outperforms ALOFT. However MFD produces larger feature vectors than ALOFT and its best results are obtained with large feature vectors. Furthermore, some documents that do not contain relevant features may hinder classification performance. For instance, if a document presents only two relevant features, but f parameter is configured with value 5, three low FEF features are added to the final feature vector.

The next FS method proposed in this family is MFDR that defines a restriction to a document to be considered in the feature selection process. A threshold is defined based on the documents relevancies (DRs). MFDR present similar or better performance than MFD and achieves its bests results with less features than MFD. Despite MFDR's good performance and its small feature vectors, with MFDR, some categories present a much worse performance than others. This is due to the fact that the method uses a single threshold for the entire database. For instance, if a category c_j is composed of documents containing lower-FEF terms than other categories, the relevancies DR of documents belonging to c_j will hardly surpass the global threshold. Thus, the category c_j may be under-represented in the feature selection process. In other words, a document considered irrelevant (DR less than the mean DR of all documents) may be important for a particular category.

Another point is that MFDR computes DR as the sum of FEF values of the valued features of a document. Thus, documents with a large number of terms can be privileged, even if they do not contain relevant terms to discriminate categories. Documents with a large number of features with low FEF values can surpass the threshold and be, therefore, considered relevant, whereas documents with a small number of features with high FEF values can be discarded. For example, a document d_1 composed of 10 terms, all with FEF value equals 1, has $DR = 10$. Another document d_2 that contains only one term, with FEF value equals to 9, has $DR = 9$. Thus, d_1 is considered more relevant than d_2, even containing only low FEF terms. If the threshold T is a value between 9 and 10, document d_1 is considered relevant while d_2 is discarded. Thus, the important information contained in d_2 is lost.

On the other hand, in cMFDR the threshold is computed for each category in $\mathcal{D}_{tr}$. Thus, documents are considered relevant or not to a specific category instead of to the whole database, as in MFDR. This approach avoids under-representation of categories in the feature selection process. Furthermore, cMFDR improves the calculation of document relevance, in order to prevent the influence of the document's size. Using the same value for f parameter, cMFDR achieves best performance than MFDR, but selects more features. However, cMFDR presents best performance than MFDR even with lower values for f, when cMFDR selects a similar number of features compared with the best subsets generated with MFDR [Fragoso et al. 2016a].

MFD, MFDR, and cMFDR methods introduce an easier approach to configure the feature vector size than classical feature selection methods. However, this approach still requires an effort to determine the value of the f parameter that produces the subset with the best classification performance. In a production environment, with new documents being presented, it is necessary to reanalyze the database, from time to time, to include these new documents in the feature selection process. Whenever this analysis is performed, the user must manually check the subsets generated by these methods to determine which has the best classification performance. Thus, several values of the f parameter need to be evaluated, which is a computational expensive process. In this sense, the Automatic Features Subsets Analyzer (AFSA) [Fragoso et al. 2016b] was proposed to provide an automatic configuration o f parameter. AFSA is described in the next section.

2.5.1 Automatic Feature Subsets Analyzer (AFSA)

AFSA is an auxiliary method for FS methods that use the f parameter to determine the number of features to be selected per document. AFSA determines in a data driven way the best value for the f parameter. AFSA generates a number of subsets, using MFD, MFDR or cMFDR FS algorithm, pre-assesses the performance of each subset using a validation set, and finally chooses the f value that generates the most effective subset. This f value is passed to the algorithm used to generate the subsets (MFD, MFDR or cMFDR), which constructs the final feature vector using the training data. The final performance is evaluated using a test set.

The method requires a parameter n, which determines the number of subsets to be generated and pre-assessed. For each value of f in $[1, n]$, MFD, MFDR or cMFDR perform feature selection and generate a training subset $\mathcal{D}tr(f)$ and validation subset $\mathcal{D}val(f)$ and construct a classifier $\mathcal{C}_f$, using $\mathcal{D}tr(f)$. Then $\mathcal{D}tr(f)$ is pre-assessed using $\mathcal{C}_f$ and $\mathcal{D}val(f)$. The result is stored in the fth position of R vector. In the next step, each value of the R vector is analyzed looking for the subset with the best classification performance. The f value that produced the subset with best performance, i.e., the position of the R vector that holds the best performance, is stored in bf. The algorithm then generates the test set $\mathcal{D}_{tt}(bf)$ and uses it with the classifier $\mathcal{C}_{bf}$ to evaluate the final performance .

3 Toy Example

This section presents a toy example to compare the features selected methods described in Section 2. Table 1 shows a training set ($\mathcal{D}_{tr}$) composed of 12 documents (d_1 to d_{12}), each represented by 7 features (w_1 to w_7) and labeled (C) as one of the two classes (A and B). The document relevance is calculated using Equation 1 for the MFDR (DR) and Equation 3 for the cMFDR (cDR). The bottom line of Table 1 shows the score (S) for each feature. The chosen FEF to calculate S is the Document Frequency (DF) that counts how many documents a given feature appears, i.e., has the value 1. DF is an unsupervised approach since it does not require class information.

Table 1: Toy example showing each training set document ($d_i \in \mathcal{D}_{tr}$) with its features (w_h), class (C) and document relevance for MFDR (DR) and cMFDR (cDR). The bottom line presents the score of each feature (vector S).

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	C	DR	cDR
d_1	1	1	1	0	0	0	1	A	24	6.00
d_2	1	1	1	0	1	0	0	A	27	6.75
d_3	1	1	1	0	1	1	0	A	30	6.00
d_4	1	1	1	1	1	1	0	A	32	5.33
d_5	1	1	1	0	0	0	0	A	23	7.67
d_6	0	1	0	0	1	1	0	B	17	5.67
d_7	0	1	1	0	0	0	0	B	17	8.50
d_8	0	1	1	1	0	0	0	B	19	6.33
d_9	1	1	0	0	0	0	0	B	16	8.00
d_{10}	0	1	0	0	0	0	0	B	10	10.00
S	6	10	7	2	4	3	1	-		

This toy example covers ALOFT, MFD, MFDR and cMFDR. Only AFSA is not showed in this toy example since it is an improvement that can be used with any of these methods. The classes of this toy problem can be perfectly classified using the features w_1 and w_3.

Starting with ALOFT, this method does not use the information of the last two columns (DR and cDR) nor the bottom line (S). ALOFT selects only one feature per document. Therefore, the new feature vector is composed of a single feature w_2. This occurs because w_2 is present in every document and has the highest FEF value. This problem can occur in real databases with the wrong choice of FEF. This aggressive feature selection is the main problem of ALOFT and it is alleviated using its successor: MFD.

MFD selects f features per document and still does not use the information of the last two columns of Table 1. Let's assume, for simplicity sake, that $f = 2$ for MFD and all the subsequent methods. In this manner, MFD selects w_2 plus the feature with the second highest FEF value in each document. Therefore, MFD selects 4 features: w_2, w_3, w_5 and w_1. Now, there are enough features to discriminate the classes. However, w_2 and w_5 are not necessary. Therefore, increasing the number of features selected per document solves the problem of ALOFT, but can result in selecting too much features. MFDR is proposed to insert restrictions to MFD and reduce the number of features selected without being as aggressive as ALOFT.

MFDR requires the DR information and also a threshold, which in this Toy Example is $T = 21.50$ (the mean of DR). Now, not every document is used during the selection. Only documents with $DR > T$ are considered. Therefore, only 5 documents (d_1, d_2, d_3, d_4 and d_5) are used to select the features. Maintaining $f = 2$, MFDR selects only 2 features: w_2 and w_3. MFDR is less aggressive than ALOFT, but still does not select the best subset of features. The main reason for such behavior is the calculation of DR, which disregards the number of features in the documents. Therefore, documents with lots of features are considered important, which is rarely the case. Documents with lots of features increase the chance of selecting irrelevant features. Furthermore, the global threshold T may discard documents belonging to classes with documents containing low FEF features. In this Toy Example, MFDR only uses documents belonging to class A. cMFDR takes into account the problems of MFDR and alleviates them.

cMFDR calculates the document relevance (cDR) per class. For the toy example, the thresholds are $CT(A) = 6.35$ and $CT(B) = 7.70$. A document is only considered when $cDR > CT(class(d_i))$. The threshold is not biased by documents from other classes, because each class is treated separately. cMFDR selects 3 features: w_2, w_3 and w_1. cMFDR presents three important aspects: i) it avoids the traps of selecting few features because of the parameter $f = 2$; ii) it does not select too much features because only five documents were considered (d_2, d_5, d_7, d_9 and d_{10}); and, iii) it avoids great part of the irrelevant features because of the class dependent threshold.

4 Experiments

This section presents the experiments carried out to analyze the performance of the feature selection methods: AFSA, cMFDR, MFDR and MFD. The 10-fold stratified cross-validation was employed in the experiments, where the database $\mathcal{D}$ was partitioned into 10 folds of similar sizes, keeping the proportion of documents per category equivalent to the proportion found in the original database. Since AFSA method requires one fold for validation, only in this method it was used eight folds for training, one for validation and one for testing.

MFD, MFDR and cMFDR methods require a positive integer for the f parameter, which determines how many features should be selected per document. The best classification performances of these methods are achieved with f assuming values between 1 and 10 [Pinheiro et al. 2015, Fragoso et al. 2016a]. Thus, the experiments were carried out with f varying from 1 to 10 for these feature selection methods. AFSA provides automatic parametrization of the f parameter for MFD, MFDR and cMFDR methods, but requires a parameter (n) that is used to determine the number of subsets to be generated by MFD, MFDR or cMFDR. The subsets are generated with f values ranging from 1 to n. Thus, $n = 10$ was adopted in the experiments to assure a fair comparison among the methods.

The classification performances of the FS methods evaluated in this paper are assessed using Multinomial Naïve Bayes classifier. This classifier is widely adopted for TC tasks because of its simplicity and efficiency in this kind of problems [Chen et al. 2009]. In order to compare our methods against benchmark classifiers, Multinomial Naïve Bayes, Support Vector Machines (SVM) and Random Forest classifiers were also executed without using any kind of dimensionality reduction on the databases. The choice of SVM for this comparison was due to its acknowledged ability to deal with high dimensionality problems [Leopold, Kindermann 2002]. Random Forest was chosen because it is one of the most successful general purpose classifiers [Biau, Scornet 2016]. Additionally, our methods are compared with Latent Semantic Analysis, a.k.a. Latent Semantic Index (LSI). LSI, is a technique for indexing and retrieval that projects document vectors of an original feature set into the space of semantic concepts from a collection of documents [Deerwester et al. 1990]. Although LSI is a feature extraction method, we included it in the comparison because it is a widely used technique for dimensionality reduction in text classification. We carried out experiments with LSI using 10, 25, 50, 100 and 300 top extracted features. The best results were achieved using 100 features, thus, in this paper, we report only the results obtained using this value.

The classification performances obtained using Multinomial Naïve Bayes on the feature vectors generated by ALOFT, MFD, MFDR, cMFDR and AFSA are compared with the classification performance obtained using Multinomial Naïve Bayes, SVM and Random Forest classifiers on the original feature vectors and Multinomial Naïve Bayes and Random Forest on the features vectors extracted using LSI. The comparison was evaluated in terms of performance and execution time.

Three feature evaluation functions (FEFs) that reported good results for TC [Pinheiro et al. 2012, Uysal, Gunal 2012] were adopted in the experiments, namely: Bi-Normal Separation (BNS) [Forman 2003], Class Discriminating Measure (CDM) [Chen et al. 2009] and Chi-Squared (CHI) [Yang, Pedersen 1997].

In their formulas (Eq. 5 to 7), the following notation was adopted: w is the evaluated feature; $P(w|c_j)$ is the probability of the feature w to occur in class c_j; $P(w|\bar{c_j})$ is the probability that the feature w does not occur in class c_j; $P(\bar{w}|\bar{c_j})$ is the probability that every feature but w does not occur in class c_j; $P(w)$ is the probability that a random document contains the feature w; $P(\bar{w})$ is the probability that a random document does not contain the feature w; $P(c_j)$ is the probability that a random document belongs to class c_j; $P(\bar{c_j})$ is the probability that a random document does not belong to class c_j; C is the number of categories. In Equation 5, F^{-1} represents the Normal inverse cumulative distribution function.

$$\text{BNS}(w) = \sum_{j=1}^{C} \left| F^{-1}(P(w|c_j)) - F^{-1}(P(w|\bar{c_j})) \right| \tag{5}$$

$$\text{CDM}(w) = \sum_{j=1}^{C} \left| \log \frac{P(w|c_j)}{P(w|\bar{c_j})} \right| \tag{6}$$

$$\text{CHI}(w) = \sum_{j=1}^{C} \frac{\left[P(w|c_j)P(\bar{w}\bar{c_j}) - P(w|\bar{c_j})P(\bar{w}|\bar{c_j})\right]^2}{P(w)P(\bar{w})P(c_j)P(\bar{c_j})} \tag{7}$$

The next subsections are organized as follows. Subsection 4.1 shows the performance measures used in this paper to evaluate the methods. The databases used in the experiments are described in Subsection 4.2. The results achieved in the experiments are presented in Subsection 4.3 and Subsection 4.3.1 details the statistical tests performed.

4.1 Performance measures

Several measures may be used to evaluate classification performance [Sebastiani 2002]. In this paper, Micro-Averaged-F1 (Mircro-F1) and Macro-Averaged-F1 (Macro-F1) are adopted as performance measures. Both measures are computed using Equation 8:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \, . \tag{8}$$

The equations used to compute precision and recall for a category c_j are:

$$\text{Precision}(c_j) = \frac{TP_j}{TP_j + FP_j} \tag{9}$$

$$\text{Recall}(c_j) = \frac{TP_j}{TP_j + FN_j}, \tag{10}$$

where TP_j is the number of instances correctly classified as belonging to the category c_j, FP_j is the number of instances incorrectly classified as belonging to the category c_j and FN_j is the number of instances incorrectly classified as not belonging to the category c_j.

Both Micro-F1 and Macro-F1 are calculated as a combination of precision and recall of each category. However, the averaging process for the entire database is different for each measure. The equations from 11 to 14 refer to the precision and recall calculations used in the Micro-F1 (μ) and Macro-F1 (M) computations. In these equations, C represents the number of categories in the database.

$$Precision_{\mu} = \frac{\sum_{j=1}^{C} TP_j}{\sum_{j=1}^{C}(TP_j + FP_j)} \quad (11) \quad Recall_{\mu} = \frac{\sum_{j=1}^{C} TP_j}{\sum_{j=1}^{C}(TP_j + FN_j)} \quad (12)$$

$$Precision_{M} = \frac{\sum_{j=1}^{C} Precision(c_j)}{C} \quad (13) \quad Recall_{M} = \frac{\sum_{j=1}^{C} Recall(c_j)}{C} \quad (14)$$

Micro-F1 expresses better the overall performance, but on databases with class imbalance, large categories would dominate the small ones. Macro-F1 reflects better the performance of minority categories, since it treats categories equally in averaging.

Table 2: Databases description.

Database	# Cat.	# Docs.	# Features	Sparsity (%)	Maj. cat. (%)	Cat. S. D.	Domain
20 Newsgroup [1]	20	18,821	70,216	99.73	5.31	0.51	E-mails
CSTR [2]	4	299	1,726	96.86	42.81	15.89	Scientific
Oh0 [2]	10	1,003	3,183	98.35	19.34	5.33	Medical
Oh5 [2]	10	918	3,013	98.19	16.23	3.72	Medical
Oh10 [2]	10	1,050	3,239	98.28	15.71	4.25	Medical
Oh15 [2]	10	913	3,100	99.97	5.06	1.26	Medical
Reuters0 [2]	13	1,504	2,887	98.21	40.43	11.56	News
Reuters1 [2]	25	1,657	3,759	98.60	22.39	5.54	News
Reuters10 [3]	10	9,980	10,987	99.18	39.72	12.15	News
Review-Polarity [2]	2	2,000	15,698	98.69	50.00	0.00	Sentiment
SyskillWebert [2]	4	334	4,340	97.85	41.02	10.75	Web pages
TDT2T30 [3]	30	9,394	36,093	99.39	19.63	5.12	News
Tr11 [2]	9	414	6,430	95.62	31.88	9.80	TREC
Tr12 [2]	8	313	5,805	95.29	29.71	7.98	TREC
Tr21 [2]	6	336	7,903	94.05	68.75	25.88	TREC
Tr23 [2]	6	204	5,833	93.39	44.61	15.58	TREC
Tr41 [2]	10	878	7,455	95.13	27.68	9.13	TREC
Tr45 [2]	10	690	8,262	96.60	23.19	6.69	TREC
WAP [2]	20	1,560	8,461	98.33	21.86	5.20	Web pages
WebKB4 [1]	4	4,199	7,770	98.05	38.99	11.18	Web pages

[1] `http://ana.cachopo.org/databases-for-single-label-text-categorization`
[2] `http://sites.labic.icmc.usp.br/text_collections`
[3] `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`

4.2 Databases

The experiments were carried out using 20 databases that present a wide variety in terms of size, dimensionality, number of categories, domains and skewness. Table 2 shows the aspects of the databases used in the experiments. In this table, "# Cat" represents the number of categories; "# Docs", the number of documents; "# Features", the number of features in BoW representation; "Sparsity, the proportion of zeros in the BoW representation; "Maj. cat." means the proportion of documents that belong to the majority category; "Cat. S.D." means the standard deviation considering the percentage of documents that belong to each class; and "Domain" represents the purpose of the database. TREC domain means that the database was obtained in Text REtrieval Conference.

4.3 Experimental Results

The experiments evaluate dimensionality reduction rates, classification performances and execution times obtained by the different combinations of feature selection methods, feature evaluation functions and parameters configurations.

The results for MFD, MFDR, cMFDR methods and their combination with AFSA are depicted in Figures 1(a), 1(b) and 1(c), respectively. Each pair of figures compares Micro-F1 and Macro-F1 performances.

MFD, MFDR and cMFDR are represented as points, one for each value for f parameter. AFSA results are represented as horizontal slashed lines. Each color represents one of the feature evaluation functions: red for BNS, green for CDM and blue for CHI. Each result shown in Figure 1 represents the average performance of a method/configuration across all 20 databases (described in Section 4.2). For benchmarking purposes, the figure also provides the classification performances obtained by Multinomial Naïve Bayes, SVM and Random Forest classifiers on the original feature vectors and Multinomial Naïve Bayes and Random Forest on the features vectors extracted by LSI. Their results are presented as horizontal lines.

Among the MFD, MFDR and cMFDR methods, the best classification performances, for both Micro-F1 and Macro-F1, are achieved by MFD. The results obtained by cMFDR are close to MFD. MFDR method presents the lower Micro-F1 and Macro-F1 scores. Regarding AFSA, its combination with MFD and cMFDR yield the best performances. AFSA+MFDR is the AFSA configuration that obtains the most feeble results. The automatic parameterization provided by AFSA leads to performances similar to the best configurations of the base feature selection methods (MFD, MFDR and cMFDR).

In general, it is noteworthy that CHI is the FEF that presents the best results. With this feature evaluation function, MFD, AFSA+MFD, cMFDR and AFSA+cMFDR methods outperform Naïve Bayes, SVM and Random Forest

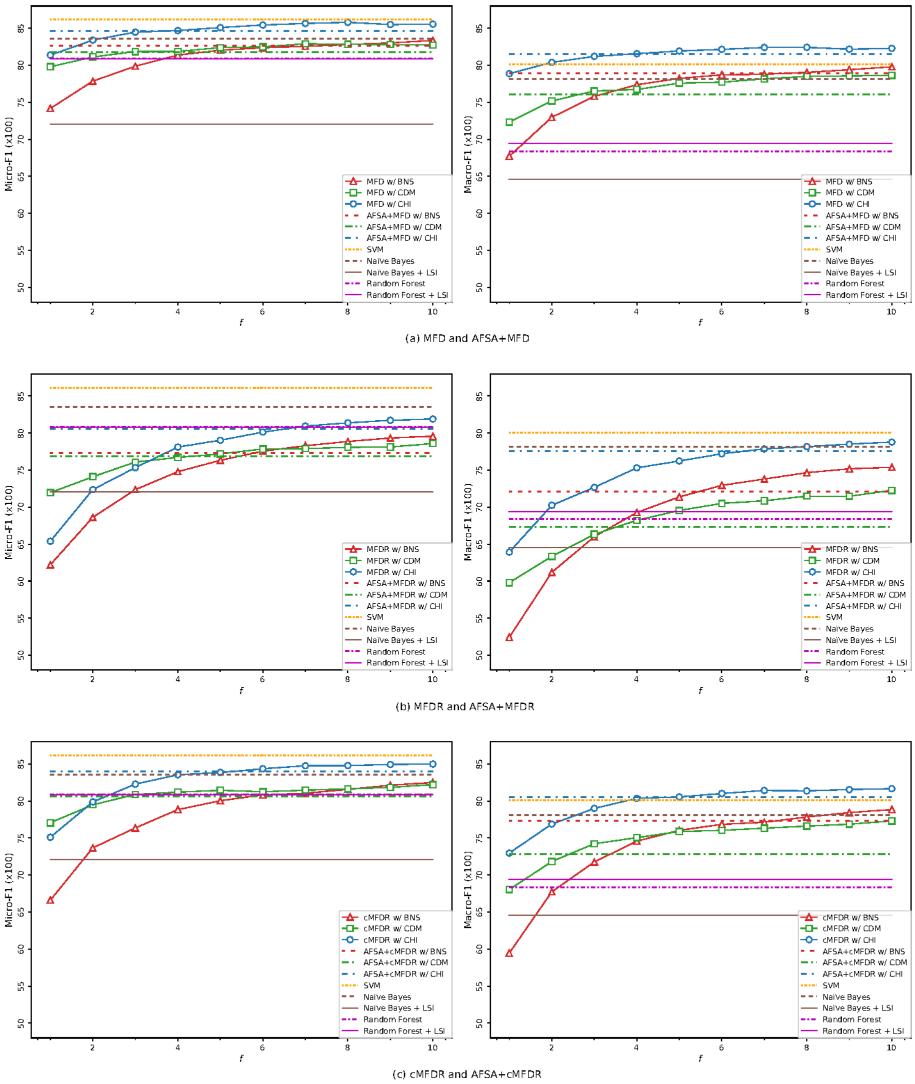

Figure 1: Comparisons: (a) MFD vs. AFSA+MFD, (b) MFDR vs. AFSA+MFDR and (c) cMFDR vs. AFSA+cMFDR.

classifiers using the original databases, in terms of Macro-F1. Furthermore, concerning to Micro-F1, MFD, AFSA+MFD, cMFDR and AFSA+cMFDR methods perform similarly to SVM classifier and outperform Naïve Bayes and Random Forest using the original databases. However, such results are achieved using less than 10% of features of the original databases, while Naïve Bayes, SVM and Random Forest classifiers use all features. Our methods outperform both Naïve Bayes and Random Forest, using the features extracted with LSI, for the majority of the configurations. These techniques presented the worst results, markedly Naïve Bayes with LSI. It is important to note that LSI presents feeble Macro-F1 score. This indicates that this technique may not deal well with examples from minority classes.

Figure 1 shows the classification performances of the FS methods. However, the classification performance alone is not sufficient to state that a method is superior to other. FS methods face a dichotomy between performance and dimensionality reduction (DR). Thus, it is also important to analyze DR rates.

Figure 2 shows scatter plots relating classification performance and dimensionality reduction. Each pair of figures, Figures 2(a), 2(b) and 2(c) present a scatter plot for Micro-F1 vs. dimensionality reduction and Macro-F1 vs. dimensionality reduction, comparing one of MFD, MFDR or cMFDR methods and its combination with AFSA. Dimensionality reduction is shown as the percentage reduction of the feature space. For MFD, MFDR or cMFDR methods, Figure 2 exhibits the results for each value for f parameter, from 1 to 10, as hollow markers in the graphs. The results of the AFSA are shown as filled markers. Dashed lines indicate the dimensionality reduction and classification performance, for each AFSA result. These lines are used to verify the efficiency of AFSA method against the MFD, MFDR and cMFDR methods. Results located in the upper right quadrant formed by the lines are considered better than the AFSA result represented by these lines. The results for Naïve Bayes, SVM and Random Forest classifiers using the original databases are not presented in the figure, since in these cases, there is no dimensionality reduction.

Among MFD, MFDR or cMFDR methods, MFDR is the one that achieved the best DR rates (99.8% for BNS and $f = 1$). For AFSA, its combination with MFDR, using BNS, achieved the highest reduction rate: 97.9%. MFD and AFSA+MFD yielded the lowest reduction rates. In general, BNS is the feature evaluation function that achieved the best reduction rates. For AFSA, CHI obtained the highest DR rates, except AFSA+MFDR.

By analyzing the results of each FS method (MFD, MFDR or cMFDR) and its combination with AFSA, it is noted that, for each AFSA result, less than 3 base method results, on average, are located in the upper right quadrant. That is, on average, less than 3 results of the base method present a better trade-off between performance and dimensionality reduction than AFSA. For

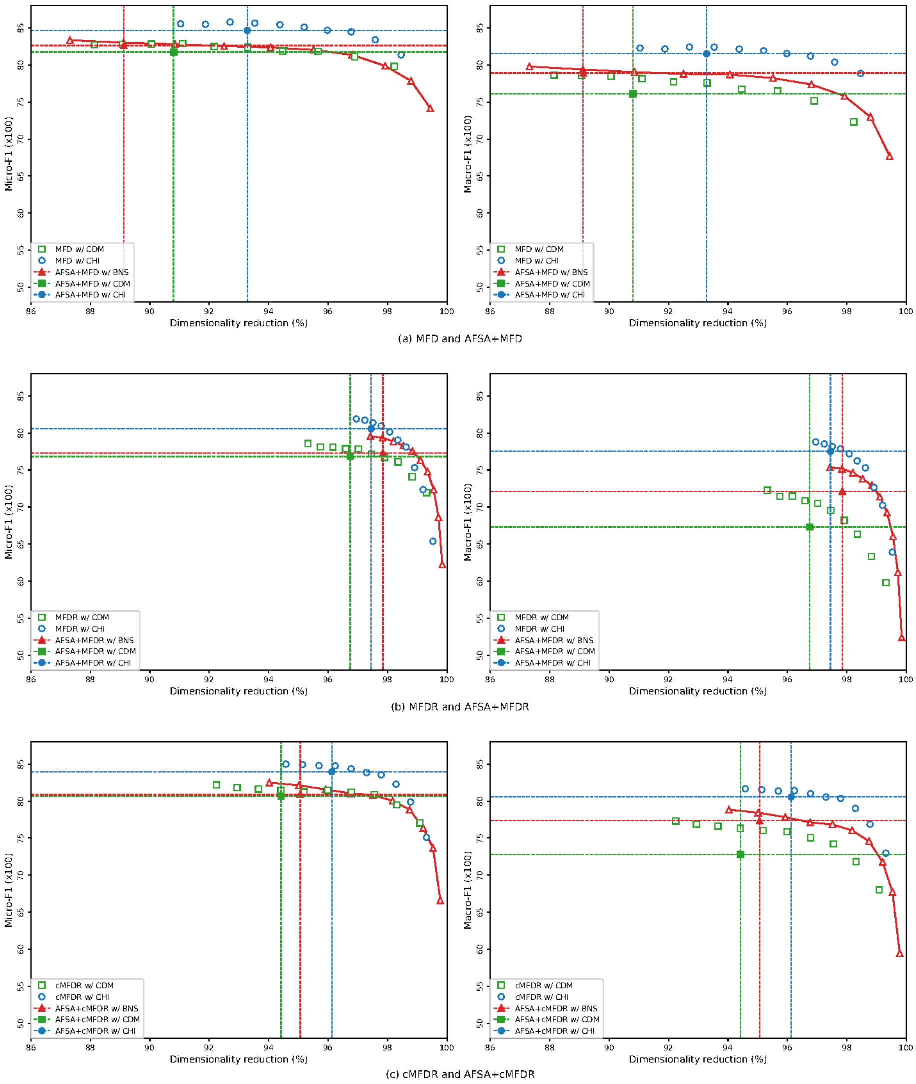

Figure 2: Scatter plot for Micro-F1 (a) and Macro-F1 (b) performances vs. dimensionality reduction (in %) for MFD and AFSA+MFD (a), MFDR and AFSA+MFDR (b) and cMFDR and AFSA+cMFDR (c).

example, only two results of cMFDR with CHI are observed in the upper right quadrant relatively to AFSA+cMFDR with CHI. Only for $f = 6$ and $f = 7$, cMFDR obtains a better relation between performance and DR. AFSA+MFDR and AFSA+cMFDR are the methods that present the best trade-offs between classification performance and DR. Both methods obtain, on average, 7.5 out of 10, a best trade-off than MFDR and cMFDR, respectively. AFSA+MFD obtains an average of 6.5 best trade-offs than MFD. Regarding the FEFs, BNS and CHI presents the best results.

MFD, MFDR and cMFDR require an additional effort to find the best value of the f parameter. This search is performed by analyzing values from 1 to 10 for f [Pinheiro et al. 2015, Fragoso et al. 2016b]. Thus, for each f value, four tasks are performed: feature evaluation function computation, feature selection, classifier training and classifier test. AFSA searches for the best f parameter configuration in an automatic fashion. With AFSA, FEF values are computed only once. Additionally, a validation set is used for f parameter configuration, meaning that AFSA uses less training data. Hence, it is expected that AFSA performs faster than the MFD, MFDR and cMFDR methods.

Table 3 shows the mean execution time for each feature selection method. In order to provide a fair comparison, for MFD, MFDR and cMFDR the execution times for f ranging from 1 to 10 are summed, since these methods require f parameter configuration. For AFSA, the method itself requires a range for f parameter (from 1 to n). In the experiments, this range is also 1 to 10 ($n = 10$). The execution time is measured as training time and test time. In this paper, training time is considered the time taken to compute FEF values, perform feature selection and train the classifier. AFSA uses a validation set to configure the best value for f parameter. Thus, for this method, training times also include time taken to validate the configuration. Test time is merely the time taken to test the classifier. Additionally, for benchmarking purposes, the comparison includes the execution time for Naïve Bayes, SVM and Random Forest classifiers on the original feature vectors. In this case, training time is simply the time taken to train the classifier, since there is no feature selection. Yet, the comparison includes the execution time for Multinomial Naïve Bayes and Random Forest classifiers on the features vectors extracted by LSI.

AFSA presents shorter execution times in comparison with the base methods, MFD, MFDR and cMFDR. For instance, while the total execution time for MFD with CDM is 472, for AFSA+MFD with CDM it is 97. AFSA presents shorter training times and test times than the base methods. Moreover, comparing AFSA with Naïve Bayes, SVM and Random Forest, using the original databases, it is noticed that there is a huge difference in execution times. AFSA+cMFDR and SVM outperforms Naïve Bayes for both Micro-F1 and Micro-F1. However, AFSA+cMFDR is 13 times faster than Naïve Bayes. In comparison with SVM,

Table 3: Execution times, in seconds, for training and testing. Total time is the sum of training and testing times of all databases.

Classifier	FS Method	Training			Test			Total		
		BNS	CDM	CHI	BNS	CDM	CHI	BNS	CDM	CHI
Naïve Bayes	MFD	66	427	123	38	45	26	104	472	149
	AFSA+MFD	77	94	51	7	3	5	84	97	56
	MFDR	13	390	140	6	25	12	19	415	152
	AFSA+MFDR	11	54	23	1	1	1	12	55	24
	cMFDR	32	422	111	16	36	18	48	458	129
	AFSA+cMFDR	34	75	37	3	3	2	37	78	39
	LSI	569			18			587		
	None	413			109			522		
SVM	None	2736			456			3192		
Random Forest	LSI	568			23			591		
	None	935			26			961		

AFSA+cMFDR presents comparable Micro-F1 and Macro-F1 scores. But, comparing execution times, AFSA+cMFDR is approximately 82 times faster than SVM. AFSA+cMFDR is 25 times faster than Random Forest and outperforms this classifier for both Micro-F1 and Macro-F1. The time taken to process LSI increased the execution time. Naïve Bayes with the original features is faster than using the features extracted by LSI. The same behavior is observed with Random Forest classifier. Furthermore, it is important to remark that the execution time computed for our methods include the evaluation of the best feature vector size while the choice for using 100 features for LSI required an effort, i.e., time for the evaluation of a set of numbers of features that is not computed in Table 3.

Figure 3 exhibits scatter plots showing the relationship between classification performance and execution time. The total execution times were considered in the graphs. Each figure, from (a) to (c), shows a scatter plot for Micro-F1 vs. execution time and Macro-F1 vs. execution time for MFD and AFSA+MFD, MFDR and AFSA+MFDR and cMFDR and AFSA+cMFDR, respectively. Again, for benchmarking purposes, the results for Naïve Bayes, SVM and Random Forest classifiers, using the original database, and Multinomial Naïve Bayes and Random Forest classifiers using the features vectors extracted by LSI are also shown.

For all FS methods, the best trade-offs between classification performance and execution time are achieved with CHI. In general, MFD, MFDR and cMFDR performances are slightly better than their combination with AFSA for both Micro-F1 and Macro-F1. For instance, MFD with CHI achieves Micro-F1 score

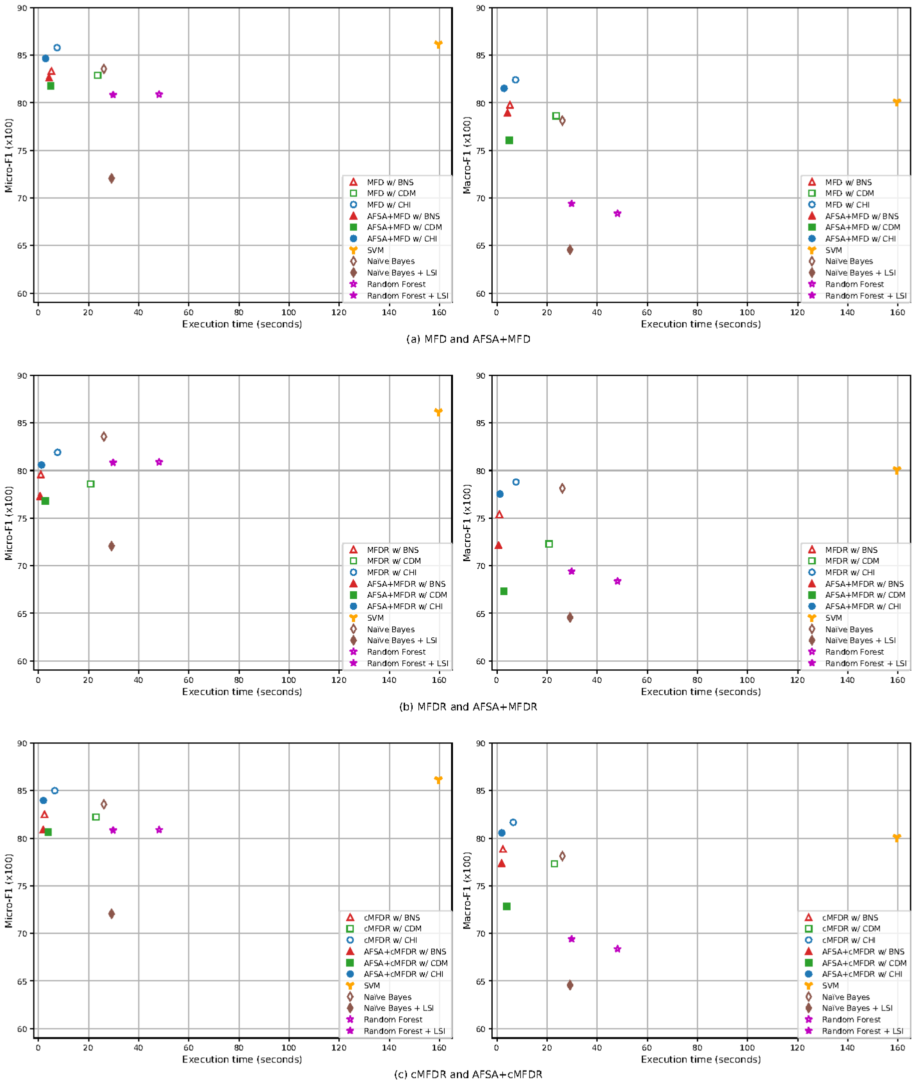

Figure 3: Scatter plot for Micro-F1 (a) and Macro-F1 (b) performances vs. execution time for MFD and AFSA+MFD (a), MFDR and AFSA+MFDR (b) and cMFDR and AFSA+cMFDR (c).

1.3% greater than AFSA+MFD with CHI. cMFDR with BNS performs 2% better for Micro-F1 than AFSA+cMFDR with BNS. However, f parameter configuration with AFSA is faster than MFD, MFDR and cMFDR. Using the previous examples, the execution time for MFD with CHI is 162% higher than AFSA+MFD with CHI. Execution time for cMFDR with BNS is 24% higher than cMFDR with BNS. Thus, AFSA achieves comparable classification performances than MFD, MFDR and cMFDR methods with shorter execution time.

Naïve Bayes Classifier using the original databases perform worse than MFD, AFSA+MFD, cMFDR and AFSA+cMFDR with CHI, in terms of Micro-F1. Concerning to Macro-F1, Naïve Bayes with the original databases is also surpassed by 9 out of 18 configurations of FS method and FEF. Naïve Bayes execution time is only comparable with MFD, MFDR and cMFDR using CDM. So, using the original database for Naïve Bayes classification is not considered a better approach than using MFD, AFSA+MFD, cMFDR or AFSA+cMFDR.

SVM, using the original database, achieves the best Micro-F1 results, but the execution time presented by SVM is very higher than any FS method. For example, MFD with CHI, a configuration that achieves comparable classification performance to SVM, is 21 times faster than SVM. AFSA+MFD with CHI is 57 times faster than SVM. Furthermore, regarding Macro-F1, MFD, AFSA+MFD, cMFDR and AFSA+cMFDR, all using CHI, overcome SVM score. Yet, MFD with BNS presents similar results to SVM. Hence, using SVM with the original databases is not a feasible approach, due to its execution time.

Random Forest using the original databases achieves lower performance than MFD, cMFDR, AFSA+MFD and AFSA+cMFDR, for Micro-F1 and Macro-F1. Additionally, the execution time presented by Random Forest is higher than MFD, MFDR, cMFDR, AFSA+MFDR, AFSA+cMFDR, AFSA+cMFDR and Naïve Bayes.

Random Forest and Naïve Bayes using LSI achieve the worst results for both Micro-F1 and Macro-F1. Furthermore, the execution time of these techniques are higher than most of the evaluated techniques, except for SVM. Hence, using LSI does not present advantages against our methods.

The feature evaluation function that produced the best classification results for each FS method was CHI. Table 4 shows the classification results for MFD, MFDR, cMFDR and each of these methods combined with AFSA, using CHI as FEF. Once again, the classification performance obtained by Naïve Bayes, SVM and Random Forest classifiers using the original databases, and Multinomial Naïve Bayes and Random Forest classifiers using the features vectors extracted by LSI are also exhibited, for benchmarking purposes. In addition, for each feature selection method result, the tables show the dimensionality reduction percentage.

The comparisons show that AFSA presents classification performance slightly below than MFD, MFDR and cMFDR, mostly for greater values for f. However, it is important to notice that, to achieve such results, MFD, MFDR and cMFDR

Table 4: Micro-F1 and Macro-F1 best average results (standard deviation in parenthesis) and dimensionality reduction percentage. Results achieved using Chi-Squared as feature evaluation function.

Classifier	DR method	Micro-F1	Reduct. %	Macro-F1	Reduct. %
Naïve Bayes	MFD	85.79 (0.84)	92.70	82.42 (1.39)	92.70
	AFSA+MFD	84.66 (1.16)	93.28	81.52 (1.17)	93.28
	MFDR	81.90 (1.72)	96.95	78.79 (1.84)	96.95
	AFSA+MFDR	80.59 (1.15)	97.45	77.53 (2.02)	97.45
	cMFDR	84.99 (0.98)	94.58	81.67 (1.38)	94.58
	AFSA+cMFDR	83.97 (1.01)	96.12	80.56 (1.84)	96.12
	LSI	72.06 (1.43)	96.54	64.56 (1.95)	96.54
	None	83.56 (0.67)	0	78.13 (1.87)	0
SVM	None	86.16 (0.76)	0	80.08 (2.19)	0
Random Forest	LSI	80.83 (1.14)	96.54	69.41 (1.56)	96.54
	None	80.89 (1.04)	0	68.37 (1.27)	0

require an effort for configuring a value for f parameter. This configuration is performed experimenting values from 1 to 10 for f [Pinheiro et al. 2015, Fragoso et al. 2016a], which implies in additional execution time. AFSA method performs this configuration in an automatic fashion. This approach leads to shorter execution times (Table 3).

The best AFSA results, concerning Micro-F1, Macro-F1 were obtained with AFSA+MFD, using the feature evaluation function CHI. This configuration performs better than all the other configurations for AFSA, MFD, MFDR and cMFDR. In addition, AFSA+MFD with CHI presents better Macro-F1 scores than Multinomial Naïve Bayes, SVM and Random Forest classifiers using the original databases, and Multinomial Naïve Bayes and Random Forest classifiers using the features vectors extracted by LSI. For Micro-F1, only SVM classifier presents a slightly better performance than AFSA+MFD with CHI.

Table 5 presents a comparison for execution time among AFSA+MFD with CHI (the best AFSA configuration), Naïve Bayes, SVM and Random Forest classifiers using the original databases, and Multinomial Naïve Bayes and Random Forest classifiers using the features vectors extracted by LSI.

AFSA presents the shortest execution times in 14 out of 20 databases. SVM presents the highest execution times for all the databases. SVM achieves the best Micro-F1 scores while AFSA+MFD obtains the best Macro-F1 scores and the shortest execution times. Additionally, AFSA+MFD Micro-F1 performance is comparable to SVM. Hence, AFSA is good approach for feature selection, since it determines the final feature vector size in a data-driven way, presents good classification performances, high dimensionality reduction rates and short

Table 5: Execution times using CHI. **x**$= 2,520$ seconds. The best value per database is in bold. NB and RF stand for Multinomial Naïve Bayes and Random Forest classifiers, respectively

Database	Execution time					
	AFSA+MFD	SVM	NB	NB+LSI	RF	RF + LSI
20Newsgroup	**1,312x**	84,757x	19,183x	19,831x	33,338x	19,137x
CSTR	1.6x	71.7x	**1x**	5.2x	16.4x	20.5x
Oh0	10.9x	284.9x	**6.5x**	15.5x	28.7x	42.5x
Oh5	9.0x	227.1x	**5.7x**	13.4x	25.9x	33.6x
Oh10	**10x**	598.1x	13.7x	15.2x	29.2x	37.7x
Oh15	8.4x	262.8x	**5.8x**	13.1x	25.1x	33.9x
Reuters0	16.7x	223.8x	**9.3x**	17.7x	34.0x	43.7x
Reuters1	37.4x	473.7x	**17.0x**	44.2x	50.4x	57.3x
Reuters10	**166.5x**	6,127x	203.4x	358.1x	1,017x	548x
Review-Polarity	**16.2x**	2,919x	47.3x	114.8x	179.3x	143.3x
Syskill Webert	**1.9x**	30.6x	2.6x	10.1x	16.7x	26.6x
TDT2-TOP30	**546.6x**	18,725x	954.4x	2,635x	2,990x	2,825x
Tr11	**3.6x**	460.5x	11.2x	17.5x	17.2x	33.9x
Tr12	**2.3x**	140.1x	3.8x	13.4x	16.8x	29.9x
Tr21	**3.0x**	219.3x	5.1x	19.6x	16.8x	35.9x
Tr23	**1.5x**	92.0x	2.5x	11.3x	16.5x	27.4x
Tr41	**10.9x**	913.7x	26.3x	29.7x	29.8x	50.4x
Tr45	**6.7x**	851.3x	23.3x	30.7x	25.4x	47.7x
WAP	**35.8x**	4,391.8x	64.6x	49.7x	66.8x	78.4x
WebKB	**45.4x**	4,889.3x	114.7x	105.2x	236.9x	171x
Wilcoxon p-value	n.a.	$3.5 * 10^{-14}$	$6.3 * 10^{-5}$	$5.9 * 10^{-6}$	$1.7 * 10^{-8}$	$4.6 * 10^{-6}$

execution time. It is worth to note that AFSA presents shorter execution times than Naïve Bayes, SVM and Random Forest for large databases, a common characteristic in text classification problems. For the 20 Newsgroup database, AFSA presents execution time 15 times faster than Naïve Bayes, 64 times faster than SVM and 25 times faster than Random Forest. Multinomial Naïve Bayes and Random Forest classifiers using LSI presented higher execution time than AFSA+MFD for all databases.

4.3.1 Statistical tests

In order to verify if the differences between the average performances obtained by the methods presented in this paper are significant, statistical tests were executed. As the reliability of parametric tests may not be assured due to the small samples size, Wilcoxon signed-rank test [Benavoli et al. 2016] was adopted. This is considered a robust option for non-parametric pairwise statistical test [Demšar 2006].

Wilcoxon signed-rank test was applied on the performance obtained by each method in the test set on a 10-fold stratified cross-validation experiment. Each method configuration was tested against all the others methods configurations. We test six feature selection methods: MFD, MFDR, cMFDR, AFSA+MFD, AFSA+MFDR and AFSA+cMFDR. For MFD, MFDR and cMFDR methods, we test 10 values for f parameter and three FEFs, what gives us 30 configurations for each of these methods. For AFSA, we test three FEFs by method. Finally, we test each method configuration against SVM and Multinomial Naïve Bayes classifiers using the original databases. Thus, each method configuration is tested against 98 methods configuration plus SVM and Naïve Bayes, for each database. As we use 20 databases in this paper, we have 2000 Wilcoxon test results for each test configuration. Table 6 summarizes these results for Micro-F1 and Macro-F1, where "$\gg$" and "$\ll$" mean that one method performs better or worse, respectively, than another with $\alpha \leq 0.01$ significance level. "$>$" and "$<$" represent a better or worse performance, respectively, of a method over another, with $0.01 < \alpha \leq 0.05$. Cases in which $\alpha > 0.05$ are represented by "$\sim$".

Table 6: Wilcoxon statistical test results. $\gg$ and $\ll$ mean strong statistical evidence that one method performs, respectively, better or worse than another. $>$ and $<$ mean statistical evidence that one method performs, respectively, better or worse than another and $\sim$ means that there is no statistical evidence of differences between performances.

	Micro-F1						Macro-F1					
FS method	FEF	$\gg$	$>$	$\sim$	$<$	$\ll$	FEF	$\gg$	$>$	$\sim$	$<$	$\ll$
AFSA+MFD	BNS	526	163	1045	84	182	BNS	481	163	1213	66	77
	CDM	500	124	1117	82	177	CDM	415	107	1130	104	244
	CHI	625	203	1021	65	86	CHI	652	201	1082	19	46
AFSA+MFDR	BNS	215	51	976	129	629	BNS	193	67	963	149	628
	CDM	251	93	902	104	650	CDM	166	62	921	107	744
	CHI	360	106	1002	149	383	CHI	446	116	1091	144	203
AFSA+cMFDR	BNS	383	107	1060	81	369	BNS	407	114	1175	128	176
	CDM	446	117	1045	86	306	CDM	352	105	969	129	445
	CHI	563	148	1033	50	206	CHI	579	153	1142	39	87

For both Micro-F1 and Macro-F1, AFSA+MFD with CHI presents the best results. This method achieves better classification results 828 times, out of 2000, for Micro-F1 and 853, for Macro-F1. In only 151 times for Micro-F1 and 65 for Macro-F1, this method presents worse performance than the method it is compared with. AFSA+cMFDR, with CHI, also achieves remarkable results. This method shows superior performance in 771 cases for Micro-F1 and 732 cases for Macro-F1. It presented a worse performance in just 256 cases for Micro-F1 and 126 cases for Macro-F1. AFSA+MFDR, with BNS and CDM, presents the worst performances, for both Micro-F1 and Macro-F1.

5 Conclusion

ALOFT, MFD, MFDR and cMFDR are feature selection methods designed to alleviate the search for the best feature vector size. These methods select f features per document, unlike the traditional FS methods that select a number m of features per Corpus. Although MFD, MFDR and cMFDR require a parameter f that defines the number of features to be selected per document, a search for the optimal value for f is less costly than the search for the best value for m. The auxiliary method AFSA is used to automate the configuration of the f parameter in MFD, MFDR and cMFDR. AFSA uses a validation set to define the best value for f in a data driven way.

This paper performed an extensive comparison among ALOFT, MFD, MFDR, cMFDR. In addition, experiments were conducted to determine if AFSA is beneficial. In previous studies, only cMFDR method was used in combination with AFSA [Fragoso et al. 2016b]. In this paper, AFSA was compared with MFD, MFDR and cMFDR. The comparison was performed using f between 1 and 10, for these methods. AFSA searched for the best value for f in the same range for each of the former methods. Furthermore, for benchmarking purposes, we also compared ALOFT, MFD, MFDR, cMFDR and AFSA with Multinomial Naïve Bayes, SVM and Random Forest classifiers using the databases without dimensionality reduction and Multinomial Naïve Bayes and Random Forest using LSI. Experiments were carried out to assess the classification performance, dimensionality reduction and execution time of the methods using 20 databases and 3 FEFs.

Experimental results showed that MFD with CHI achieved the best results among the presented FS methods. This configuration obtained the best Micro-F1 and Macro-F1 scores, 85.79% and 82.42%, respectively. The dimensionality reduction rate of MFD was lower than MFDR and cMFDR, however the execution time of MFD was similar to those methods.

The results also demonstrated that AFSA does not significantly affect the classification performance and dimensionality reduction rate of the FS methods (MFD, MFDR and cMFDR). Additionally, AFSA presented shorter execution time than these methods. The best AFSA configuration was AFSA+MFD with CHI. With this configuration, AFSA overcame all the other configurations of AFSA, for both Micro-F1 and Macro-F1.

The experiments indicated that, in an environment with time and computational resources constraints, MFD and AFSA+MFD are better options than Multinomial Naïve Bayes, SVM and Random Forest using the databases without dimensionality reduction and Multinomial Naïve Bayes and Random Forest using LSI. Using CHI, MFD and AFSA+MFD, presented superior Macro-F1 performance than Multinomial Naïve Bayes, SVM and Random Forest using the databases without dimensionality reduction. Regarding Micro-F1, MFD and AFSA+MFD overcame Multinomial Naïve Bayes and Random Forest classifiers and presented a comparable performance to SVM. It is important to note that MFD and AFSA+MFD achieved such results using less than 10% of the original features. In addition, their execution times were shorter than Multinomial Naïve Bayes, SVM and Random Forest, specially for the bigger databases. If new documents are constantly added to the database, AFSA may be a better option than MFD. In this case, it is necessary to re-execute the feature selection process, in order to embrace the new documents. AFSA can automatically determine the best feature subset, in contrast with MFD that requires human interaction.

References

[Benavoli et al. 2016] Benavoli, A., Corani, G., Mangili, F.: Should we really use post-hoc tests based on mean-ranks; *The Journal of Machine Learning Research*, 17: 1–10, 2016.

[Chen et al. 2009] Chen, J., Huang, H., Tian, S., Qu, Y.: Feature selection for text classification with Naive Bayes; *Expert Systems with Applications*, 36: 5432–5435, 2009.

[Demšar 2006] Demšar, J.: Statistical comparisons of classifiers over multiple data sets; *The Journal of Machine Learning Research*, 7: 1–30, 2006.

[Forman 2003] Forman, G.: An extensive empirical study of feature selection metrics for text classification; *The Journal of Machine Learning Research*, 3: 1289–1305, 2003.

[Fragoso et al. 2016a] Fragoso, R.C.P., Pinheiro, R.H.W., Cavalcanti, G.D.C.: Class-dependent feature selection algorithm for text categorization; In *International Joint Conference on Neural Networks*, IEEE, 3508–3515, 2016.

[Fragoso et al. 2016b] Fragoso, R.C.P., Pinheiro, R.H.W., Cavalcanti, G.D.C.: A method for automatic determination of the feature vector size for text categorization; In *5th Brazilian Conference on Intelligent Systems*, IEEE, 259–264, 2016.

[Gabrilovich, Markovitch 2004] Gabrilovich, E., Markovitch, S.: Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4. 5; In *21th International Conference on Machine learning*, ACM, 41, 2004.

[Ghareb et al. 2016] Ghareb, A.S., Bakar, A.A., Hamdan, A.R.: Hybrid feature selection based on enhanced genetic algorithm for text categorization; *Expert Systems with Applications*, 49: 31–47, 2016.

[Guru et al. 2018] Guru, D.S., Suhil, M., Raju, L. N., Kumar, V.: An Alternative Framework for Univariate Filter based Feature Selection for Text Categorization; *Pattern Recognition Letters*, 103: 23–31, 2018.

[Guyon, Elisseeff 2003] Guyon, I., Elisseeff, A.: An introduction to variable and feature selection; *The Journal of Machine Learning Research*, 3: 1157–1182, 2003.

[Leopold, Kindermann 2002] Leopold, E., Kindermann, J.: Text categorization with support vector machines. How to represent texts in input space?; *Machine Learning*, 46: 423–444, 2002.

[Lewis, Ringuette 1994] Lewis, D.D., Ringuette, M.: A comparison of two learning algorithms for text categorization; In *3rd Annual Symposium on Document Analysis and Information Retrieval*, 81–93, 1994.

[Ogura et al. 2011] Ogura, H., Amano, H., Kondo, M.: Comparison of metrics for feature selection in imbalanced text classification; *Expert Systems with Applications*, 38: 4978–4989, 2011.

[Pinheiro et al. 2012] Pinheiro, R.H.W., Cavalcanti, G.D.C., Correa, R.F., Ren, T.I.: A global-ranking local feature selection method for text categorization; *Expert Systems with Applications*, 39: 12851–12857, 2012.

[Pinheiro et al. 2015] Pinheiro, R.H.W., Cavalcanti, G.D.C., Ren, T.I.: Data-driven global-ranking local feature selection methods for text categorization; *Expert Systems with Applications*, 42: 1941–1949, 2015.

[Sebastiani 2002] Sebastiani, F.: Machine learning in automated text categorization; *ACM Computing Surveys*, 34: 1–47, 2002.

[Sun et al. 2013] Sun, X., Liu, Y., Xu, M., Chen, H., Han, J., Wang, K.: Feature selection using dynamic weights for classification; *Knowledge-Based Systems*, 37: 541–549, 2013.

[Tang et al. 2016] Tang, B., He, H., Baggenstoss, P.M., Kay, S.: A Bayesian classification approach using class-specific features for text categorization; *IEEE Transactions on Knowledge and Data Engineering*, 28: 1602–1606, 2016.

[Uğuz 2011] Uğuz, H.: A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm; *Knowledge-Based Systems*, 24: 1024–1032, 2011.

[Uysal, Gunal 2012] Uysal, A.K., Gunal, S.: A novel probabilistic feature selection method for text classification; *Knowledge-Based Systems*, 36: 226–235, 2012.

[Uysal 2016] Uysal, A.: An improved global feature selection scheme for text classification; *Expert systems with Applications*, 43: 82–92, 2016.

[Yang, Pedersen 1997] Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization; In *International Conference on Machine Learning*, 412–420, 1997.

[Yu, Liu 2003] Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution; In *International Conference on Machine Leaning*, 856–863, 2003.

[Biau, Scornet 2016] Biau, G., Scornet, E.: A random forest guided tour; In *Test*, 25: 197–227, 2016.

[Deerwester et al. 1990] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis; In *Journal of the American society for information science*, 41(6), 391–407, 1990.