Applied Soft Computing Journal 76 (2019) 564–574

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Combining binary classifiers in different dichotomy spaces for text categorization

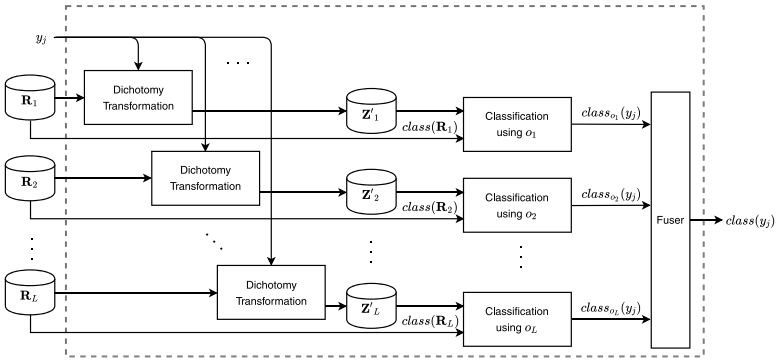
Roberto H.W. Pinheiro [a,*], George D.C. Cavalcanti [b], Tsang Ing Ren [b]

[a] Centro de Ciências e Tecnologia, Universidade Federal do Cariri, Ceará, Brazil
[b] Centro de Informática, Universidade Federal de Pernambuco, Pernambuco, Brazil

HIGHLIGHTS

- Combined Dichotomy Transformations (CoDiT) method is proposed.
- CoDiT is a multiple classifier system for text categorization.
- CoDiT uses Dichotomy Transformation to transform a multi-class problem.
- Each classifier in CoDiT is trained using a different transformed set.
- CoDiT obtains better results than literature multi-classifier systems.

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 23 February 2017
Received in revised form 7 December 2018
Accepted 14 December 2018
Available online 7 January 2019

Keywords:
Text categorization
Multiple classifier system
Dichotomy transformation

ABSTRACT

Several supervised machine learning applications are commonly represented as multi-class problems, but it is harder to distinguish several classes rather than just two classes. In contrast to the approaches one-against-all and all-pairs that transform a multi-class problem into a set of binary problems, Dichotomy Transformation (DT) converts a multi-class problem into a different problem where the goal is to verify if a pair of documents belongs to the same class or not. To perform this task, DT generates a dichotomy set obtained by combining a pair of documents, each belongs to either a positive class (documents in the pair that have the same class) or a negative class (documents in the pair that come from different classes). The definition of this dichotomy set plays an important role in the overall accuracy of the system. So, an alternative to avoid searching for the best dichotomy set is using multiple classifier systems because we can have many different sets where each one is used to train one binary classifier instead of having only one dichotomy set. Herein we propose Combined Dichotomy Transformations (CoDiT), a Text Categorization system that combines binary classifiers that are trained with different dichotomy sets using DT. By using DT, the number of training examples increases exponentially when compared with the original training set. This is a desirable property because each classifier can be trained with different data without reducing the number of examples or features. Therefore, it is possible to compose an ensemble with diverse and strong classifiers. Experiments using 14 databases show that CoDiT achieves statistically better results in comparison to SVM, Bagging, Random Subspace, BoosTexter, and Random Forest.

1. Introduction

Several supervised machine learning applications are represented as multi-class problems, for example, in digit recognition the objective is to determine the digit (labels from 0, 1, . . . , 9) in an

* Corresponding author.
E-mail addresses: roberto.hugo@ufca.edu.br (R.H.W. Pinheiro), gdcc@cin.ufpe.br (G.D.C. Cavalcanti), tir@cin.ufpe.br (Tsang I.R.).

https://doi.org/10.1016/j.asoc.2018.12.023
1568-4946/

image. However, it is usually harder to distinguish several classes instead of just two classes [1]. Some machine learning algorithms have been proposed to solve binary (two classes) problems, such as Perceptron [2], Support Vector Machine [3], Adaboost [4], among others [5]. When these algorithms are used to solve multi-class problems, some modifications are necessary or multiple versions of the binary algorithms are required. The two most popular approaches for reducing a multi-class problem into a set of binary problems are one-against-all [6] and all-pairs [7].

In contrast to these approaches, Dichotomy Transformation (DT) [8] transforms the multi-class problem into a different problem where the goal is to decide if two documents are from the same class. For instance, Cha and Srihari [8] used DT to transform a handwriting identification multi-class problem (several authors) into a single binary problem (authorship or non-authorship). In this manner, instead of matching handwriting with one of the available authors, a pair of handwriting is classified as authorship or non-authorship. DT has been used not only in handwriting identification [9–11] but also in voice verification [12], iris biometric [13], and keystroke biometric [14].

Given the original multi-class training data, DT generates a new training data called dichotomy set that is used to train the binary classifier. The dichotomy set is generated using two input sets. In this manner, each new example in the dichotomy set is a combination of a pair of documents, one for each input set, and belongs to either a positive class (where the pair of documents have the same class) or a negative class (where the pair of documents have different classes). So, a classifier trained using this dichotomy set is able to classify an example into positive or negative.

However, defining the two input sets necessary to perform DT is not a trivial task. Moreover, the dichotomy set obtained from DT is directly responsible for the performance of the system. An alternative to avoid searching for the best dichotomy set is to use multiple classifier systems (MCS) because instead of having a single dichotomy set we can have many different input sets which generates different dichotomy sets that are used to train many binary classifiers. So, DT and MCS can mutually benefit to generate a strong and diverse ensemble.

In this paper, we propose an approach for Text Categorization that combines binary classifiers trained with different dichotomy sets called Combined Dichotomy Transformations (CoDiT). Thus, each classifier of the ensemble is trained on a different space that is computed using DT. This procedure increases the diversity of the ensemble that plays a major role in multiple classifier systems [15]. The majority of ensemble methods tries to achieve diversity by randomly segmenting features and/or examples of the original training set [16], such segmentations are present in Bagging [17] and Random Subspace [18]. In a similar way, CoDiT also achieves diversity by segmenting examples.

By using DT in CoDiT, the number of examples used for training increases exponentially, which makes it possible to obtain several classifiers in different dichotomy spaces without reducing the amount of training data. This is a desirable property because it is possible for the ensemble to be diverse, since each classifier is trained with different data. Moreover, the base classifier of CoDiT is Support Vector Machine (SVM), considered to be the state-of-the-art [19–22] of Text Categorization. Since SVM performs better in binary problems [10] and CoDiT is a combination of binary problems, it benefits from the best classifier in an environment that provides its best performance.

Through a set of experiments on 14 databases, we show that CoDiT obtains superior performance when compared with SVM, Bagging, Random Subspace, BoosTexter and Random Forest, using macro-F1 and micro-F1. Macro-F1 results of CoDiT deserve particular attention because they were significantly better than other approaches from the literature which indicates that CoDiT strongly benefits from the use of binary classifiers to avoid bias towards unbalanced classes.

The remainder of this paper is organized as follows: the next section shows how to transform a multi-class problem into a two class problem using the Dichotomy Transformation. Section 3 presents the proposed method CoDiT, a Text Categorization approach that combines binary classifiers in the dichotomy space. Section 4 describes the experimental methodology, reports and analyzes the results. Section 5 concludes the paper and presents some future works.

2. Dichotomy transformation

The Dichotomy Transformation (DT) [8] converts a multi-class problem into a different problem aiming to determine if two patterns belong to the same class. This section describes how this transformation is applied to the problem of Text Categorization.

The required inputs of DT are two sets of documents, and the output is a new set of documents with two possible classes: positive (class •) or negative (class ○). Let us assume the following input example: set **A** and set **B**. Each document $a_i \in \mathbf{A} = \{a_1, a_2, \ldots, a_A\}$ is composed of its feature vector $w^{a_i} = [w_1^{a_i}, w_2^{a_i}, \ldots, w_V^{a_i}]^{\mathrm{T}}$ with V features using Bag-of-Words[1] and the document class is defined as $class(a_i) \in \mathbf{C} = \{1, 2, \ldots, C\}$. Whereas, each document $b_j \in \mathbf{B} = \{b_1, b_2, \ldots, b_B\}$ uses the same Bag-of-Words approach and has the feature vector defined as $w^{b_j} = [w_1^{b_j}, w_2^{b_j}, \ldots, w_V^{b_j}]^{\mathrm{T}}$. The output of DT is the dichotomy set **Z**.

The dichotomy set **Z** is composed of $A \times B$ documents, one for each pair $\langle a_i, b_j \rangle$ of the documents in **A** and **B**, where each document pair is considered either positive ($class(a_i) = class(b_j)$) or negative ($class(a_i) \neq class(b_j)$). Each dichotomy document $z_{i,j} \in \mathbf{Z}$ is obtained by calculating $DT(a_i, b_j)$ for each pair $a_i \in \mathbf{A}$ and $b_j \in \mathbf{B}$. The most common way to calculate this function uses the absolute value of the difference:

$$DT(a_i, b_j) = \begin{bmatrix} |w_1^{a_i} - w_1^{b_j}| \\ |w_2^{a_i} - w_2^{b_j}| \\ \vdots \\ |w_V^{a_i} - w_V^{b_j}| \end{bmatrix} \quad (1)$$

where $w_h^{a_i}$ and $w_h^{b_j}$ are the hth features of the documents a_i and b_j respectively.

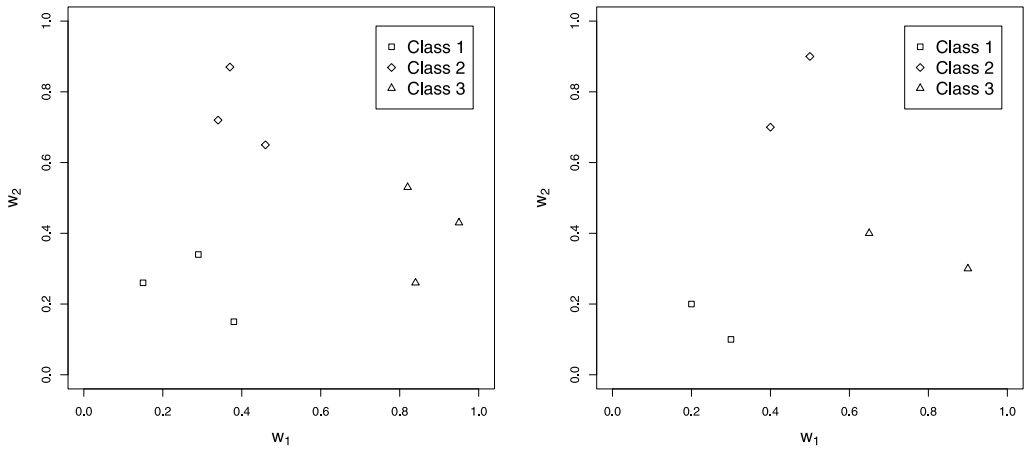
Fig. 1 illustrates an example of the Dichotomy Transformation that uses two input sets: set **A** (Fig. 1(a)) has 9 documents ($A = 9$), and set **B** (Fig. 1(b)) has 6 documents ($B = 6$), both having 3 classes ($C = 3$). Fig. 1(c) presents the resulting dichotomy set **Z** that has 54 documents ($A \times B$); this approach has an interesting property since it increases the number of available documents. The documents in **Z** are divided into two classes: positive (•), close to the origin, with 18 documents, obtained from pairs with the same class; and negative (○), far from the origin, with 36 documents, obtained from pairs with different classes. For simplicity, all sets are represented with 2 features (V) showed in x-axis (w_1) and in y-axis (w_2).

A classifier trained using the dichotomy set **Z** can be used to verify if a pair of documents belongs to the same class (positive) or not (negative).

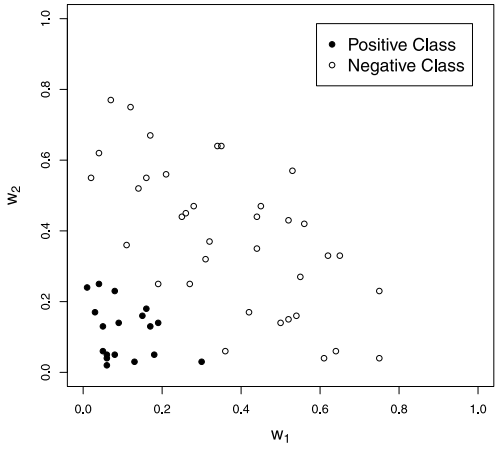
3. Combined dichotomy transformations

Combined Dichotomy Transformations (CoDiT) is a Text Categorization system that combines binary classifiers in the dichotomy space. Following, Section 3.1 describes the notation of CoDiT. In

[1] Bag-of-Words is the most used feature vector representation for Text Categorization, where each word of a given database is considered a feature.



(a) Input set **A** (b) Input set **B**



(c) Dichotomy set **Z**

Fig. 1. Example of Dichotomy Transformation with inputs sets **A** (top-left) and **B** (top-right), and dichotomy set **Z** (bottom). All sets are represented by two features (w_1 and w_2).

Section 3.2 the training phase of CoDiT is evaluated using L classifiers o_s and in Section 3.3 the system is evaluated using a test set **Y**. A toy example, showing how CoDiT operates is described in Section 3.4.

3.1. Notation

CoDiT uses several different sets: input sets, output sets and sets generated during the training phase. The training set **X** is composed of N documents $x_i \in \mathbf{X} = \{x_1, x_2, \ldots, x_N\}$, each represented by the pair $\langle w^{x_i}, class(x_i)\rangle$, where $w^{x_i} = [w_1^{x_i}, w_2^{x_i}, \ldots, w_V^{x_i}]^{\mathrm{T}}$ is the vector of terms using the Bag-of-Words and $class(x_i) \in \mathbf{C} = \{1, 2, \ldots, C\}$ is the class of the document. The test set **Y** is composed of M documents $y_j \in \mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$ and uses the same Bag-of-Words representation as the training set. The subsets $\mathbf{X}_s$ are subsets of the training set **X**, each training subset $\mathbf{X}_s$ is composed of N' documents ($N' \leq N$). The representation sets $\mathbf{R}_s$ are composed of Q documents $p_k \in \mathbf{R}_s = \{p_1, p_2, \ldots, p_Q\}$. Finally, the dichotomy sets $\mathbf{Z}_s$ are composed of $N' \times Q$ documents generated by the Dichotomy Transformation using $\mathbf{X}_s$ and $\mathbf{R}_s$; and the dichotomy sets $\mathbf{Z}'_s$ are composed of Q documents generated by the Dichotomy Transformation using y_j and $\mathbf{R}_s$.

3.2. Training phase

Fig. 2 shows a flowchart of the training phase. In this phase, the inputs are the training set **X**, the number of classifiers in the ensemble (L), the training subsets size (N') and the representation sets size (Q); and the outputs are L classifiers. Four different modules are used in this phase: Subsets Generation, Representation Sets Generation, Dichotomy Transformation and Classifier Training.

The Subsets Generation module receives the training set **X** as input, and generates L training subsets ($\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$) as output, each with N' documents ($N' < N$). This module uses random under-sampling that is a random sampling without replacement of N' documents for each training subset. This approach is similar to Bagging [17], a diversity creation method that varies the set of accessible hypotheses [16].

The Representation Sets Generation module receives the training set **X**, the ensemble size (L), and the representation set size (Q) as inputs, and generates L representations sets ($\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$) each with Q documents. Several approaches can be used to define which documents are selected for the representation set. For instance, the representation set can be obtained using the whole training set, however this procedure is time consuming and unnecessary, because smaller sets achieve similar or better

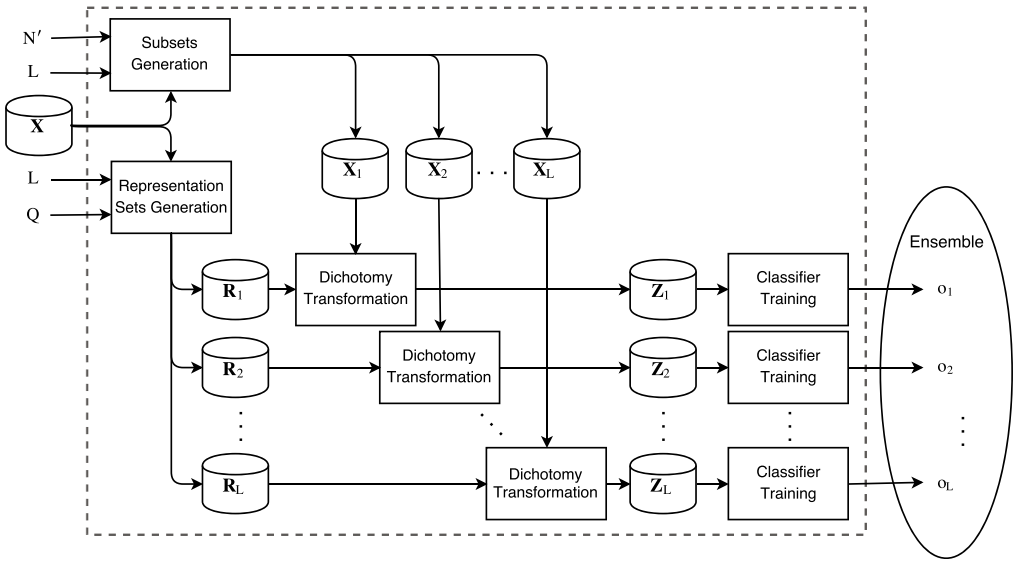


Fig. 2. Training phase of CoDiT. $\mathbf{X}$ is the training set, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$ are subsets of the training set, $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$ are the representation sets, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_L$ are the dichotomy sets, L is the size of the ensemble, N' is the number of documents in the training subsets, Q is the number of documents in the representation set and $o_1, o_2, \ldots, o_L$ are the classifiers of the ensemble.

performance [23]. Other alternatives generate a representation set smaller than the original training set, such as: selection of the most dissimilar documents [23], prototype selection algorithms, such as condensed nearest neighbor, edition nearest neighbor, among others [24,25], and selection of random documents. CoDiT uses a random selection by class. This process may not appear to be coherent, however, previous work [26] shows that Representation Sets obtained with random prototype selection can achieve good results. Random selection by class is a sampling without replacement, i.e., no document is selected two times; with the restriction of having the same number of documents in each class. This restriction is necessary to avoid bias towards any class during the test phase. Therefore, Q documents are selected to compose each Representation Set and each class has Q/C documents.

The Dichotomy Transformation follows the same steps described in Section 2, but in the training phase the inputs are a training subset $\mathbf{X}_s$ and a representation set $\mathbf{R}_s$, where $s = \{1, 2, \ldots, L\}$. Due to the definition of CoDiT, some characteristics are expected from the input sets ($\mathbf{X}_s$ and $\mathbf{R}_s$): (i) since the Dichotomy Transformation module increases the number of documents available for training, the input sets should have a small number of documents to avoid an unnecessary increase in the computational cost; (ii) CoDiT is a multiple classifier system and, in such systems, diversity plays an important role [27], thus it is beneficial for the system for all input subsets of Dichotomy Transformation ($\mathbf{X}_s$ and $\mathbf{R}_s$) have different combinations of documents to increase the diversity of the ensemble; (iii) due to CoDiT's classification rule (explained in Section 3.3), the representation sets must have the same number of documents per class. The first characteristic is addressed setting the value of N' and Q as low as possible (Section 4 presents our choice); the second characteristic is addressed using random selection to generate both sets of the Dichotomy Transformation, which is a powerful way to generate diversity [28,29], additionally, the low values of N' and Q facilitates the generation of different sets; and the third characteristic is addressed by the Representation Set Generation module, which provides equal distribution of documents among the classes.

So, given a training subset $\mathbf{X}_s$ and a representation set $\mathbf{R}_s$, the Dichotomy Transformation generates the dichotomy set $\mathbf{Z}_s$:

$$\mathbf{Z}_s = \left\{ \begin{matrix} DT(x_1, p_1), & DT(x_1, p_2), & \ldots, & DT(x_1, p_Q), \\ DT(x_2, p_1), & DT(x_2, p_2), & \ldots, & DT(x_2, p_Q), \\ \vdots & \vdots & \ddots & \vdots \\ DT(x_{N'}, p_1), & DT(x_{N'}, p_2), & \ldots, & DT(x_{N'}, p_Q) \end{matrix} \right\}, \tag{2}$$

where each function $DT(x_i, p_k)$ generates a new document as described in Eq. (1). In this manner, each dichotomy set $\mathbf{Z}_s$ has $N' \times Q$ documents, where each document belongs to either positive class $\bullet$ ($class(x_i) = class(p_k)$) or negative class $\circ$ ($class(x_i) \neq class(p_k)$). By substituting Eq. (1) into Eq. (2), we obtain:

$$\mathbf{Z}_s = \left\{ \begin{matrix} \begin{bmatrix} |w_1^{x_1} - w_1^{p_1}| \\ |w_2^{x_1} - w_2^{p_1}| \\ \vdots \\ |w_V^{x_1} - w_V^{p_1}| \end{bmatrix}, & \begin{bmatrix} |w_1^{x_1} - w_1^{p_2}| \\ |w_2^{x_1} - w_2^{p_2}| \\ \vdots \\ |w_V^{x_1} - w_V^{p_2}| \end{bmatrix}, & \ldots, & \begin{bmatrix} |w_1^{x_1} - w_1^{p_Q}| \\ |w_2^{x_1} - w_2^{p_Q}| \\ \vdots \\ |w_V^{x_1} - w_V^{p_Q}| \end{bmatrix}, \\ \begin{bmatrix} |w_1^{x_2} - w_1^{p_1}| \\ |w_2^{x_2} - w_2^{p_1}| \\ \vdots \\ |w_V^{x_2} - w_V^{p_1}| \end{bmatrix}, & \begin{bmatrix} |w_1^{x_2} - w_1^{p_2}| \\ |w_2^{x_2} - w_2^{p_2}| \\ \vdots \\ |w_V^{x_2} - w_V^{p_2}| \end{bmatrix}, & \ldots, & \begin{bmatrix} |w_1^{x_2} - w_1^{p_Q}| \\ |w_2^{x_2} - w_2^{p_Q}| \\ \vdots \\ |w_V^{x_2} - w_V^{p_Q}| \end{bmatrix}, \\ \vdots & \vdots & \ddots & \vdots \\ \begin{bmatrix} |w_1^{x_{N'}} - w_1^{p_1}| \\ |w_2^{x_{N'}} - w_2^{p_1}| \\ \vdots \\ |w_V^{x_{N'}} - w_V^{p_1}| \end{bmatrix}, & \begin{bmatrix} |w_1^{x_{N'}} - w_1^{p_2}| \\ |w_2^{x_{N'}} - w_2^{p_2}| \\ \vdots \\ |w_V^{x_{N'}} - w_V^{p_2}| \end{bmatrix}, & \ldots, & \begin{bmatrix} |w_1^{x_{N'}} - w_1^{p_Q}| \\ |w_2^{x_{N'}} - w_2^{p_Q}| \\ \vdots \\ |w_V^{x_{N'}} - w_V^{p_Q}| \end{bmatrix} \end{matrix} \right\} \tag{3}$$

where $w_h^{x_i}$ and $w_h^{p_k}$ are the hth features of documents x_i and p_k respectively, N' is the number of documents in the training subset $\mathbf{X}_s$, Q is the number of documents in the representation set $\mathbf{R}_s$ and V is the number of features.

Finally, each one of the L dichotomy sets $\mathbf{Z}_s$ is presented to its respective Classifier Training module to train the classifier o_s. The classifier o_s is binary, since the problem now has only two classes, regardless of the original number of classes (C). In this

way, the final output of the training phase are L trained classifiers $(o_1, o_2, \ldots, o_L)$.

3.3. Test phase

Fig. 3 shows a flowchart of the test phase. During this phase, each document from the test set $y_j \in \mathbf{Y}$ is presented, one at a time, for classification. The test phase also receives, as inputs, L representation sets $(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L)$ and L classifiers; and, as output, a response $class(y_j)$ for each document of the test set. This phase is composed of three modules: Dichotomy Transformation, Classification, and Fuser.

The Dichotomy Transformation module receives a single document $y_j \in \mathbf{Y}$ and a representation set $\mathbf{R}_s$, obtained during the training set as inputs, and generates the dichotomy set $\mathbf{Z}'_s$:

$$\mathbf{Z}'_s = \left\{z'_{y,1}, z'_{y,2}, \ldots, z'_{y,Q}\right\} \tag{4}$$

$$= \left\{DT(y_j, p_1), DT(y_j, p_2), \ldots, DT(y_j, p_Q)\right\}, \tag{5}$$

where the dichotomy document $z'_{j,k}$ is obtained with the function $DT(y_j, p_k)$ as showed in Eq. (1), having $p_k \in \mathbf{R}_s$. In this manner, each Dichotomy Transformation generates Q documents. By substituting Eq. (1) into Eq. (5), we obtain:

$$\mathbf{Z}'_s = \left\{ \begin{bmatrix} |w_1^{y_j} - w_1^{p_1}| \\ |w_2^{y_j} - w_2^{p_1}| \\ \vdots \\ |w_V^{y_j} - w_V^{p_1}| \end{bmatrix}, \begin{bmatrix} |w_1^{y_j} - w_1^{p_2}| \\ |w_2^{y_j} - w_2^{p_2}| \\ \vdots \\ |w_V^{y_j} - w_V^{p_2}| \end{bmatrix}, \ldots, \begin{bmatrix} |w_1^{y_j} - w_1^{p_Q}| \\ |w_2^{y_j} - w_2^{p_Q}| \\ \vdots \\ |w_V^{y_j} - w_V^{p_Q}| \end{bmatrix} \right\} \tag{6}$$

where $w_h^{y_j}$ and $w_h^{p_k}$ are the hth features of documents y_j and p_k respectively and V is the number of features.

Then, the classification is performed using the output of the Dichotomy Transformation ($\mathbf{Z}'_s$) and the classes of each document in the representation set ($class(\mathbf{R}_s)$). For clarity, Fig. 4 expands one of the modules "Classification using o_s" showed in Fig. 3. So, each document $z'_{j,k} \in \mathbf{Z}'_s$ is given as input to the classifier o_s and the output of o_s is the probability of $z'_{j,k}$ to be classified as positive ($p(z'_{j,k}|\bullet)$). The probabilities for every $z'_{j,k}$ are combined in the Dichotomy Fusion module (Fig. 4), resulting in:

$$class_{o_s}(y_j) = \underset{c=\{1,2,\ldots,C\}}{\operatorname{argmax}} \sum_{\substack{p_k \in \mathbf{R}_s \\ class(p_k)=c}} p(z'_{j,k}|\bullet). \tag{7}$$

In other words, the output of the Dichotomy Fusion module is given the class of the documents (p_k) of the representation set with the highest sum of probabilities of belonging to the positive class ($p(z'_{j,k}|\bullet)$).

Fig. 4 shows only one of the "Classification using o_s" modules in Fig. 3. So, the output $class_{o_s}(y_j)$ for all L modules should be computed and combined by the Fuser module (Fig. 3) that gives the final response $class(y_j)$. Majority vote [30] is used to combine the L responses. Among a myriad of combination rules, majority vote is the simplest one. Other advantages of the majority vote are: (i) it does not assume prior knowledge about the classifiers, (ii) it is a non-trainable rule, (iii) it can be used with any classifier since it is a hard level combination rule.

3.4. Toy example

This section presents a run-through of the whole CoDiT process with a in depth view of classifier o_1. Fig. 5(a) shows the training set $\mathbf{X}$ with 22 documents ($N = 22$) distributed into 3 classes ($\mathbf{C} = \{1, 2, 3\}$). The rest of the input parameters for CoDiT are: the ensemble size ($L = 3$) and both the training subset and the representation set have the same number of documents ($N' = Q = 6$).

CoDiT starts with the Subsets Generation module and the Representation Sets Generation module. Both modules generate $L = 3$ training subsets, but each module works differently in regards to the selection of documents from the training set. The Subsets Generation modules generate the training subsets $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$, each with $N' = 6$ documents selected at random from the training set. Fig. 5(b) shows the documents of $\mathbf{X}_1$. Whereas, the Representation Sets Generation module generates the representation sets $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$, each with $Q = 6$ documents selected at random with the restriction of selecting the same number of documents per class, thus $Q/C = 2$ documents are selected per class. Fig. 5(c) shows the documents of $\mathbf{R}_1$. After, the dichotomy set $\mathbf{Z}_1$ is obtained with $DT(\mathbf{X}_1, \mathbf{R}_1)$ using Eq. (1). Fig. 5(d) presents the transformed documents of $\mathbf{Z}_1$, divided into 2 classes, the positive class (•) and the negative class (○).

The final step of the training phase is to train the $L = 3$ classifiers using the data of each $\mathbf{Z}_s$, where $s = \{1, 2, 3\}$. Fig. 6(a) shows classifier o_1 hyperplane dividing the 2 classes trained with the documents of $\mathbf{Z}_1$.

Fig. 6(b) shows a test document y_j in regard to the representation set $\mathbf{R}_1$, to demonstrate how the original documents are before the Dichotomy Transformation. Fig. 6(c) shows the 6 documents obtained with the Dichotomy Transformation $DT(y_j, \mathbf{R}_1)$ using Eq. (1). Fig. 6(d) shows the probabilities of each transformed document in regards to the classifier o_1. Then, document y_j is classified by o_1 as belonging to class 3, because the sum of the probabilities $p(z'_{j,k}|\bullet)$ when $class(p_k) = 3$ is higher than the sum for the other classes. More specifically, the sums are 1.06 (0.65 + 0.41) for class 1, 0.72 (0.34 + 0.38) for class 2 and 1.25 (0.7 + 0.55) for class 3.

Giving that $L = 3$, y_j is also classified by o_2 that responds class 1 and classifier o_3 that responds class 3. Then, using majority vote, $class(y_j) = 3$, because class 3 has two votes (o_1 and o_3), whereas class 1 has only one vote (o_2).

4. Experiments

All the experiments were performed with 10-fold stratified cross-validation. Table 1 shows the characteristics and the input parameters of CoDiT in all 14 databases: number of classes, number of documents and number of features; also, average number of documents in the training subsets ($\bar{N}'$) and average number of documents in the representation set ($\bar{Q}$). Both N' and Q are presented as average because the value can change within different folds. All databases were obtained in the following site: http://sites.labic.icmc.usp.br/text_collections/.

Macro averaged F1 (macro-F1) and micro averaged F1 (micro-F1) [31] were the evaluation metrics used for all the experiments. F1 is calculated using this general formula:

$$\text{F1} = \frac{2 \times \text{recall} \times \text{precision}}{(\text{recall} + \text{precision})}. \tag{8}$$

However, the definitions of precision and recall differ for macro-F1 and micro-F1, as follows:

$$\text{precision-macro} = \frac{\sum_{c=1}^{C} \frac{TP_c}{(TP_c + FP_c)}}{C}, \tag{9}$$

$$\text{recall-macro} = \frac{\sum_{c=1}^{C} \frac{TP_c}{(TP_c + FN_c)}}{C}, \tag{10}$$

$$\text{precision-micro} = \frac{\sum_{c=1}^{C} TP_c}{\left(\sum_{c=1}^{C} TP_c + \sum_{c=1}^{C} FP_c\right)}, \tag{11}$$

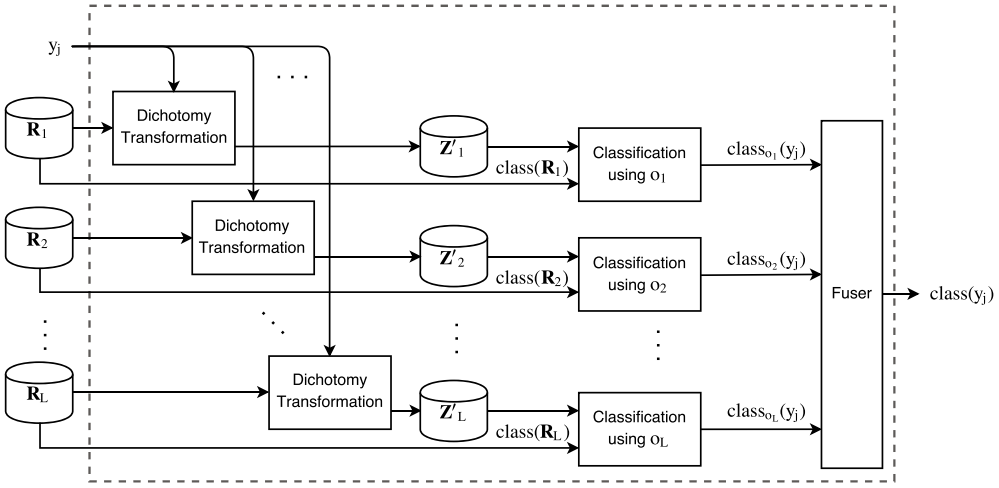


Fig. 3. Test phase of CoDiT. $y_j \in \mathbf{Y}$ is a test set document, $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$, are the representation sets, $\mathbf{Z}'_1, \mathbf{Z}'_2, \ldots, \mathbf{Z}'_L$ are the dichotomy sets based on $DT(y_j, \mathbf{R}_s)$, $class(\mathbf{R}_s)$ are the classes of all $p_k \in \mathbf{R}_s$, $class(y_j)$ is the resulting class of $y_j \in \mathbf{Y}$ and L is the ensemble size.

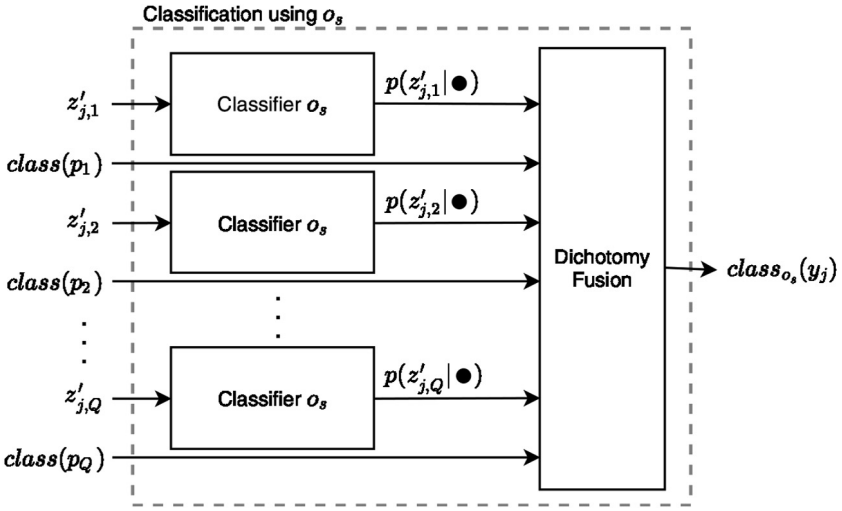


Fig. 4. Expansion of the module "Classification using o_s" showed in Fig. 3. $z'_{j,k}$, $class(p_k)$ and $p(z'_{j,k}|\bullet)$ $(k = \{1, 2, \ldots, Q\}$ are the dichotomy documents from $\mathbf{Z}_s$, the classes of the documents in the representation set $\mathbf{R}_s$ and the probability of classifying the dichotomy document as positive, respectively. The output $class_{o_s}(y_j)$ is the resulting class of each module.

$$\text{recall-micro} = \frac{\sum_{c=1}^{C} TP_c}{\left(\sum_{c=1}^{C} TP_c + \sum_{c=1}^{C} FN_c\right)}, \tag{12}$$

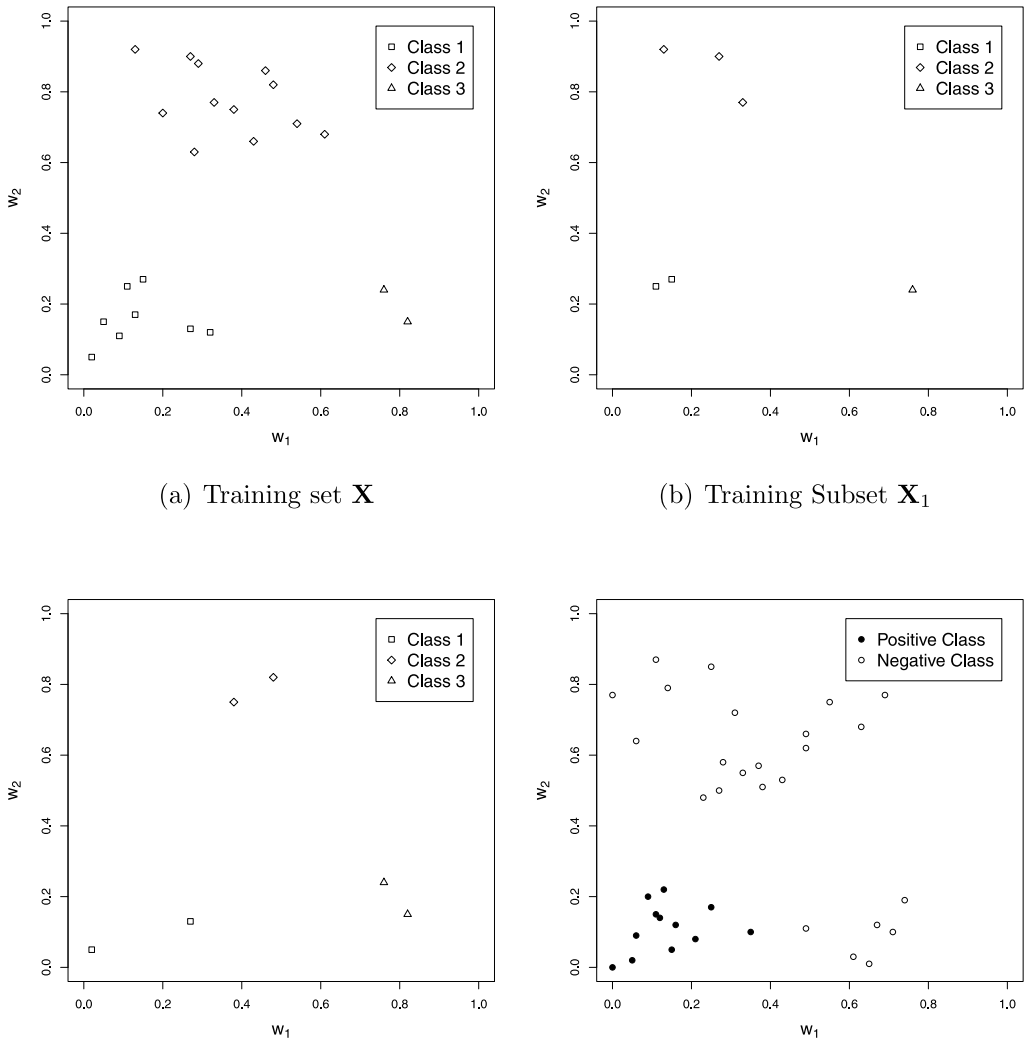
where C is the number of classes, TP_c is the number of correctly classified documents as class c, FN_c is the number of incorrectly classified documents to other class but c and FP_c is the number of incorrectly classified documents as class c.

Fig. 7 shows how the number of documents in the training set increases after the dichotomy transformation based on the size of the training subset and representation set. Due to the significantly growth of documents after the Dichotomy Transformation, both the number of documents in the training subsets (N') and the number of documents in the representation sets (Q) were defined as 10% of the number of documents in the training set $(0.1 \times N)$. However, the representation set must have the same number of documents for all classes, thus in some databases this value is not feasible. For example, the database Tr31 (Table 1) has 7 classes. In one of the folds used in the experiments, the number of documents of the training set is 834. Thus, the value of Q should be 83 $(834 \times 0.1 = 83.4)$, but this number cannot be used because the minority class has only 2 documents. In this case, Q is equal to the number of documents in the minority class multiplied by C $(2 \times 7 = 14)$.

Table 1
The characteristics of the databases and the input parameters of CoDiT.

id	Database	Classes	Documents	Features	$\bar{N}'$	$\bar{Q}$	$\bar{Q}/C$
1	CSTR	4	299	1726	26.4	24	6
2	Oh0	10	1003	3183	89.9	89	8.9
3	Oh5	10	918	3013	82.1	80	8
4	Oh10	10	1050	3239	94	90	9
5	Oh15	10	913	3103	81.6	80	8
6	Re0	13	1504	2887	135.1	128.7	9.9
7	Re1	25	1657	3759	148.6	127.5	5.1
8	Syskill	4	334	4340	29.5	28	7
9	Tr11	9	414	6430	36.9	36	4
10	Tr12	8	313	5805	27.7	24	3
11	Tr31	7	927	10129	82.9	12.6	1.8
12	Tr41	10	878	7455	78.6	70	7
13	Tr45	10	690	8262	61.6	60	6
14	WAP	20	1560	8461	140	90	4.5

CoDiT was compared with 5 methods: (i) SVM that is a common classifier used in Text Categorization [20,21]; (ii) Bootstrap AGGregatING [17], also known as Bagging, commonly used as baseline for multiple classifier systems [32–34]; (iii) Random Subspace [35], due to its capability of dealing with problems of



(a) Training set $\mathbf{X}$ (b) Training Subset $\mathbf{X}_1$

(c) Representation Set $\mathbf{R}_1$ (d) Dichotomy Set $\mathbf{Z}_1$

Fig. 5. Example of Dichotomy Transformation with inputs sets $\mathbf{X}_1$ (top-right) and $\mathbf{R}_1$ (bottom-left), and dichotomy set $\mathbf{Z}_1$ (bottom-right). All sets are represented by two features (w_1 and w_2).

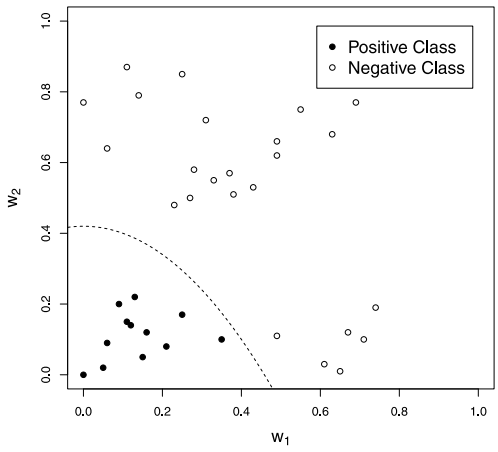
high-dimensionality [36,37]; (iv) BoosTexter [38], a boosting-based algorithm specifically designed to work with text categorization; (v) Random Forest [18], because of its robustness [39].

Our implementations of Bagging, Random Subspace, and CoDiT use SVMs as base classifiers (o_s). All experiments with SVM were performed with linear kernel, which is more suitable for text categorization problems [40]. The ensemble size varied in 10 different configurations ($L = \{10, 20, \ldots, 100\}$) for all ensemble methods, with the exception of BoosTexter and Random Forests that uses $L = 100$. Other setup configurations of Random Forest are: (i) the number of features for each classifier, which was chosen as one-tenth of the original features ($V/10$), as presented in [41,42]; and (ii) the number of features used to train each classifier, which was chosen as 20% of the original features, following the configuration used in [43,44].

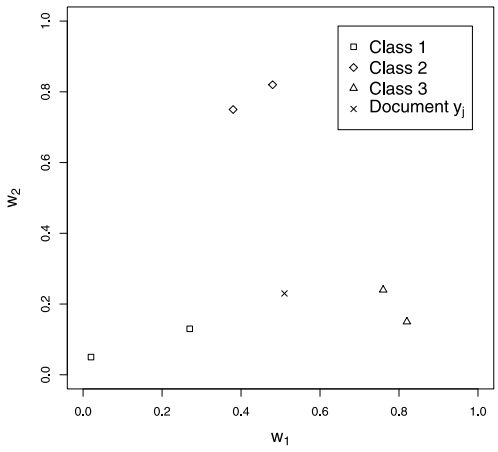
Table 2 shows a comparison of the results, the ensemble size (L), macro-F1, and micro-F1 obtained for each method. For Bagging, Random Subspace, and CoDiT, all values of L were evaluated and only the best one is shown. The best results of each database are highlighted in bold. The last row shows the p-value of the Wilcoxon signed rank test. This test verifies if CoDiT is better than other methods. Based on such statistical test, CoDiT is considered statistically better than all other methods with a confidence level of 99%.

Further analysis was performed with the Friedman test used to compare the results of all methods considering the 14 databases. Fig. 8 shows two Critical Difference (CD) diagrams [45], one for each performance measure. In regard to macro-F1, CoDiT is statistically better when compared to the other methods; while, in regard to micro-F1, CoDiT is statistically better than Bagging, Random Subspace and Single SVM, but it is considered statistically similar to Random Forest and BoosTexter. Moreover, CoDiT obtained the lowest average rank in both measures.

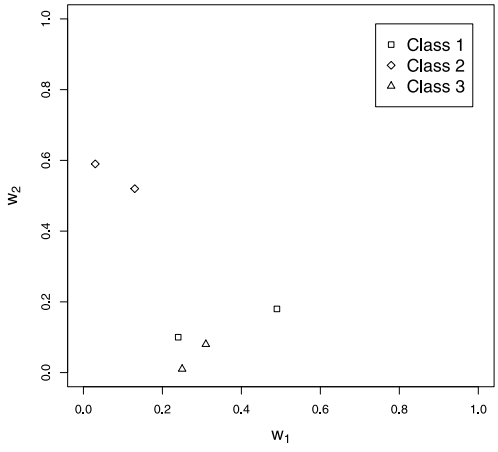
Fig. 9 shows the difference between the results of CoDiT and SVM for different number of documents per class in the representation set (Q/C). The value of Q is the average among all folds, because the number can change depending on the class distribution in each fold. Each point in the plots of Fig. 9 represents the result of each database, thus each plot has 14 points. Points below the dotted line mean that SVM is better than CoDiT in the indicated database, while points above the dotted line mean that CoDiT is better than SVM in the indicated database. The improvement of CoDiT increases with $\bar{Q}/C$, as denoted by the continuous line that represent the correlation. This behavior is expected, since more documents per class in the representation sets indicates an



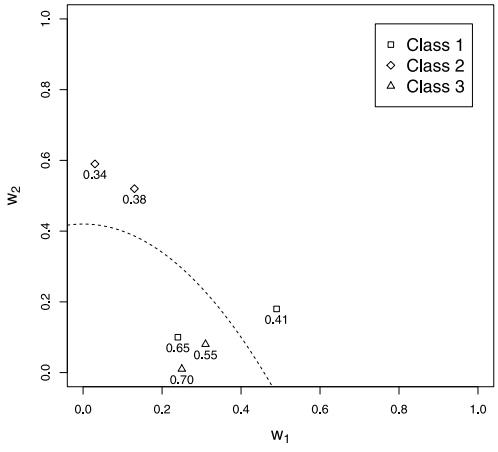
(a) Classifier o_1 (dashed line) trained with data from $\mathbf{Z}_1$.



(b) y_j and $\mathbf{R}_1$ before DT.



(c) $\mathbf{Z}'_1 = \{z'_{j,1}, z'_{j,2}, \ldots, z'_{j,Q}\}$



(d) Classifier o_1 (dashed line) and probabilities

Fig. 6. Document y_j is presented (b), pass through the Dichotomy Transformation and generates $Q = 6$ documents (c). The numbers below the documents in (d) are the probabilities $p(z'_{j,k}|\bullet)$. Thus, $class_{o_1}(y_j) = 3$. All documents of this example are represented by two features (w_1 and w_2).

Table 2
Macro-F1 and micro-F1 results for each method. The ensemble size (L) is presented in some methods. In bold are the best results for each database. The last two rows present the average of the performance measures and the p-value of Wilcoxon signed rank test.

	Single SVM		BoosTexter		Random Forest		Bagging			Random Subspace			CoDiT		
Bases	macro	micro	macro	micro	macro	micro	L	macro	micro	L	macro	micro	L	macro	micro
CSTR	80.56	79.21	77.37	76.70	75.60	77.67	90	79.09	77.90	50	**84.39**	**80.92**	90	80.92	80.73
Oh0	85.64	86.55	86.99	86.84	88.00	87.07	50	86.38	87.41	70	87.37	88.42	80	**92.26**	**92.45**
Oh5	86.55	86.73	86.47	87.10	88.88	89.06	60	86.78	86.83	90	87.09	86.9	100	**91.72**	**91.91**
Oh10	74.91	77.59	76.00	77.69	84.00	84.34	30	77.69	79.97	100	80.26	81.82	100	**84.68**	**86.43**
Oh15	80.56	81.00	84.12	83.84	84.40	83.70	60	80.76	81.01	100	82.67	82.67	70	**88.60**	**88.44**
Re0	78.03	83.65	73.78	83.56	75.07	**84.31**	100	**79.18**	83.97	100	71.13	79.90	90	77.21	78.71
Re1	74.43	83.99	73.36	84.80	74.96	86.30	10	73.64	84.54	80	67.90	81.38	50	**79.40**	**87.52**
Syskill	92.73	92.77	95.09	95.16	**95.48**	**95.46**	10	93.39	93.05	100	92.73	92.98	30	94.97	94.88
Tr11	73.91	87.20	80.91	91.14	78.80	**91.82**	10	74.23	88.12	10	77.50	90.32	90	**81.05**	89.18
Tr12	83.21	87.23	**89.95**	**91.77**	87.65	91.44	30	84.30	88.16	100	84.12	86.96	90	88.09	89.15
Tr31	83.36	97.74	**85.15**	**99.47**	83.71	98.40	60	83.38	97.74	80	83.46	97.95	20	82.96	97.08
Tr41	89.04	96.02	90.09	**96.99**	86.37	95.83	40	87.21	95.21	50	89.14	96.32	90	**90.55**	96.25
Tr45	85.97	91.59	**93.72**	**97.09**	87.81	94.69	50	85.34	90.86	100	88.10	93.26	50	93.30	94.60
WAP	**74.08**	**85.69**	60.22	78.59	61.86	81.00	60	68.45	84.24	70	71.61	85.65	90	72.06	85.42
Average	81.64	86.92	82.37	87.91	82.32	88.65	47.14	81.41	87.07	78.57	81.96	87.53	74.28	85.55	89.48
p-value	2.25e−10	8.61e−09	1.68e−08	4.95e−04	4.36e−09	7.96e−03	–	1.03e−11	1.98e−08	–	2.64e−10	2.59e−07	–	-	–

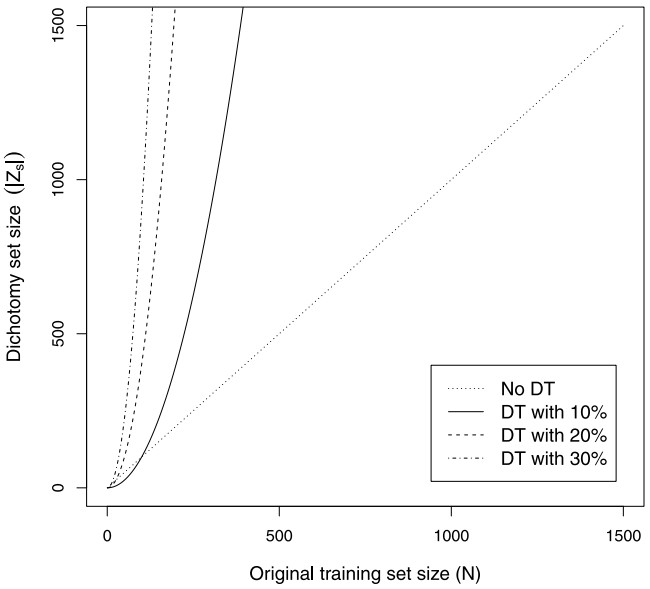


Fig. 7. Number of documents after the Dichotomy Transformation (DT) using three different sizes (10%, 20% and 30% of the training set) for the input sets ($\mathbf{X}_s$ and $\mathbf{R}_s$). The dotted straight line (no DT) is a baseline for comparison.

increase of the amount of information used to classify an unlabeled test document. Some databases do not present the expected relationship, the most noticeable examples are Re0 (6) and WAP (14). For Re0, CoDiT achieves the worst results than any other method in micro-F1, probably because the majority class (40.43% of the whole database) was more neglected than expected, and for WAP, the SVM achieved the best results in comparison to any other method, so it is expected for CoDiT to have a negative performance when compared to SVM.

Based on the results of the experiments, we evaluate the performance (macro-F1 and micro-F1) variation over the ensemble size (L) for CoDiT as showed in Fig. 10. Wilcoxon signed rank test was performed to compare every increment of L ($L = 10$ vs. $L = 20$, $L = 20$ vs. $L = 30$, and so on). The Wilcoxon test showed that CoDiT continues to improve the performance until $L = 50$, with a confidence level of 90%. From this moment forward, $L = 50$ is considered statistically similar to $L = \{60, 70, 80, 90, 100\}$. The only exception is the comparison of $L = 50$ with $L = 90$ in regards to macro-F1, where $L = 90$ is considered statistically better. Therefore, the best configuration regarding of performance is $L = 90$, but the best configuration taking into consideration the tradeoff between the size of the ensemble L and performance is $L = 50$.

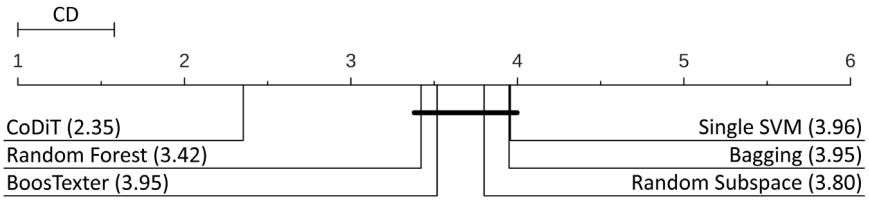
Table 2 shows that Random Forest obtains higher micro-F1 value in the following datasets: Re0, Syskill, Tr11, Tr12, Tr31, and Tr45. We can observe that these datasets shared the following characteristics: low Q/C and more than 4000 features. So, these characteristics seem to be an interesting indicator when choosing the best model. However, it is worth mention that CoDiT was better than the Random Forest in WAP dataset (high number of features and low Q/C).

5. Conclusions

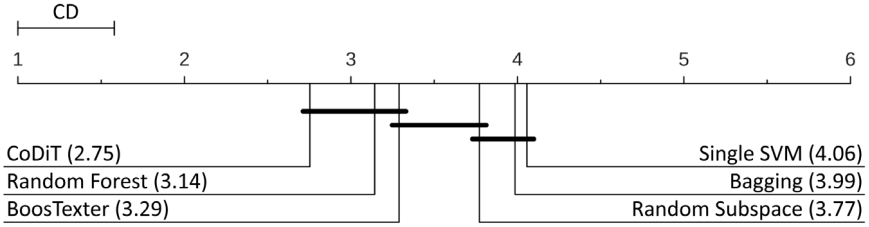
This paper presented Combined Dissimilarity Transformations (CoDiT), a Text Categorization system that combines binary classifiers trained with different dichotomy sets provided by a Dichotomy Transformation. CoDiT is composed of several modules, where each module can be replaced with other methodology: a different way of generating the subsets or the representation sets; a different classifier; or different rules for combination. However, each module was chosen carefully to generate a strong diverse ensemble. Also, due to the large number of possible iterations, some parameters (N' and Q) were fixed to keep the experiments under a feasible computational time and simply because of the exponential increase of the number of documents after the Dichotomy Transformation.

In the experiments, CoDiT is compared with other methods from the literature, such as Bagging, Random Subspace, BoosTexter, Random Forest and individual SVM. CoDiT stands out in the majority of the cases and is considered statistically better than any other method with a level of confidence of at least 99% with the Wilcoxon signed rank test. Friedman test shows a superior average rank for CoDiT, and statistically greater performance in regards to macro-F1.

Experiments were performed to verify the influence of the ensemble size in the performance of CoDiT. CoDiT's macro-F1 and



(a) Macro-F1



(b) Micro-F1

Fig. 8. CD diagrams for macro-F1 (a) and micro-F1 (b) comparing all methods used in the experiments. Average rank is presented for each method and CD = 0.5788 in both diagrams.

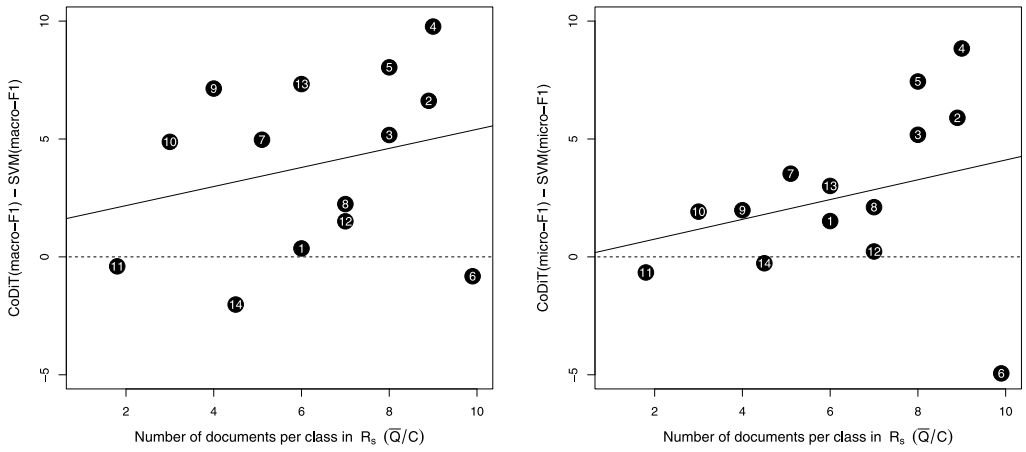


Fig. 9. Influence of average number of documents per class in the Representation Set $\mathbf{R}_s$ $(\bar{Q}/C)$ over the macro-F1 (a) and micro-F1 (b) of CoDiT minus the macro-F1 and micro-F1 of SVM (baseline). Each point is a database of Table 1, the number is a direct reference to 'id' column. The dashed line is a reference of no improvement of CoDiT and the continuous line is the correlation of the observed axes.

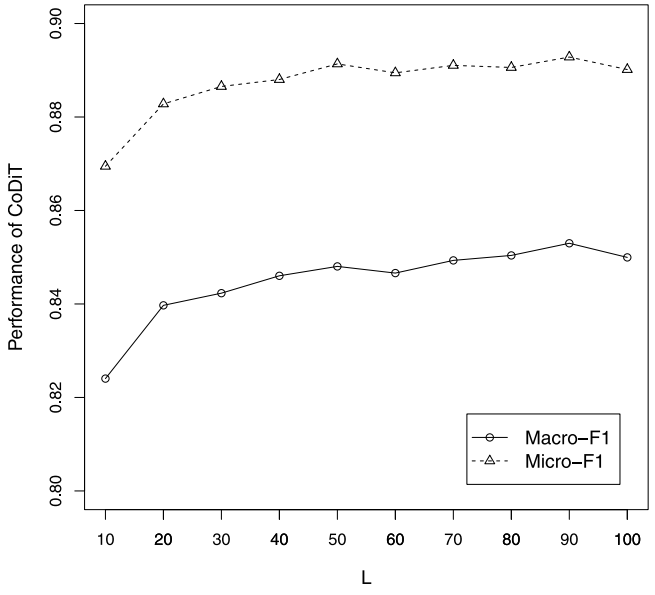


Fig. 10. Performance of CoDiT (macro-F1 and micro-F1) when different values of the ensemble size (L) are used.

micro-F1 improves gradually until $L = 50$ and achieves its highest values with $L = 90$. Therefore, the best choices are $L = 50$ or $L = 90$, where the final choice depends on how important is the computational time for the application. We also evaluate the relationship between the number of documents per class in the representation set and the improvement of CoDiT over a baseline (SVM). This evaluation showed a positive correlation with the observed variables, which indicates some *a priori* information for future experiments and improvements of CoDiT.

Among the future works we include, generate the representation set to deal with the problem of unbalanced classes in the databases, allowing to increase the number of documents per class in the representation set; evaluation of CoDiT in more databases; and replacing the use of classifier by some simple heuristics, based on the Dichotomy Transformation characteristics. Also, this work introduces the use of Dichotomy Transformation for other applications instead of only in identification and verification problems.

Acknowledgments

This research has been supported by the following Brazilian agencies: CNPq, Brazil (#446831/2014-0) and FACEPE, Brazil (APQ-0192-1.03/14 and IBPG-0613-1.03/12). We also thank Dayvid Victor Rodrigues de Oliveira for helping with Fig. 8.

References

[1] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, J. Mach. Learn. Res. 1 (Dec) (2000) 113–141.
[2] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychol. Rev. 65 (6) (1958) 386.
[3] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[4] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: European Conference on Computational Learning Theory, Springer, 1995, pp. 23–37.
[5] M. Aly, Survey on multiclass classification methods, Neural Netw. (2005) 1–9.
[6] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Tech. rep., DTIC Document, 1985.
[7] T. Hastie, R. Tibshirani, Classification by pairwise coupling, in: Advances in Neural Information Processing Systems, 1998, pp. 507–513.
[8] S.H. Cha, S.N. Srihari, Writer identification: statistical analysis and dichotomizer, in: Advances in Pattern Recognition, Springer, 2000, pp. 123–132.
[9] S.N. Srihari, A. Xu, M.K. Kalera, Learning strategies and classification methods for off-line signature verification, in: International Workshop on Frontiers in Handwriting Recognition, IEEE, 2004, pp. 161–166.
[10] D. Bertolini, L.S. Oliveira, E. Justino, R. Sabourin, Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers, Pattern Recognit. 43 (1) (2010) 387–396.
[11] D. Rivard, E. Granger, R. Sabourin, Multi-feature extraction and selection in writer-independent off-line signature verification, Int. J. Doc. Anal. Recognit. 16 (1) (2013) 83–103.
[12] N.P. Trilok, S.-H. Cha, C.C. Tappert, Establishing the uniqueness of the human voice for security applications, CSIS Research Day of Pace University, 2004.
[13] S. Yoon, S.-S. Choi, S.-H. Cha, Y. Lee, C.C. Tappert, On the individuality of the iris biometric, in: International Conference Image Analysis and Recognition, Springer, 2005, pp. 1118–1124.
[14] C.C. Tappert, M. Villani, S.-H. Cha, Keystroke biometric identification and authentication on long-text input, Behav. Biom. Hum. Identif. Intell. Appl. (2009) 342–367.
[15] L. Lam, Classifier combinations: implementations and theoretical issues, in: International Workshop on Multiple Classifier Systems, Springer, 2000, pp. 77–86.
[16] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, Inf. Fusion 6 (1) (2005) 5–20.
[17] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[18] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[19] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, Eur. Conf. Mach. Learn. (1998) 137–142.

[20] G. Pang, S. Jiang, A generalized cluster centroid based classifier for text categorization, Inf. Process. Manage. 49 (2) (2013) 576–586.
[21] R.G. Rossi, A. de Andrade Lopes, T. de Paulo Faleiros, S.O. Rezende, Inductive model generation for text classification using a bipartite heterogeneous network, J. Comput. Sci. Tech. 29 (3) (2014) 361–375.
[22] L. Borrajo, A.S. Vieira, E.L. Iglesias, TCBR-HMM: An HMM-based text classifier with a CBR system, Appl. Soft Comput. 26 (2015) 463–473.
[23] E. Pekalska, R.P. Duin, Dissimilarity representations allow for building good classifiers, Pattern Recognit. Lett. 23 (8) (2002) 943–956.
[24] R.P. Duin, E. Pekalska, The dissimilarity representation for structural pattern recognition, in: Iberoamerican Congress on Pattern Recognition, Springer, 2011, pp. 1–24.
[25] R. Pinheiro, G. Cavalcanti, T. Ren, Text categorization based on dissimilarity representation and prototype selection, in: Brazilian Conference on Intelligent Systems, 2015, pp. 163–168.
[26] E. Pekalska, R.P. Duin, P. Paclik, Prototype selection for dissimilarity-based classifiers, Pattern Recognit. 39 (2) (2006) 189–208.
[27] C.A. Shipp, L.I. Kuncheva, Relationships between combination methods and measures of diversity in combining classifiers, Inf. Fusion 3 (2) (2002) 135–148.
[28] J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: A new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (10) (2006) 1619–1630.
[29] L.I. Kuncheva, J.J. Rodriguez, Classifier ensembles with a random linear oracle, IEEE Trans. Knowl. Data Eng. 19 (4) (2007) 500–508.
[30] Y.H. Li, A.K. Jain, Classification of text documents, Comput. J. 41 (8) (1998) 537–546.
[31] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (2002) 1–47.
[32] Y. Qian, Y. Liang, M. Li, G. Feng, X. Shi, A resampling ensemble algorithm for classification of imbalance problems, Neurocomputing 143 (2014) 57–67.
[33] C. Catal, M. Nangir, A sentiment classification model based on multiple classifiers, Appl. Soft Comput. 50 (2017) 135–141.
[34] M.-J. Kim, D.-K. Kang, Classifiers selection in ensembles using genetic algorithms for bankruptcy prediction, Expert Syst. Appl. 39 (10) (2012) 9308–9314.
[35] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844.
[36] Y. Piao, H.W. Park, C.H. Jin, K.H. Ryu, Ensemble method for classification of high-dimensional data, in: International Conference on Big Data and Smart Computing, IEEE, 2014, pp. 245–249.
[37] H. Yu, J. Ni, An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data, IEEE Trans. Comput. Biol. Bioinform. 11 (4) (2014) 657–666.
[38] R.E. Schapire, Y. Singer, BoosTexter: A boosting-based system for text categorization, Mach. Learn. 39 (2–3) (2000) 135–168.
[39] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, J. Mach. Learn. Res. 15 (2014) 3133–3181.
[40] N.M. Sharef, T. Martin, Evolving fuzzy grammar for crime texts categorization, Appl. Soft Comput. 28 (2015) 175–187.
[41] Y. Ye, Q. Wu, J.Z. Huang, M.K. Ng, X. Li, Stratified sampling for feature subspace selection in random forests for high dimensional data, Pattern Recognit. 46 (3) (2013) 769–787.
[42] Q. Wu, Y. Ye, H. Zhang, M.K. Ng, S.-S. Ho, ForesTexter: an efficient random forest algorithm for imbalanced text categorization, Knowl.-Based Syst. 67 (2014) 105–116.
[43] Y. Campos, C. Morell, F.J. Ferri, A local complexity based combination method for decision forests trained with high-dimensional data, in: International Conference on Intelligent Systems Design and Applications, IEEE, 2012, pp. 194–199.
[44] M.J. Gangeh, M.S. Kamel, R.P. Duin, Random subspace method in text categorization, in: International Conference on Pattern Recognition, 2010, pp. 2049–2052.
[45] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (Jan) (2006) 1–30.

Information Sciences 406–407 (2017) 87–101

Contents lists available at ScienceDirect

Information Sciences

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

Combining dissimilarity spaces for text categorization

Roberto H.W. Pinheiro*, George D.C. Cavalcanti, Ing Ren Tsang

Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Av. Jornalista Anibal Fernandes s/n, Cidade Universitária 50740-560, Recife, PE, Brazil

ARTICLE INFO

Article history:
Received 20 June 2016
Revised 29 March 2017
Accepted 9 April 2017
Available online 12 April 2017

Keywords:
Text categorization
Multiple classifier system
Dissimilarity representation

ABSTRACT

Text categorization systems are designed to classify documents into a fixed number of predefined categories. Bag-of-words is one of the most used approaches to represent a document. However, it generates high-dimensional sparse data matrix with a high feature-to-instance ratio. An aggressive feature selection can alleviate these drawbacks, but such selection degrades the classifier's performance. In this paper, we propose an approach for text categorization based on Dissimilarity Representation and multiple classifier systems. The proposed system, Combined Dissimilarity Spaces (CoDiS), is composed of multiple classifiers trained on data from different dissimilarity spaces. Each dissimilarity space is a transformation of the original space that reduces the dimensionality, feature-to-instance ratio, and sparseness. Experiments using forty-seven text categorization databases show that CoDiS presents a better performance in comparison to literature systems.

1. Introduction

Text categorization, also known as text classification or topic spotting, aims at classifying a document into one or more previously known categories. A common way to represent a document as a feature vector is using Bag-of-Words (BoW), in which each feature is a term that appears in any document of the *Corpus* (database of documents). BoW was first introduced for text retrieval [17] but now is the most used approach to represent a document in text categorization problems. However, BoW has three well-known drawbacks: i) High-dimensionality [22]: Each document is represented by a feature vector, in which each feature is a term that appears in any document of the *Corpus*. Since all terms of the *Corpus* are considered as features, it is common to have tens of thousands of features; ii) High feature-to-instance ratio [12]: the number of features (tens of thousands) is usually much higher than the number of documents, or instances, in the Corpus; iii) Sparse data matrix [37]: The value for the features of the terms that do not appear in the document is zero. Since each document has only a small percentage of all existing terms, the data matrix is sparse.

Feature selection algorithms reduce the dimensionality of the feature vectors by removing features based on heuristics. This strategy is commonly used to address the drawbacks of BoW. To determine which features should be discarded, different criteria are employed, such as feature evaluation functions [6,11,39] which generate a score per feature. In this case, features with a score greater than a threshold is considered relevant. There are other alternatives to select features such as the proposal of Mitra et al. [25] that uses structural similarity; ALOFT [30] that relies on the idea that each document should contribute to the feature vector; and, the proposal of Maldonado et al. [24] that is based on loss functions. Consequently, feature selection algorithms reduce the dimensionality and, consequently, the feature-to-instance ratio; but, sparseness is

* Corresponding author.
E-mail addresses: rhwp@cin.ufpe.br (R.H.W. Pinheiro), gdcc@cin.ufpe.br (G.D.C. Cavalcanti), tir@cin.ufpe.br (I.R. Tsang).

http://dx.doi.org/10.1016/j.ins.2017.04.025
0020-0255/

Table 1
Related works.

Method	Feature	Sparseness	Ensemble	Summary
Prabowo and Thelwall [33]	✓		✓	Combines different types of classifiers
Ozgur and Gungor [27]	✓			Bag of Words extension based on pruning concepts
Xia et al. [46]			✓	Integrates different types of classifiers, features sets and combination approachs
Pinheiro et al. [30]	✓			Feature selection method, called ALOFT
De Silva et al. [7]	✓		✓	Combines different feature representation with different types of classifiers
Wang et al. [41]	✓		✓	Compares ensemble methods
Wu et al. [45]	✓		✓	Hybrid (SVM and Random Forest) method for imbalanced text data
Jun et al. [18]	✓	✓		Document support vector clustering with dimension reduction
Pinheiro et al. [32]	✓	✓		Cosine representation with prototype selection
Zhang and He [49]		✓	✓	Two classifier ensemble with enriched Bag-of-Words
Altinel et al. [1]	✓	✓		Class weighting kernel
Onan et al. [26]	✓		✓	Statistical keyword extraction
Proposed method	✓	✓	✓	Combines different dissimilarity spaces

slightly diminished. An aggressive feature selection is required to address these three drawbacks. However, removing many features can increase the classification error because of information loss [47].

Here we propose an approach, called Combined Dissimilarity Spaces (CoDiS), for text categorization. CoDiS is a multiple classifier system in which each classifier is trained using a different Dissimilarity Representation [28] that transforms the feature vectors to a new low-dimensional representation. In this representation, each document is represented by a dissimilarity vector composed of distances to all documents that belongs to a representation set (a subset of the training set). Since the representation set contains documents from different categories, the dissimilarity between documents of the same category should be small, and the distance between documents in different categories should be significant, this property increases the discrimination between the categories [29].

Multiple classifier system (MCS) relies on the assumption that combining classifiers can lead to an improvement in the accuracy rate and usually performs better than individual classifiers [50]. Gangeh et al. [12] proposed an MCS that presented better performance in text categorization than state-of-the-art individual classifiers, such as Support Vector Machines (SVM). They used a classifier generation method called Random Subspace (RSS) [15] in which each classifier is trained with a random subset of the original features. Thus, each classifier can deal with a different part of the feature space, and the final answer is given by the combination of the classifiers responses. In the literature, there are other works [4,48] that successfully used Random Subspace to deal with high-dimensional problems.

The proposed approach, CoDiS, uses Dissimilarity Representation and multiple classifier systems to improve the accuracy rate while diminishes the drawbacks of BoW. We claim that the Dissimilarity Representation entails less information loss compared to an aggressive feature selection because feature selection discards features, whereas all terms are built in the calculation of the distance between documents in the Dissimilarity Representation. As a consequence of the dimensionality reduction promoted by the Dissimilarity Representation and the transformation of the original features to distances, all three BoW drawbacks are diminished. Multiple classifiers is a more robust solution than a single classifier to deal with different databases [48]. CoDiS also avoids searching for the best representation set since multiple classifiers can overcome local optima by combining different initial conditions [9].

Through a set of comprehensive experiments on 47 databases, we show that the proposed method can considerably reduce the dimensionality while improving the recognition rates. The results reached by our method compare favorably to other ensemble methods using three statistical tests.

Our contributions and findings can be summarized as follows: i) A new Text Categorization system, CoDiS; ii) Dissimilarity Representation can benefit from multiple classifiers; iii) Dissimilarity Representation can reduce sparseness and dimensionality with a good compromise between performance and reduction; iv) Euclidean distance obtains better results than cosine similarity for CoDiS because Dissimilarity Representation with Euclidean distance is able to generate diverse classifiers; v) Random prototype selection methods are more adequate than prototype selection algorithms to produce a diverse ensemble in the CoDiS architecture.

This paper is organized as follows: The next section reviews the literature focusing on works that deal with a high number of features, the sparseness of data and that use ensemble in Text Categorization. Section 3 reviews the Dissimilarity Representation, which is a transformation capable of reducing the drawbacks of Bag-of-Words. Section 4 presents the proposed system, called Combined Dissimilarity Spaces (CoDiS), a Text Categorization system that combines different dissimilarity spaces. Section 5 describes the methodology of the experiments, presents preliminary experiments and the analysis of the final experimental results. Finally, Section 6 concludes this paper and describes some future works.

2. Related works

In this section, we discuss some important Text Categorization works available in the literature. The articles showed in Table 1 focus on three aspects: i) feature reduction (column Feature) which indicates the use of feature selection, feature extraction, or other feature transformation; ii) sparseness treatment (column Sparseness) which indicates that the work

reduces the sparseness of the data; and, iii) combination of classifiers (column Ensemble) which indicates the use of combination of classifiers. The last column of Table 1 shows a summary of the method proposed in each paper.

In Özgür and Güngör [27], Ozgur and Gungor proposed a feature selection method to reduce the high-dimensionality generated by Bag-of-Words. However, in contrast to regular feature selection methods that require the number of features as an input parameter, they used a parameter called Pruning Level that is used to find the final number of features in a data-driven way. This strategy of finding the number of features per Corpus in a data-driven way was also employed by other recent works that perform feature selection for Text Categorization [30,31]. Altinel et al. [1] introduced a semantic kernel for SVM that smooths the representation of documents by introducing class-based dependencies between terms, reducing the dimensionality and sparseness. The proposed kernel performs fast and improves the results in text data. Jun et al. [18] also proposed a technique to reduce high-dimensionality and sparseness in text data performing a dimension reduction by combining singular value decomposition and principal component analysis. The main difficulty of this approach is to define the number of clusters, and the authors stated that a combined model could provide more valuable results. A similar idea was used by Pinheiro et al. [32], a cosine representation and prototype selection techniques were combined to reduce high-dimensionality and sparseness.

Ensemble was employed in text categorization problems by Prabowo and Thelwall [33]. Their method uses a small pool of classifiers, two to four, that are combined in a pipeline using rule-based classifiers and SVM. Only one of the classifiers of the ensemble, statistics based classifier, performs feature selection. The combinations with statistics based classifier were among the best configurations. Xia et al. [46] proposed a framework that integrates different types of classifiers (Naive Bayes, maximum entropy, and SVM) using different feature sets extracted from the Bag-of-Words representation. These heterogeneous classifiers were combined using different combination rules, such as fixed, weighted, and meta-classifier. In the end, several different configurations of the proposed framework were evaluated in five databases, exhibiting promising results for combining different feature vectors. De Silva et al. [7] also used a small and heterogeneous ensemble. This ensemble is composed of four classifiers: multinomial naive Bayes, SVM, Random Forest, and logistic regression; and the majority vote was adopted as the combination rule. They used feature hashing to reduce the number of features; however, Bag-of-Words without reduction achieved the best accuracy. The authors stated that more effort should be devoted to increase the classifiers' diversity to improve the accuracy rate. Wang et al. [41] evaluated the effectiveness of three ensemble generation methods (Bagging, Boosting, and Random Subspace) using an ensemble composed of five base classifiers (Naive Bayes, maximum entropy, decision tree, k Nearest Neighbors, and SVM). The best configuration was Random Subspace with SVM as base classifiers. It is important to remark that in contrast to the other two ensemble generation method, Random Subspace deals with high-dimensional data since each classifier is trained using a random subset of the features. Wu et al. [45] proposed a hybrid method for imbalanced text data combining the concepts of SVM and Random Forest, called ForestTexter. It manages the high-dimensionality of the data by sampling features similar to Random Subspace. Each node of the tree is an SVM classifier. Onan et al. [26] used keyword extraction to reduce the number of features and performed several experiments using four base learning algorithms (Naive Bayes, SVM, logistic regression, and Random Forest) with five ensemble methods (AdaBoost, Bagging, Dagging, Random Subspace, and Majority Voting). Despite the usage of several representations with keyword extraction, these extractions are never combined. In general, Bagging and Random Subspace ensembles combined with Random Forest yield promising results. Zhang and He [49] used latent topics and related words to enrich the Bag-of-Words representation which provides a way to make sparse text more related and topic-focused. Both representations are used to create an ensemble of two base classifiers: Naive Bayes and maximum entropy.

Some characteristics can be highlighted: i) Bag-of-Words is still a very common representation. Despite its problems, it is usually used as the first layer for a more complex representation; ii) SVM is a common and promising choice for text categorization problems, even when used in ensembles; iii) some works use multiple classifier system, however, the size of the pool of classifiers is commonly small. Larger pools could lead to an increase in diversity which plays a major role in this kind of systems; iv) the sparseness is usually addressed using some transformation.

Given such characteristics and the purpose of this paper, the proposed method offers a broader solution than the other works presented in this section. Only three works deals with sparseness and high-dimensionality [1,18,32], but none of them uses ensemble to obtain a more robust approach. Whereas the works focused in ensemble do not deal with both sparseness and high-dimensionality. Therefore, the proposed method (CoDiS) uses Dissimilarity Representation to deal with both problems of Bag-of-Words, as stated in Section 3 and showed in the Toy Example of Section 4.4. The strategy employed by CoDiS to generate the pool of classifier favors the creation of a diverse ensemble.

3. Dissimilarity Representation

Pekalska and Duin [28] proposed three different approaches for Dissimilarity Representation: pretopological, dissimilarity space, and embedding. The first approach refers to the well known k-NN classifier that can be considered as the first dissimilarity-based classifier, and the third approach requires modifications in each classifier depending on the measure used. Herein, we use the second approach, dissimilarity space, because of its universality. In this case, universality is the capability of using the Dissimilarity Representation with any measure and with any classifier that requires a feature vector as input.

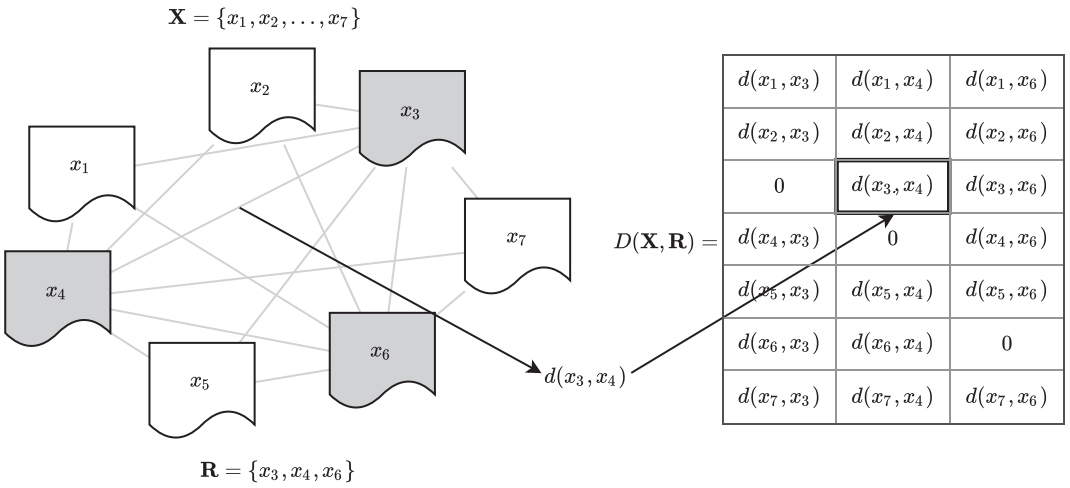


Fig. 1. An example showing the generation of the dissimilarity matrix $D(\mathbf{X}, \mathbf{R})$ using the training set $\mathbf{X}$ and the representation set $\mathbf{R}$.

The Dissimilarity Representation is a mapping defined by calculating the dissimilarity of a document (or set of documents) to every document of a representation set $\mathbf{R} = \{p_1, \ldots, p_Q\}$. The representation set is composed of Q prototypes selected from a larger set using any prototype selection method.

Given a training set $\mathbf{X}$ composed of N documents x_i represented by BoW as $w^{x_i} = [w_1^{x_i}, w_2^{x_i}, \ldots, w_V^{x_i}]^{\mathrm{T}}$, $w_h^{x_i}$ is the hth feature of x_i and V is the total number of features (size of the Vocabulary). The Dissimilarity Representation of a single document x_i is $D(x_i, \mathbf{R}) = [d(x_i, p_1), d(x_i, p_2), \ldots, d(x_i, p_Q)]^{\mathrm{T}}$, where $d(\cdot, \cdot)$ is the dissimilarity between two documents, usually defined as the Euclidean distance:

$$d(x_a, x_b) = \sqrt{\sum_{h=1}^{V} (w_h^{x_a} - w_h^{x_b})^2}, \tag{1}$$

where $w_h^{x_a}$ and $w_h^{x_b}$ are the hth features of documents x_a and x_b respectively. In our case $x_a \in \mathbf{X}$ and $x_b \in \mathbf{R}$. So, the Dissimilarity Representation of the training set $\mathbf{X}$ using the representation set $\mathbf{R}$ generates a dissimilarity matrix with $N \times Q$ dimensions:

$$D(\mathbf{X}, \mathbf{R}) = \begin{bmatrix} d(x_1, p_1) & d(x_1, p_2) & \ldots & d(x_1, p_Q) \\ d(x_2, p_1) & d(x_2, p_2) & \ldots & d(x_2, p_Q) \\ \vdots & \vdots & \ddots & \vdots \\ d(x_N, p_1) & d(x_N, p_2) & \ldots & d(x_N, p_Q) \end{bmatrix}. \tag{2}$$

The dissimilarity matrix can be used as a data matrix, where each line represents a document and each column accounts for a feature. Fig. 1 shows an example of the dissimilarity matrix $D(\mathbf{X}, \mathbf{R})$ that was generated using a training set $\mathbf{X}$ with seven documents and representation set $\mathbf{R}$ with three documents. $D(\mathbf{X}, \mathbf{R})$ has 7×3 dimensions, where each line represents each one of the seven training documents. We can observe that $D(\mathbf{X}, \mathbf{R})$ has three cells with zero values; this happens whenever a document x belongs simultaneously to the training and the representation sets, i.e., $d(x, x) = 0$.

It is important to remark that Dissimilarity Representation reduces all three previously pointed drawbacks of BoW. i) High-dimensionality: the number of features, after Dissimilarity Representation, is equal to the number of documents in the representation set (Q), which is a subset of the training set ($Q \leq N$). So, given that most Text Categorization corpus has a high feature-to-instance ratio, i.e., the original number of features (V) is higher than the number of documents in the training set ($V \gg N$). It is expected a reduction in the number of features by the dissimilarity representation ($Q \ll V$). ii) Sparse data matrix: after the application of Dissimilarity Representation, each feature is a distance between two documents from two different sets ($d(x_i, p_k)$). This distance is zero only if both documents have the same value for all features (which is rare). So, the sparseness is drastically reduced as shown in a Toy example in Section 4.4 where the sparseness level decreased from 61 to 7% after Dissimilarity Representation. iii) High feature-to-instance ratio: it is reduced based on the same rationality behind the dimensionality reduction ($Q \leq N$), actually, the feature-to-instance ratio is at most equal to 1 ($Q/N \leq 1$).

4. Combined Dissimilarity Spaces

The proposed method, Combined Dissimilarity Spaces (CoDiS), is a multiple classifier system. The main idea of CoDiS is to classify documents based on the combined response of multiple classifiers, where each classifier is trained on a different

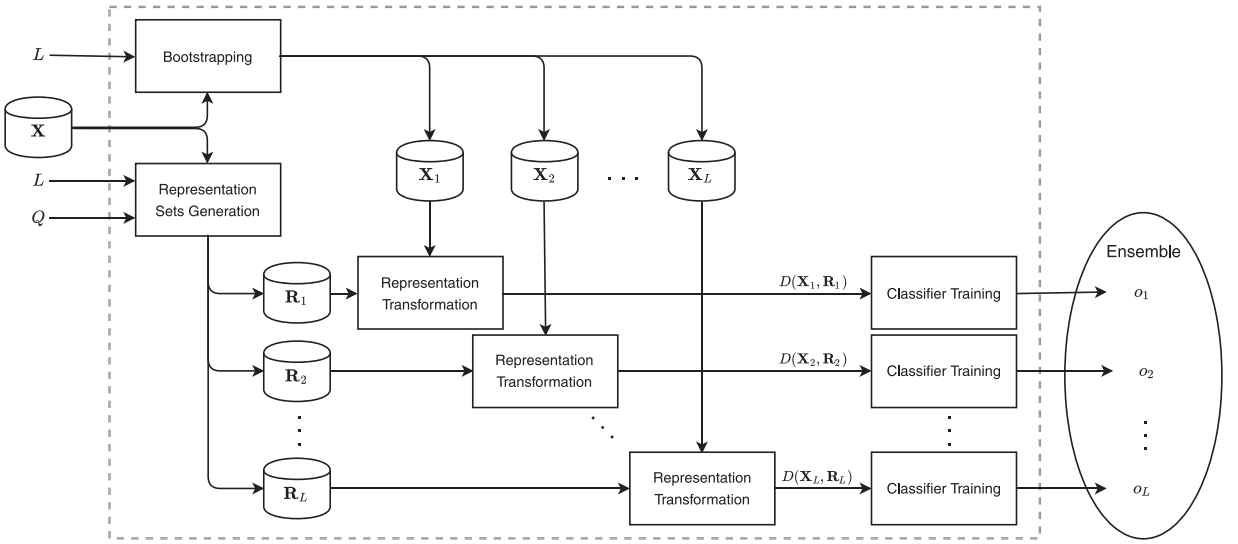


Fig. 2. Training phase of CoDiS. $\mathbf{X}$ is the training set, $\mathbf{X}_s$ are subsets of the training set, $\mathbf{R}_s$ are the representation sets, L is the number of classifiers in the pool, Q is a vector with the number of documents for each representation set, $D(\mathbf{X}_s, \mathbf{R}_s)$ are the Dissimilarity Representation performed over $\mathbf{X}_s$ using $\mathbf{R}_s$ as reference and o_s are the classifiers of the pool. Note that $s = \{1, 2, \ldots, L\}$.

dissimilarity space. The following sections describe the notation of the proposed method (Section 4.1); the two phases of the proposed method, training (Section 4.2), and test (Section 4.3); and a Toy example, showing how the proposed method works (Section 4.4).

4.1. Notation

CoDiS can be applied with several different sets, used as input, output or generated during the training phase. The notation of every set is presented as follows:

- The training set $\mathbf{X}$ is composed of N documents $x_i \in \mathbf{X} = \{x_1, x_2, \ldots, x_N\}$. Each x_i is represented by the pair $\langle w^{x_i}, class(x_i)\rangle$, where $w^{x_i} = [w_1^{x_i}, w_2^{x_i}, \ldots, w_V^{x_i}]^{\mathrm{T}}$ is the vector of terms using the Bag-of-Words and $class(x_i) \in \mathbf{C} = \{c_1, c_2, \ldots, c_C\}$ is the class of the document.
- The test set $\mathbf{Y}$ is composed of M documents $y_j \in \mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$ and uses the same Bag-of-Words representation as the training set.
- Subsets $\mathbf{X}_s$ ($s = \{1, 2, \ldots, L\}$) are subsets of the training set $\mathbf{X}$, each training subset $\mathbf{X}_s$ has the same size of the original training set (N).
- Representation sets $\mathbf{R}_s$ are composed of Q documents $p_k \in \mathbf{R}_s = \{p_1, p_2, \ldots, p_Q\}$.

4.2. Training phase

The training phase of CoDiS (Fig. 2) is composed of four modules: Bootstrapping, Representation Sets Generation, Representation Transformation and Train Classifier. The input parameters are the training set ($\mathbf{X}$), the number of classifiers (L) and the number of prototypes for each representation set $\mathbf{R}_s$ ($Q = \{q_1, q_2, \ldots, q_L\}$). In some cases, the number of prototypes is the same for every $\mathbf{R}_s$, so $q_1 = q_2 = \ldots = q_L$.

The Representation Transformation module works as the Dissimilarity Representation, described in Section 3. Here we call Representation Transformation instead of Dissimilarity Representation because the proposed method can use dissimilarity measure as well as other measures. The input sets of each Representation Transformation module are a training subset $\mathbf{X}_s$ and a representation set $\mathbf{R}_s$, having $s = \{1, 2, \ldots, L\}$. Given that CoDiS is a multiple classifier system, diversity plays an important role [36], thus both input sets should have different documents. Based on this premise, the Bootstrapping module and Representation Sets Generation module creates a diverse ensemble.

The Bootstrapping module generates different subsets of the training set using sampling with replacement as in Bootstrap AGGregatING (Bagging) [3]. This module receives the original training set $\mathbf{X}$ as input and generates L training subsets $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)$ as output. All subsets have the same size (N) as the original training set with some documents appearing multiple times. We use Bagging to increase the degree of instability of the base classifier [3] and to improve the robustness against outliers [14], without increasing the size of the original training set.

The Representation Sets Generation module generates different subsets of the training using any prototype selection algorithm. This module receives the original training set $\mathbf{X}$ as input, and generates L representation sets $(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L)$ as

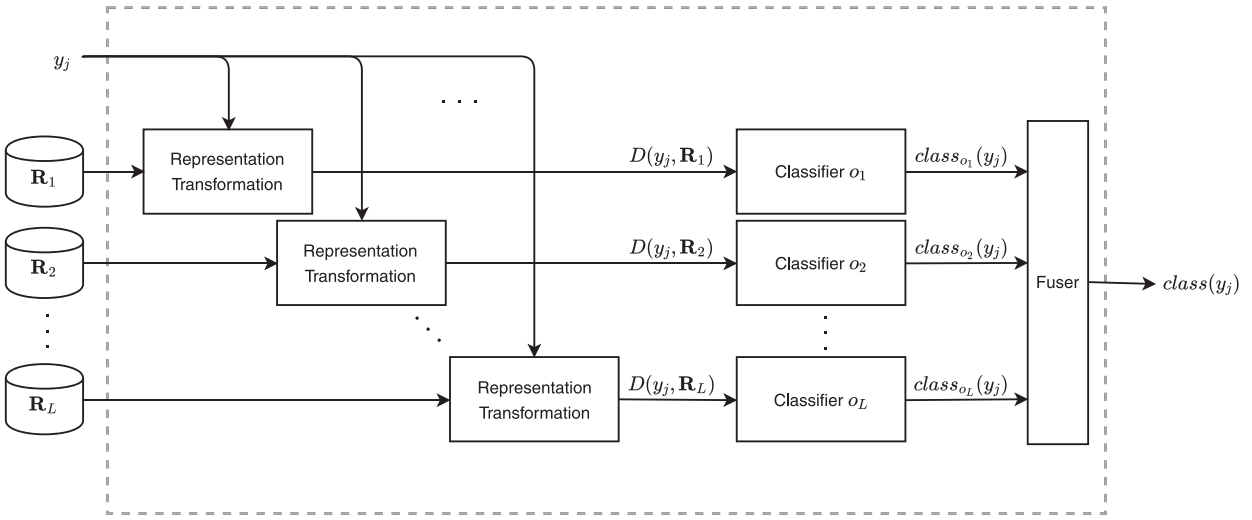


Fig. 3. Test phase of CoDiS. y_j is a document from the test set **Y**, $\mathbf{R}_s$ are the representation sets, L is the number of classifiers in the pool, $D(y_j, \mathbf{R}_s)$ are the Dissimilarity Representation performed over y_j using $\mathbf{R}_s$ as reference, $class_{o_s}(y_j)$ are the responses for each classifier o_s and $class(y_j)$ is the combined response for document y_j. Note that $s = \{1, 2, \ldots, L\}$..

output. Each representation set has q_s documents ($q_s \leq N$). However, it is important to highlight that deterministic prototype selection algorithm always generates the same subset given a training set, so, to obtain L different representation sets, we can use L different deterministic prototype selection algorithms or non-deterministic ones (such as random selection).

At this point, CoDiS generated $2 \times L$ subsets of the training set. The first half $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)$ from Bootstrapping, having N documents each, and the second half $(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L)$ from prototype selection, having q_s documents each. Now, each Representation Transformation module receives a pair $\mathbf{X}_s$ and $\mathbf{R}_s$ to generate a dissimilarity matrix $D(\mathbf{X}_s, \mathbf{R}_s)$:

$$D(\mathbf{X}_s, \mathbf{R}_s) = \begin{bmatrix} d(x_1, p_1) & d(x_1, p_2) & \ldots & d(x_1, p_{q_s}) \\ d(x_2, p_1) & d(x_2, p_2) & \ldots & d(x_2, p_{q_s}) \\ \vdots & \vdots & \ddots & \vdots \\ d(x_N, p_1) & d(x_N, p_2) & \ldots & d(x_N, p_{q_s}) \end{bmatrix}, \tag{3}$$

where each function $d(x_i, p_k)$ generates the kth feature of the document x_i as showed in Eq. (1). In this manner, each dissimilarity matrix $D(\mathbf{X}_s, \mathbf{R}_s)$ can be seen as a transformed set with N documents (rows of the matrix) represented by q_s features (columns of the matrix).

The output of the training phase is L trained classifiers $(o_1, o_2, \ldots, o_L)$, where each classifier o_s is trained using a dissimilarity matrix $D(\mathbf{X}_s, \mathbf{R}_s)$.

4.3. Test phase

The test phase of CoDiS (Fig. 3) is composed of three modules: Representation Transformation, Classifier and Fuser. The input parameters are a query document ($y_j \in \mathbf{Y}$), L representation sets $(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L)$ and L trained classifiers $(o_1, o_2, \ldots, o_L)$.

The Representation Transformation module receives a single document $y_j \in \mathbf{Y}$ and a representation set $\mathbf{R}_s$, obtained during the training set, as inputs; and generates a transformed document:

$$D(y_j, \mathbf{R}_s) = [d(y_j, p_1), d(y_j, p_2), \ldots, d(y_j, p_{q_s})]^{\mathrm{T}}, \tag{4}$$

represented by q_s features. Therefore, each element of the feature vector represents the distance between the query document y_j and one document p_k in the representation set $\mathbf{R}_s$. Note that the representation sets must be the same ones obtained in the training phase.

Each transformed document $D(y_j, \mathbf{R}_s)$ is presented to its respective classifier o_s that provides an output $class_{o_s}(y_j)$. Then, each outputs $class_{o_s}(y_j)$ of the L classifiers are combined by the Fuser module that gives the final response $class(y_j)$. There are three main approaches for combining [44]: i) class label fusion, where each classifier responds a class, and the responses are used as votes, such as unanimous voting, simple majority, majority vote [21] or weight differently each vote [38]; ii) support function fusion, where each classifier responds a probability for each class and some operation combines those responses, such as *a posteriori* probability [19], simple operator of mean value [10], among others; and, iii) trainable fuser, where the responses of all classifiers are used to train a classifier to give the combined answer based on a validation set, such as Perceptron with evolutionary algorithm [23], Dempster's and Shafer's theory [43] or using statistical approach [16]. CoDiS uses Majority vote [20] as the class label fusion approach. The Majority vote response for a given test document

Table 2
Toy example of CoDiS. The first column is to identify the set of the following document (second column), the last column ($class(\cdot)$) shows the class of each document and the columns between ($w_1, w_2, \ldots, w_9$) shows the features. Each line represents a document.

Set	Docs.	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	$class(\cdot)$
$\mathbf{X}_1$	x_1	2	1	5	0	0	0	1	5	0	A
	x_2	3	0	0	3	2	0	0	6	4	A
	x_3	0	6	0	0	0	0	0	0	0	A
	x_4	0	2	0	0	0	0	1	1	0	A
	x_5	0	0	0	3	0	3	0	0	6	B
	x_6	0	0	0	3	0	3	0	0	6	B
	x_7	1	0	1	4	2	0	0	0	0	B
	x_8	0	0	0	2	0	2	2	0	7	B
	x_9	5	0	1	3	1	0	5	0	0	B
	x_{10}	0	0	0	4	2	0	0	0	3	B
$\mathbf{R}_1$	p_1	3	0	0	3	2	0	0	6	4	A
	p_2	0	2	0	0	0	0	1	1	0	A
	p_3	0	0	0	6	0	6	0	0	0	B
	p_4	5	0	1	3	1	0	5	0	0	B
$\mathbf{Y}$	y_j	2	0	0	4	0	3	0	0	0	B

Table 3
Dissimilarity matrix $D(\mathbf{X}_1, \mathbf{R}_1)$ of the Toy example and the respective class of each document ($class(x_i)$).

$D(\mathbf{X}_1, \mathbf{R}_1)$	p_1	p_2	p_3	p_4	$class(x_i)$
$D(x_1, \mathbf{R}_1)$	7.615773	6.782330	11.313708	8.774964	A
$D(x_2, \mathbf{R}_1)$	0.000000	8.246211	10.488088	9.110434	A
$D(x_3, \mathbf{R}_1)$	10.488088	4.242641	10.392305	9.848858	A
$D(x_4, \mathbf{R}_1)$	8.246211	0.000000	8.831761	7.549834	A
$D(x_5, \mathbf{R}_1)$	7.874008	7.745967	7.348469	9.848858	B
$D(x_6, \mathbf{R}_1)$	7.874008	7.745967	7.348469	9.848858	B
$D(x_7, \mathbf{R}_1)$	7.615773	5.291503	6.782330	6.557439	B
$D(x_8, \mathbf{R}_1)$	8.185353	7.937254	9.219544	9.486833	B
$D(x_9, \mathbf{R}_1)$	9.110434	7.549834	9.848858	0.000000	B
$D(x_{10}, \mathbf{R}_1)$	6.855655	5.916080	7.280110	7.874008	B

y_j is the class having the highest number of votes among the classifiers of the ensemble. Majority vote is used in CoDiS due to its simplicity, since it neither requires probabilities calculation, nor training the outputs based on a validation set and its improvement over the individual classifiers performance [42].

4.4. Toy Example

This section presents a hypothetical example of the proposed method. Table 2 shows the training subset $\mathbf{X}_1$ with 10 documents ($N = 10$), the representation set $\mathbf{R}_1$ with 4 documents ($Q = 4$), and one test example $y_j \in \mathbf{Y}$. All documents are represented by 9 features and divided into two classes ($\mathbf{C} = \{A, B\}$).

The training subset $\mathbf{X}_1$ is generated using Bagging. Therefore, some documents of the original training set are not present, while other documents appear more than once, for instance, $x_5 = x_6$. The representation set $\mathbf{R}_1$ is also randomly sampled from the original training set, so, it share some documents with the training subset, for instance, $p_1 = x_2$, $p_2 = x_4$ and $p_4 = x_9$. The next step is to calculate the dissimilarity matrix $D(\mathbf{X}_1, \mathbf{R}_1)$ showed in Table 3.

CoDiS is a multiple classifier system, suppose the size of the pool of classifiers for this Toy example is $L = 5$ given by the classifiers $\{o_1, o_2, o_3, o_4, o_5\}$. Each classifier o_s is trained using its respective dissimilarity matrix, obtained by the representation set $\mathbf{R}_s$. So, the classifier o_1 is trained with the ten documents obtained with the Dissimilarity Representation having four features each as shown in Table 3, where each feature is a distance between a document in $\mathbf{X}_1$ and a document in $\mathbf{R}_1$.

After training, the class of the test example $y_j \in \mathbf{Y}$ (Table 2) is estimated by the combination of the responses of the 5 classifiers. The input of each classifier o_s is given by $D(y_j, \mathbf{R}_s)$, where $s = \{1, 2, 3, 4, 5\}$. For example, the input of the classifier o_1 for the test example y_j is given by $D(y_j, \mathbf{R}_1) = [8.185353, 5.916080, 4.123106, 6.782330]^{\mathrm{T}}$ (Eq. 4). Suppose the classifiers o_1 and o_3 classifies the test example y_j as class A, which is an incorrect answer. In contrast, let's assume o_2, o_4, and o_5 classify y_j as class B. The final response of CoDiS is given by the combination of all classifiers. So, CoDiS classifies the test example y_j as class B, since the class B has the majority of the votes (3 against 2).

It is important to remark that the original data matrix ($\mathbf{X}_1$ in Table 2) has 60% of zeros (54 zeros divided by the 90 cells), whereas the dissimilarity matrix (Table 3) has only 7.5% of zeros (3 zeros divided by the 40 cells). This is a huge reduction

in sparseness, which cannot be achieved with feature selection. The best sparseness obtained with feature selection with 4 features is only 50% (selecting w_4, w_7, w_1, and w_5). Therefore, the dissimilarity representation reduces the number of features and the sparseness.

5. Experiments

This section presents the experiments performed using 10-fold stratified cross-validation on forty-seven databases. We adopted the SVM classifier [40] with linear kernel and the size of the pool of classifiers (L) varied from 10 to 100 using ten step. The next subsections show the evaluation measures (Section 5.1), the characteristics of the databases (Section 5.2), the preliminary experiments (Section 5.3), and the final experiments (Section 5.4).

5.1. Evaluation methodology

The effectiveness was measure using micro averaged F1 (micro-F1) and macro averaged F1 (macro-F1) [35]. F1 is calculated as:

$$\text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{(\text{Recall} + \text{Precision})}.$$

However, the definitions of Recall and Precision are different for micro-F1 and macro-F1. For micro-F1 is:

$$\text{Precision}_{\text{micro}} = \frac{\sum_{k=1}^{C} TP_k}{\left(\sum_{k=1}^{C} TP_k + \sum_{k=1}^{C} FN_k\right)},$$

$$\text{Recall}_{\text{micro}} = \frac{\sum_{k=1}^{C} TP_k}{\left(\sum_{k=1}^{C} TP_k + \sum_{k=1}^{C} FP_k\right)},$$

whereas, for macro-F1 is:

$$\text{Precision}_{\text{macro}} = \frac{\sum_{k=1}^{C} \frac{TP_k}{(TP_k + FN_k)}}{C},$$

$$\text{Recall}_{\text{macro}} = \frac{\sum_{k=1}^{C} \frac{TP_k}{(TP_k + FP_k)}}{C},$$

where C is the number of classes, TP_k is the number of correctly classified documents as class c_k, FN_k is the number of incorrectly classified documents to other class but c_k and FP_k is the number of incorrectly classified documents as class c_k.

5.2. Databases

Forty-seven (47) databases were used in the experiments. Table 4 shows the number of documents, number of features and number of categories for each database. Most databases were obtained from http://sites.labic.icmc.usp.br/text_collections/ , except TDT2T30 which comes from http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html. Classic3 and WebKB4 are reduced versions of the Classic4 and WebKB datasets, respectively. Classic3 uses only three categories (CISI, CRAN, and MED) of Classic4; and WebKB4 uses only four categories (course, faculty, project, and student) of WebKB.

5.3. Preliminary experiments

This preliminary experiments should answer the following questions: i) what prototype selection algorithm should be used in the Representation Sets Generation module?; ii) what is the best measure to be employed in the Representation Transformation module (cosine similarity or Euclidean distance)?; iii) what is the best size (Q) for the representation sets in CoDiS? Therefore, this section evaluates the sensitivity of CoDiS to these three parameters: prototype selection, measure, and the size of the representation set. Six databases (CSTR, Irish, Oh0, Polarity, Tr23, and WAP) from five different domains (scientific, sentiment, medical, TREC[1], and web) were selected to perform the sensitivity analysis.

The first experiment focuses on defining the prototype selection algorithm for the Representation Sets Generation module. Based on the classification proposed by Garcia et al. [13], we applied three different categories of prototype selection algorithms: condensation, edition, and hybrid. We selected seven prototype algorithms that contemplate these three categories: condensation: CNN, RNN, IB2; edition: ENN, All-kNN; and hybrid: Drop3, and ATISA2 [5]. Random prototype selection was also considered because a previous work of dissimilarity representation [29] showed that representation sets obtained with random prototype selection can achieve good results. Therefore, two configurations were evaluated: the combination of deterministic prototype selection algorithms against the combination of non-deterministic prototype selection algorithms, i.e., random selection.

Table 4
Databases characteristics.

Database	Documents	Features	Categories
20 Newsgroup (20Ng)	18,828	45,433	20
ACM	3493	60,768	40
Classic3	3891	7749	3
Classic4	7095	7749	4
CSTR	299	1726	4
Dmoz Computers 500 (DmComp)	9500	5011	19
Dmoz Health 500 (DmHealth)	6500	4217	13
Dmoz Science 500 (DmScience)	6000	4821	12
Dmoz Sports 500 (DmSport)	13,500	5682	27
Enron Top20 (Enron20)	13,199	18,194	20
FBIS	2463	2001	17
Hitech	2301	12,942	6
Irish Sentiment (Irish)	1660	8659	3
La1s	3204	13,196	6
La2s	3075	12,433	6
LATimes	6278	10,020	6
Multi Domain Sentiment (MDS)	8000	13,360	2
New 3	9558	26,833	44
NFS	10,524	3888	16
Oh0	1003	3183	10
Oh5	918	3013	10
Oh10	1050	3239	10
Oh15	3103	54,142	10
Ohscal	11,162	11,466	10
Ohsumed-400 (Ohsumed)	9200	13,512	23
Opinosis	6457	2693	51
Re0	1504	2887	13
Re1	1657	3759	25
Re8	7674	8901	8
Reuters-21578	14,175	14,035	120
Review Polarity (Polarity)	2000	15,697	2
Reviews	4069	22,927	5
SpamAssassin (SpamAss)	9348	97,851	2
SpamTrec-3000 (Spam3000)	5982	100,464	2
SyskillWebert (SsW)	334	4340	4
TDT2T30	9394	36,771	30
Tr11	414	6430	9
Tr12	313	5805	8
Tr21	335	7903	6
Tr23	204	5833	6
Tr31	927	10,129	7
Tr41	878	7455	10
Tr45	690	8262	10
WAP	1560	8461	20
WebACE	3900	8880	21
WebKB	8282	22,892	7
WebKB4	4199	22,892	4

Table 5
Comparison of CoDiS using two approaches for the generation of the representation sets: each representation set obtained with a different prototype selection algorithm (PSA), and all representation sets obtained with random prototype selection (RPS). The best results are shown in bold and in parenthesis are the standard deviations.

Databases	Measure	PSA	RPS
CSTR	cosine	80.68 (7.42)	79.96 (7.65)
	euclidean	80.31 (5.70)	**81.25** (4.62)
Irish	cosine	**67.29** (3.83)	67.11 (2.88)
	euclidean	65.17 (3.83)	65.05 (4.07)
Oh0	cosine	85.77 (3.12)	**86.37** (3.71)
	euclidean	84.79 (3.75)	85.68 (4.12)
Polarity	cosine	82.05 (2.95)	82.40 (3.04)
	euclidean	**82.90** (1.91)	82.00 (2.46)
Tr23	cosine	74.07 (6.96)	75.29 (7.64)
	euclidean	**86.41** (8.24)	83.27 (12.12)
WAP	cosine	83.45 (2.83)	83.70 (2.46)
	euclidean	**85.28** (2.41)	**85.28** (2.71)

Table 5 shows the results obtained by the combination of the classifiers using two measures (cosine similarity and Euclidean distance). Since the interest of this preliminary experiment is to identify the best results of the combination, results of individual classifiers are not presented. The column "PSA" shows the results after combining the seven classifiers and each classifier is trained using the prototypes selected by the following algorithms: CNN, RNN, ENN, All-kNN, IB2, Drop3, ATISA2. The column "RPS" shows the results of the combination of seven classifiers that uses random prototype selection. For a fair comparison, the number of selected prototypes in the random selection was the same obtained by each prototype selection algorithm. The seven prototype selection algorithms work with five neighbors. The performance of PSA and RPS are similar based on the statistical t-test with 95% of confidence. Since prototype selection algorithms are slower than a random prototype selection, the last one is the best choice. It is important to remark that these prototype selection algorithms search for the best prototypes to represent the original database and, in general, similar prototypes are selected by different techniques. So, different classifiers trained using very similar dissimilarity matrices are not interesting for multiple classifier systems.

The second preliminary experiment evaluates which is the best measure to be used in the Representation Transformation modules: cosine similarity or Euclidean distance. Fig. 4 shows the results of CoDiS using three different sizes of the representation sets (10%, 20% and 30% of the training set) and the results of the single best classifier for each configuration (symbol +). The representation sets were obtained using random prototype selection. The single best is the classifier with the best performance in the test set.

For text categorization, it is well known that Euclidean distance is not appropriated to assess documents [34]. Cosine similarity is commonly the best choice and achieves better results than the Euclidean distance. However, this is not the case for CoDiS. As observed in Fig. 4, the results of CoDiS using cosine similarity achieved worse or similar performance when compared with the single best classifier in several databases. This behavior is only observed in ensembles with low diversity [36] and in this case, it is not worth combining the classifiers. In comparison, the combination of classifiers using Euclidean distance in the Representation Transformation module obtains better results than the combination using the cosine similarity. Moreover, in this case, the result of the single best classifier is worse than the combination using Euclidean distance in all cases. Therefore, we adopt Euclidean distance for the next experiments.

The last preliminary experiment evaluate the size of the representation sets using CoDiS and this evaluation considers both performance (micro-F1) and diversity (double fault). Among the diversity measures with good correlation to the accuracy rate, Double fault requires less computational effort [36]. Double fault is calculated as:

$$df(o_a, o_b) = \frac{O^{00}}{O^{11} + O^{10} + O^{01} + O^{00}}, \tag{5}$$

where o_a and o_b are two classifiers, and O^{ab} is the number of documents y_j in the test set $\mathbf{Y}$ that the given classifier (o_a for a and o_b for b) classifies correctly (1) or incorrectly (0). In summary, the double fault calculates the probability of both classifiers misclassifying the same document. Since double fault is a paired diversity measure (calculates the diverse between only two classifiers), the average of all pairs are used to compute the double fault of the whole ensemble.

Fig. 5 shows a comparison of three values of Q using the paired t-test. In regards to diversity, 10% is the best value in 4 out of 6 databases; whereas in regards to performance, 30% is the best value in all databases. However, 10% and 30% do not show a good performance and diversity, respectively. Therefore, the parameter is defined as 20% of the training set for having the best balance of performance and diversity.

After the preliminary experiments discussion, we adopt the following parameters for the final experiments: i) Random prototype selection to obtain the representation sets; ii) Euclidean distance to calculate the dissimilarity matrix; and, iii) the representation sets are formed using 20% of the training set, $Q = N \times 0.2$, where N is the size of the training set.

5.4. Final experiments

The proposed system, Combined Dissimilarity Spaces (CoDiS), was compared with Bagging, usually used as baseline [26,41], and Random Subspace (RSS), often used for problems with high-dimensionaly [12,48]. In the case of Bagging, a feature selection using the ALOFT [30] algorithm was performed before the pool generation. ALOFT is a feature selection algorithm designed specifically for text categorization that significantly reduces the number of variables without decreasing the performance. Two feature evaluation function were analyzed: Bi-Normal Separation [11] and Class Discriminating Measure [6]. In the case of Random Subspace, no feature selection was required because Random Subspace generates several feature subspaces of reduced dimensions for each classifier. In CoDiS, the reduction is achieved using a representation set smaller than the training set, since the final dimensionality is equal to the number of documents in the representation set. Another difference is the final number of features. In ALOFT, this number is defined in execution time, while in Random Subspace it is an input parameter that must be defined beforehand. For a fair comparison, Random Subspace uses the same number of features obtained by CoDiS (20% of the training set).

Table 6 shows the best results, and their respective number of classifiers (L), for each of the four methods: proposed system (CoDiS), Bagging using ALOFT with BNS, Bagging using ALOFT with CDM and Random Subspace. The "Average" row

[1] Text Retrieval Conference, http://trec.nist.gov/

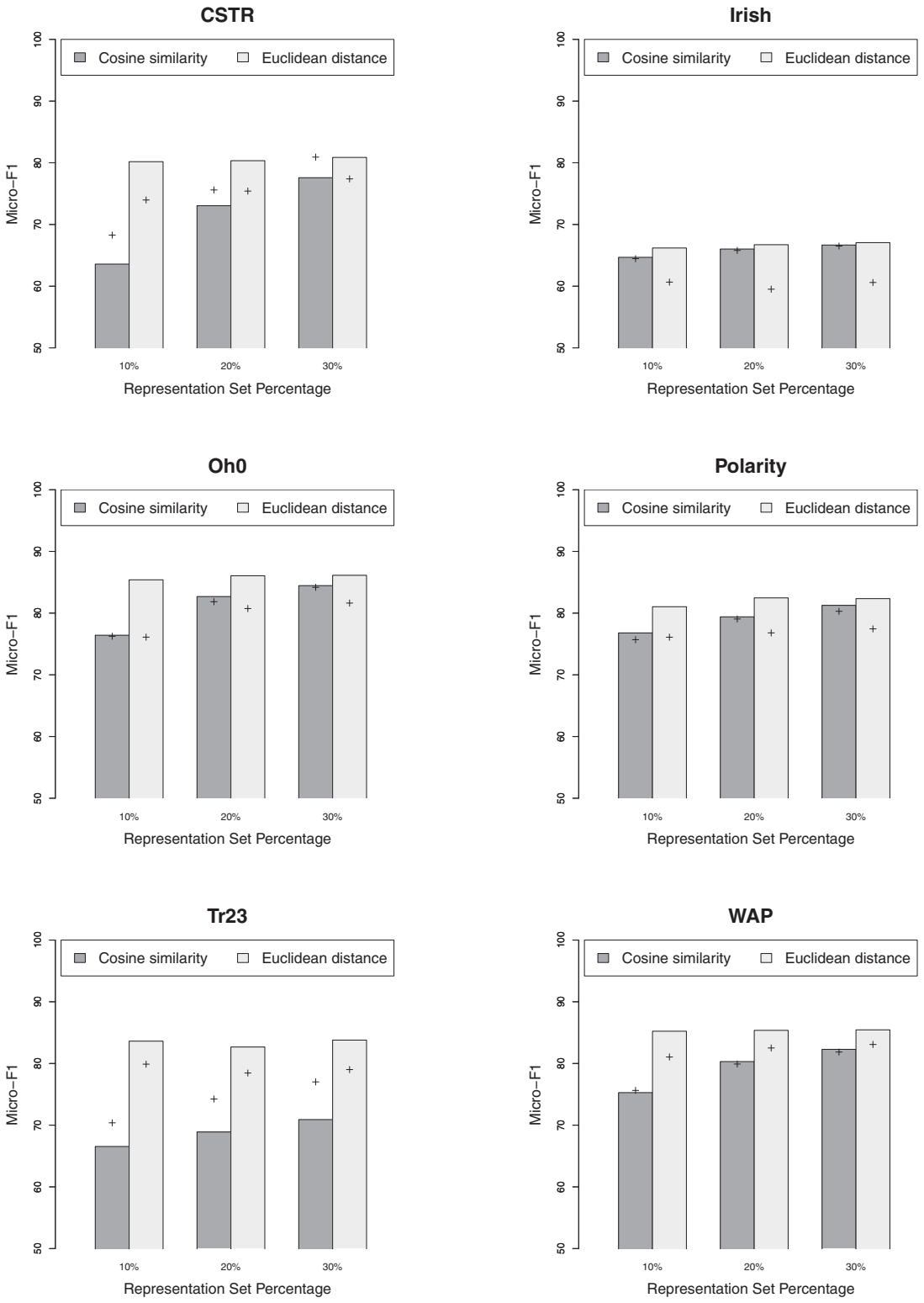


Fig. 4. Comparison of CoDiS using two different measures in the Representation Transformation module: cosine similarity and Euclidean distance. The bars show the mean value of the micro-F1 for 10 different sizes of ensembles ($L = \{10, 20, \ldots, 100\}$). The results for the single best classifier is represented by the symbol +.

Table 6
Results of the proposed system (CoDiS), Bagging (with ALOFT) and Random Subspace. Best results for each database are shown in bold.

	CoDiS			Bagging (ALOFT-BNS)			Bagging (ALOFT-CDM)			Random Subspace		
Database	*L*	macro-F1	micro-F1	*L*	macro-F1	micro-F1	*L*	macro-kF1	micro-F1	*L*	macro-F1	micro-F1
20Ng	80	84.20 (0.66)	84.30 (0.63)	100	79.90 (1.00)	80.05 (0.98)	70	81.05 (1.00)	81.15 (1.01)	90	**88.00** (0.49)	**85.59** (0.50)
ACM	100	73.65 (1.95)	72.93 (2.10)	70	39.04 (3.52)	39.06 (3.77)	80	64.99 (2.10)	64.19 (2.21)	100	**79.38** (1.70)	**76.77** (1.81)
Classic3	20	**98.75** (0.51)	**98.77** (0.48)	80	98.18 (1.12)	98.20 (1.08)	60	98.43 (0.80)	98.46 (0.77)	90	97.17 (1.01)	97.15 (1.05)
Classic4	20	**96.64** (0.43)	**96.57** (0.39)	70	95.65 (0.81)	95.48 (0.86)	40	95.87 (0.65)	95.70 (0.64)	100	95.51 (1.02)	95.19 (1.08)
CSTR	90	**83.89** (8.40)	**81.68** (7.66)	40	72.95 (11.95)	75.91 (4.60)	100	74.76 (10.79)	76.31 (7.36)	10	40.90 (14.07)	49.93 (6.85)
DmComp	100	64.76 (1.95)	64.41 (2.05)	90	63.99 (2.06)	63.66 (2.03)	100	64.01 (1.86)	63.71 (1.85)	100	**71.00** (1.34)	**71.32** (1.33)
DmHealth	50	79.04 (1.81)	78.60 (1.85)	40	79.85 (1.23)	79.23 (1.25)	20	79.32 (1.21)	78.66 (1.25)	100	**84.12** (1.07)	**83.77** (0.98)
DmScience	100	66.64 (1.98)	66.38 (1.92)	90	67.73 (1.38)	67.28 (1.40)	100	66.29 (2.11)	65.95 (2.23)	90	**70.34** (1.36)	**69.12** (1.43)
DmSport	40	84.38 (0.75)	84.19 (0.82)	40	84.88 (0.88)	84.51 (0.96)	90	84.92 (0.98)	84.64 (1.06)	100	**89.52** (0.66)	**89.24** (0.64)
Enron20	90	65.53 (1.33)	67.26 (1.07)	70	65.13 (0.69)	67.20 (0.54)	90	65.65 (1.16)	**67.69** (1.02)	100	**66.45** (0.80)	67.49 (0.63)
FBIS	60	78.64 (4.48)	84.45 (1.92)	70	72.40 (4.13)	80.45 (2.67)	80	66.25 (3.45)	77.30 (2.30)	70	**83.96** (2.78)	**86.70** (1.20)
Hitech	80	**72.16** (3.50)	**75.41** (2.67)	50	68.81 (3.92)	71.74 (2.63)	100	61.92 (3.99)	68.88 (2.99)	60	53.35 (5.18)	61.50 (3.27)
Irish	60	**65.68** (4.28)	**67.28** (4.09)	80	58.77 (3.77)	60.11 (3.70)	100	57.50 (4.11)	57.70 (3.92)	80	59.56 (4.98)	60.06 (1.43)
La1s	100	**88.02** (1.70)	**89.64** (1.81)	20	62.36 (3.12)	67.07 (1.72)	80	78.17 (2.14)	81.87 (1.55)	80	75.54 (3.10)	76.80 (2.56)
La2s	80	**89.43** (2.27)	**91.10** (1.75)	50	61.55 (2.66)	65.10 (1.95)	70	80.55 (2.57)	83.71 (2.29)	60	78.77 (1.67)	80.49 (1.58)
LATimes	70	**81.73** (1.41)	**84.38** (1.51)	20	61.84 (2.74)	66.71 (2.28)	90	76.78 (1.72)	79.93 (1.69)	90	74.08 (2.14)	73.91 (2.23)
MDS	100	**81.75** (1.17)	**81.74** (1.16)	80	81.00 (1.55)	80.96 (1.57)	90	80.57 (1.55)	80.54 (1.57)	70	79.30 (1.05)	78.99 (1.09)
New 3	90	**86.63** (1.24)	**87.26** (1.08)	80	54.81 (1.72)	57.80 (1.35)	100	67.27 (1.11)	71.43 (0.96)	100	85.49 (1.56)	84.49 (1.64)
NFS	100	78.30 (0.76)	79.42 (0.89)	40	77.26 (1.05)	78.01 (1.13)	100	76.68 (1.27)	77.63 (1.49)	100	**84.49** (0.71)	**84.20** (0.65)
Oh0	100	85.04 (3.84)	**86.63** (3.95)	20	83.59 (3.04)	84.17 (2.41)	70	**85.08** (3.09)	85.00 (2.61)	70	74.82 (4.12)	77.86 (2.84)
Oh5	80	**85.94** (4.79)	**86.17** (4.51)	70	85.56 (5.38)	86.10 (4.63)	30	83.00 (2.99)	83.32 (2.43)	90	71.72 (5.28)	70.90 (4.25)
Oh10	70	74.65 (3.12)	**77.91** (2.95)	20	**76.38** (4.08)	77.46 (3.42)	20	74.64 (4.61)	75.89 (4.79)	70	69.61 (4.12)	72.77 (4.21)
Oh15	40	77.62 (4.75)	**78.49** (4.94)	30	**78.87** (5.18)	77.87 (6.04)	70	78.69 (3.79)	78.02 (4.91)	100	66.35 (7.60)	65.86 (4.07)
Ohscal	90	**78.31** (1.08)	**79.22** (0.79)	100	77.26 (1.42)	78.06 (1.36)	90	76.38 (1.64)	77.34 (1.48)	100	77.53 (1.39)	78.38 (1.31)
Ohsumed	50	31.72 (1.20)	31.30 (1.23)	100	34.34 (0.93)	33.83 (1.02)	60	33.42 (1.12)	32.91 (1.03)	100	**35.96** (1.05)	**36.03** (1.14)
Opinosis	100	61.07 (2.16)	60.97 (1.15)	10	62.35 (2.19)	**62.70** (1.30)	30	60.91 (2.20)	61.28 (1.62)	70	**62.59** (1.87)	62.50 (1.42)
Re0	50	64.81 (8.82)	82.03 (2.29)	40	51.35 (7.92)	76.40 (3.46)	40	**71.96** (6.24)	**83.03** (2.56)	100	47.46 (10.02)	67.86 (4.27)
Re1	90	64.60 (4.12)	81.90 (2.13)	70	**78.19** (4.69)	**86.94** (2.04)	40	56.46 (5.70)	78.97 (2.25)	100	40.55 (5.11)	61.83 (2.50)
Re8	60	89.98 (2.28)	**96.32** (0.59)	60	**90.58** (2.23)	95.41 (0.42)	80	89.97 (2.41)	95.69 (0.86)	100	89.09 (2.36)	95.28 (0.52)
Reuters-21578	10	15.63 (0.89)	57.44 (1.51)	10	13.55 (0.94)	**59.36** (1.08)	20	11.47 (0.83)	58.12 (1.32)	100	**17.74** (1.21)	59.02 (1.05)
Polarity	80	**83.13** (1.69)	**83.05** (1.62)	80	76.94 (2.10)	76.90 (2.11)	20	74.32 (2.04)	74.20 (2.02)	50	77.07 (2.26)	76.10 (2.50)
Reviews	90	**93.96** (2.42)	**95.89** (1.52)	30	90.08 (2.54)	93.66 (1.27)	80	89.43 (3.00)	93.66 (1.57)	80	82.36 (1.25)	86.31 (1.27)
SpamAss	100	**98.79** (0.49)	**99.08** (0.37)	100	96.68 (0.84)	97.47 (0.63)	80	97.62 (0.56)	98.18 (0.42)	10	74.89 (1.55)	81.68 (0.98)
Spam3000	40	**98.58** (0.40)	**98.58** (0.40)	10	97.76 (0.61)	97.76 (0.60)	50	98.52 (0.45)	98.51 (0.45)	10	86.53 (1.58)	85.44 (1.78)
SsW	20	93.04 (3.34)	93.06 (3.29)	60	**94.14** (3.40)	**94.00** (3.55)	10	93.07 (4.20)	92.74 (4.53)	90	39.38 (9.64)	51.05 (6.38)
TDT2T30	90	**95.87** (1.01)	**97.19** (0.47)	50	93.91 (1.17)	96.28 (0.66)	100	92.93 (1.72)	96.10 (0.62)	50	84.14 (1.71)	86.23 (0.97)
Tr11	50	72.07 (8.12)	88.01 (4.73)	20	73.99 (6.82)	86.84 (5.27)	80	**78.98** (9.62)	**88.75** (6.01)	20	25.78 (4.72)	49.83 (4.49)
Tr12	60	82.48 (8.01)	85.58 (3.52)	70	81.47 (9.54)	**87.97** (5.55)	70	**86.21** (6.64)	87.72 (4.94)	10	15.02 (7.71)	33.22 (5.61)
Tr21	70	65.37 (11.35)	92.10 (3.65)	20	**76.12** (14.94)	**92.52** (3.74)	30	47.50 (14.03)	81.48 (7.38)	10	13.58 (0.41)	68.82 (3.53)
Tr23	70	66.50 (15.20)	84.13 (8.85)	10	**80.66** (10.09)	**91.33** (4.39)	20	60.90 (12.06)	83.63 (7.45)	10	13.99 (5.58)	46.40 (4.17)
Tr31	40	81.24 (1.52)	96.77 (1.11)	30	**81.91** (2.48)	**97.53** (1.33)	70	81.65 (2.09)	97.19 (1.30)	100	59.22 (8.71)	85.26 (4.56)
Tr41	100	**84.86** (5.09)	**94.61** (1.96)	60	72.14 (6.67)	89.78 (2.49)	50	82.41 (6.30)	93.68 (2.19)	60	63.85 (7.21)	78.92 (5.97)
Tr45	60	**85.80** (7.10)	**91.89** (3.20)	100	83.12 (5.66)	90.10 (3.09)	90	78.37 (3.47)	89.60 (2.28)	70	64.04 (3.07)	74.90 (2.63)
WAP	70	**68.30** (4.59)	**85.67** (2.83)	20	58.16 (4.73)	74.38 (3.09)	90	50.69 (4.72)	71.76 (3.94)	100	51.20 (3.28)	70.21 (3.62)
WebACE	100	**67.44** (2.61)	**89.05** (1.69)	20	7.10 (1.28)	44.70 (1.07)	50	49.81 (2.76)	76.74 (1.95)	90	52.79 (3.39)	80.70 (1.80)
WebKB	50	**72.32** (2.02)	**80.38** (1.50)	60	63.17 (2.60)	73.72 (1.50)	90	56.53 (3.08)	69.79 (0.88)	30	36.72 (1.49)	55.09 (1.10)
WebKB4	70	**90.34** (1.32)	**91.05** (1.21)	20	88.27 (1.46)	89.36 (1.30)	40	85.62 (2.11)	87.78 (1.66)	30	60.93 (2.43)	60.58 (2.10)
Average	70.85	**77.64** (15.71)	**82.98** (12.81)	52.55	72.20 (19.52)	78.10 (15.49)	66.59	73.35 (17.20)	79.71 (13.07)	72.55	64.08 (22.64)	72.16 (14.82)
win-tie-loss	-	n.a.	n.a.	-	25-17-5	25-18-4	-	27-17-3	26-19-2	-	32-4-11	34-2-11
Wilcoxon p	-	n.a.	n.a.	-	8.303e-04	2.257e-04	-	1.297e-05	1.57e-06	-	1.803e-06	4.284e-07

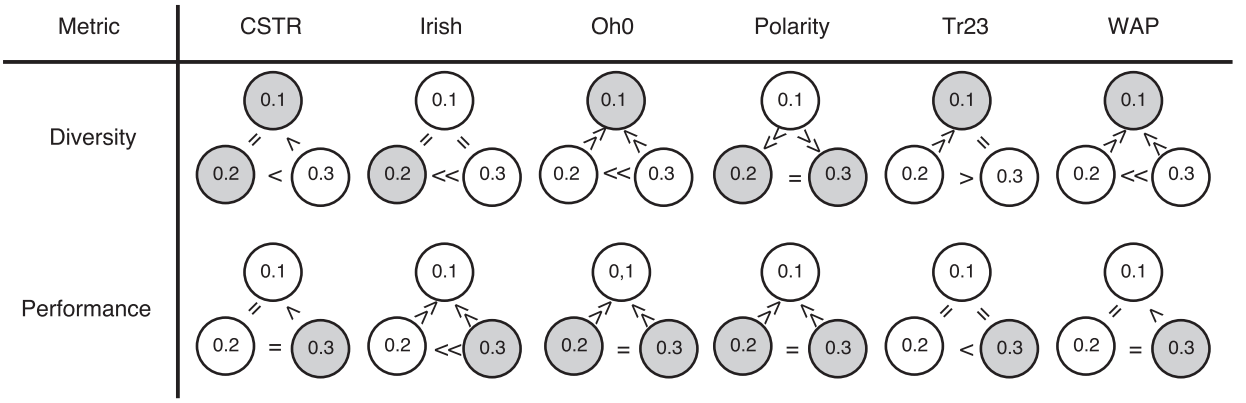


Fig. 5. Diversity (double fault) and performance (micro-F1) of CoDiS on different values of Q (10%, 20% and 30% of the training set). The symbols were defined by the paired t-test: p-value $\leq 0.01 \therefore (\gg)/(\ll)$, p-value $\leq 0.05 \therefore (>)/(<)$ and p-value $> 0.05 \therefore (=)$. The best configurations are in gray.

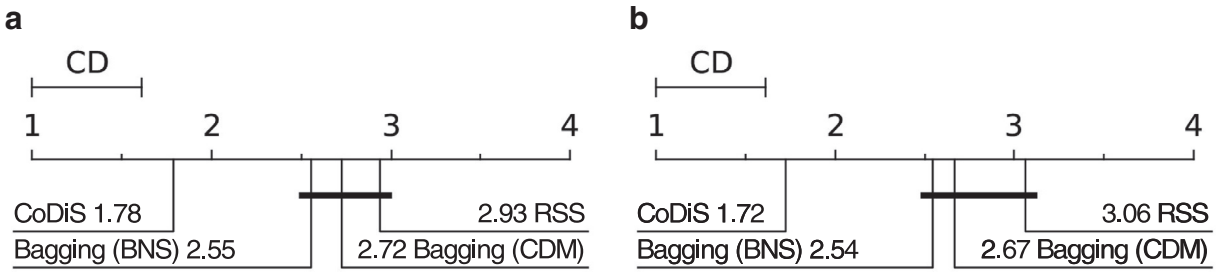


Fig. 6. Critical Difference (CD) diagrams for macro-F1 (a) and micro-F1 (b) comparing all methods used in the experiments. Average rank is presented for each method and CD = 0.6102 was obtained in both diagrams.

shows the average of L, macro-F1, and micro-F1. The last two rows compare the results of CoDiS with Bagging and Random Subspace in regards of macro-F1 and micro-F1. The "win-tie-loss" row counts how many times CoDiS was better, similar or worse than the other method based on the paired t-test results with 95% of significance. The value in the row labeled "Wilcoxon p" of the table indicates the resulting p-value of the Wilcoxon signed rank test. The proposed approach, CoDiS, is significantly better than the other ones since the application of the Wilcoxon test gives p-value close to zero. For a more extensive comparison, Friedman test was used to verify the rank of all methods considering the 47 databases. Fig. 6 shows two Critical Difference (CD) diagrams [8], one for macro-F1 and one for micro-F1. CoDiS achieved the lowest average rank in both measures and is considered statistically superior than the other methods.

Unlike CoDiS, we observed that Random Subspace performance strongly depends on the number of features obtained after the Representation Transformation (Q). Since the number of features depends on the number of documents in the training set ($Q = N \times 0.2$), the results of Random Subspace have a direct relation to N. In the majority of the cases, Random Subspace obtained the worst results with a low number of documents ($N < 2000$) and the best result with a higher number of documents ($N > 6000$). The best results are particularly noticeable in databases where the original number of features is larger than the number of documents (17% of the databases used in this paper).

The results of both approaches using Bagging are influenced by the feature selection applied before the pool generation. Bagging (ALOFT-BNS) was the second best method based on the results shown in the last two rows of Table 6.

Fig. 7 explores the relationship between the performance and the feature reduction rate. Each point on the plot represents the result of one method in one database. Given that 47 datasets and 4 methods were evaluated, a total of 188 (47 $\times$ 4) points were generated. Random Subspace performs better if the reduction rate is less than 80%. ALOFT defines the number of features automatically, and the reduction rate was 85% or more in all cases, regardless of the feature evaluation function used. The right side of Fig. 7 is a magnification of the top-right corner of its left side, showing only the results with reduction higher than 80% and micro-F1 higher than 80. In this magnified section, CoDiS appears 31 times, following by Bagging (ALOFT-CDM) with 25 appearances, Bagging (ALOFT-BNS) with 23 and, Random Subspace with 12 appearances.

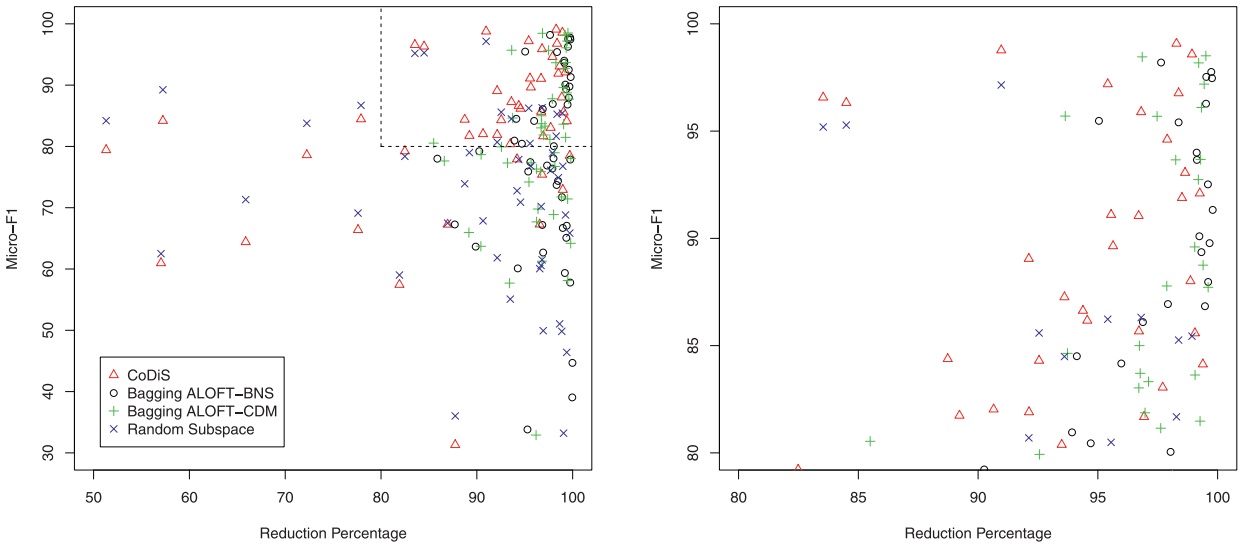


Fig. 7. Reduction rate versus performance (micro-F1). The plot at the right is a magnification of the top-right corner of the plot at the left.

6. Conclusions

Even though Bag-of-words has been widely used for text categorization, this approach presents some drawbacks because it generates high-dimensional sparse data matrix with a high feature-to-instance ratio. This paper proposed a multiple classifier system – Combined Dissimilarity Spaces (CoDiS) – for text categorization that uses dissimilarity representation to reduce these drawbacks.

On the other hand, compared to Bag-of-Words, CoDiS generates a low-dimensional non-sparse data matrix with low feature-to-instance ratio. The sparseness is reduced because each feature is represented by the distance between two documents and this distance is only zero when both documents are equal, which rarely occurs. The high-dimensionality and high feature-to-instance ratio are also reduced since the number of features is equal to the number of prototypes in the representation set, and this number is often small.

The preliminary experiments show that using random prototype selection and Euclidean distance to generate the dissimilarity matrix is more advantageous regarding precision than using prototype selection algorithms and cosine representation. This result is of importance since in a previous work we showed good results with monolithic classifiers that used a cosine representation with prototype selection algorithm [32]. In contrast, CoDiS is a multiple classifier system in which diversity plays an important role. An alternative to create diversity is to generate unstable classifiers based on random samples. Those classifiers are trained using a small representation set composed of Q random selected prototypes. The size of Q was set to 20% of the size of the training set, because of a better trade-off between diversity and accuracy. Due to a small Q value and the randomness of the selection, CoDiS generates unstable classifiers capable to improve the overall accuracy rate.

When compared with other methods in the literature, CoDiS obtained better micro-F1 and macro-F1 on forty-seven databases based on three statistical test: t-test, Wilcoxon, and Friedman. For the t-test, the comparison was done for each database, indicating that CoDiS presented better results in 59.9% of the databases. Moreover, CoDiS achieved the best balance between feature reduction rate and performance.

For future work, we plan to perform an optimization search for the best representation set per classifier to compose the final ensemble. We will also verify how the proposed approach behaves using the Dichotomy Transformation as shown in [2]. The Dichotomy Transformation is different from Dissimilarity Representation, because it generates a new document for each document in the representation set. Thus, this approach increases the number of documents of the original training set.

Acknowledgments

This research was supported by the following Brazilian agencies: CNPq (#446831/2014-0) and FACEPE (APQ-0192-1.03/14 and IBPG-0613-1.03/12).

References

[1] B. Altınel, B. Diri, M.C. Ganiz, A novel semantic smoothing kernel for text classification with class-based weighting, Knowl. Based Syst. 89 (2015) 265–277.

[2] D. Bertolini, L.S. Oliveira, E. Justino, R. Sabourin, Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers, Pattern Recognit. 43 (1) (2010) 387–396.

[3] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[4] Y. Campos, C. Morell, F.J. Ferri, A local complexity based combination method for decision forests trained with high-dimensional data, in: International Conference on Intelligent Systems Design and Applications, 2012, pp. 194–199.
[5] G. Cavalcanti, T. Ren, C.L. Pereira, Atisa: adaptive threshold-based instance selection algorithm, Expert. Syst. Appl. 40 (17) (2013) 6894–6900.
[6] J. Chen, H. Huang, S. Tian, Y. Qu, Feature selection for text classification with naive bayes, Expert Syst. Appl. 36 (3) (2009) 5432–5435.
[7] N.F. Da Silva, E.R. Hruschka, E.R. Hruschka, Tweet sentiment analysis with classifier ensembles, Decis. Support Syst. 66 (2014) 170–179.
[8] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. learn. Res. 7 (Jan) (2006) 1–30.
[9] T.G. Dietterich, Ensemble methods in machine learning, in: Multiple Classifier Systems, 2000, pp. 1–15.
[10] R.P. Duin, The combining classifier: to train or not to train? in: Pattern Recognition, 2002. Proceedings. 16th International Conference on, 2, IEEE, 2002, pp. 765–770.
[11] G. Forman, An extensive empirical study of feature selection metrics for text classification, J. Mach. Learn. Res. 3 (2003) 1289–1305.
[12] M.J. Gangeh, M.S. Kamel, R.P. Duin, Random subspace method in text categorization, in: International Conference on Pattern Recognition, 2010, pp. 2049–2052.
[13] S. Garcia, J. Derrac, J.R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.
[14] Y. Grandvalet, Bagging equalizes influence, Mach. Learn. 55 (3) (2004) 251–270.
[15] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844.
[16] Y.S. Huang, C.Y. Suen, A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1) (1995) 90–94.
[17] K. Jones, A statistical interpretation of term specificity and its application in retrieval, J. Doc. 28 (1) (1972) 11–21.
[18] S. Jun, S.-S. Park, D.-S. Jang, Document clustering method using dimension reduction and support vector clustering to overcome sparseness, Expert Syst. Appl. 41 (7) (2014) 3204–3212.
[19] J. Kittler, F.M. Alkoot, Sum versus vote fusion in multiple classifier systems, IEEE Trans. Pattern Anal. Mach. Intell. 25 (1) (2003) 110–115.
[20] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach Intell. 20 (3) (1998) 226–239.
[21] L.I. Kuncheva, Combining pattern classifiers: methods and algorithms, John Wiley & Sons, 2004.
[22] D.D. Lewis, Feature selection and feature extraction for text categorization, in: Workshop on Speech and Natural Language, 1992, pp. 212–217.
[23] L. Lin, X.-L. Wang, B.-Q. Liu, Combining multiple classifiers based on statistical method for handwritten chinese character recognition, in: Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on, 1, IEEE, 2002, pp. 252–255.
[24] S. Maldonado, R. Weber, F. Famili, Feature selection for high-dimensional class-imbalanced data sets using support vector machines, Inf. Sci. (Ny) 286 (2014) 228–246.
[25] S. Mitra, P.P. Kundu, W. Pedrycz, Feature selection using structural similarity, Inf. Sci. (Ny) 198 (2012) 48–61.
[26] A. Onan, S. Korukoğlu, H. Bulut, Ensemble of keyword extraction methods and classifiers in text classification, Expert Syst. Appl. 57 (2016) 232–247.
[27] L. Özgür, T. Güngör, Text classification with the support of pruned dependency patterns, Pattern Recognit Lett. 31 (12) (2010) 1598–1607.
[28] E. Pekalska, R.P. Duin, Dissimilarity representations allow for building good classifiers, Pattern Recognit Lett. 23 (8) (2002) 943–956.
[29] E. Pekalska, R.P. Duin, P. Paclik, Prototype selection for dissimilarity-based classifiers, Pattern Recognit. 39 (2) (2006) 189–208.
[30] R. Pinheiro, G. Cavalcanti, R. Correa, T. Ren, A global-ranking local feature selection method for text categorization, Expert Syst. Appl. 39 (17) (2012) 12851–12857.
[31] R. Pinheiro, G. Cavalcanti, T. Ren, Data-driven global-ranking local feature selection methods for text categorization, Expert Syst. Appl. 42 (4) (2015) 1941–1949.
[32] R. Pinheiro, G. Cavalcanti, T. Ren, Text categorization based on dissimilarity representation and prototype selection, in: Brazilian Conference on Intelligent Systems, 2015, pp. 163–168.
[33] R. Prabowo, M. Thelwall, Sentiment analysis: a combined approach, J. Informetrics 3 (2) (2009) 143–157.
[34] A.M. Qamar, E. Gaussier, J.-P. Chevallet, J.H. Lim, Similarity learning for nearest neighbor classification, in: International Conference on Data Mining, 2008, pp. 983–988.
[35] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (2002) 1–47.
[36] C.A. Shipp, L.I. Kuncheva, Relationships between combination methods and measures of diversity in combining classifiers, Inf. Fusion 3 (2) (2002) 135–148.
[37] J. Su, J. Sayyad-Shirabad, S. Matwin, Large scale text classification using semi-supervised multinomial naive bayes, in: International Conference on Machine Learning, 2011, pp. 97–104.
[38] M. Van Erp, L. Vuurpijl, L. Schomaker, An overview and comparison of voting methods for pattern recognition, in: Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on, IEEE, 2002, pp. 195–200.
[39] O. Van Laere, S. Schockaert, B. Dhoedt, Georeferencing flickr resources based on textual meta-data, Inf. Sci. (Ny) 238 (2013) 52–74.
[40] V. Vapnik, The nature of statistical learning theory, Berlin: Springer-Verlag, 1995.
[41] G. Wang, J. Sun, J. Ma, K. Xu, J. Gu, Sentiment classification: the contribution of ensemble learning, Decis. Support Syst. 57 (2014) 77–93.
[42] C. Whitaker, L. Kuncheva, Examining the relationship between majority vote accuracy and diversity in bagging and boosting, School of Informatics, University of Wales, Bangor, 2003.
[43] M. Wozniak, Experiments with trained and untrained fusers, in: Innovations in Hybrid Intelligent Systems, Springer, 2007, pp. 144–150.
[44] M. Woźniak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (2014) 3–17.
[45] Q. Wu, Y. Ye, H. Zhang, M.K. Ng, S.-S. Ho, Forestexter: an efficient random forest algorithm for imbalanced text categorization, Knowl. Based Syst. 67 (2014) 105–116.
[46] R. Xia, C. Zong, S. Li, Ensemble of feature sets and classification algorithms for sentiment classification, Inf. Sci. (Ny) 181 (6) (2011) 1138–1152.
[47] Y. Yang, J. Pedersen, A comparative study on feature selection in text categorization, in: International Conference on Machine Learning, 1997, pp. 412–420.
[48] H. Yu, J. Ni, An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data, IEEE Trans. Comput. Biol. and Bioinformatics 11 (4) (2014) 657–666.
[49] P. Zhang, Z. He, Using data-driven feature enrichment of text representation and ensemble technique for sentence-level polarity classification, J. Inf. Sci. 41 (4) (2015) 531–549.
[50] J. Zhu, Q. Xie, K. Zheng, An improved early detection method of type-2 diabetes mellitus using multiple classifier system, Inf. Sci. (Ny) 292 (2015) 1–14.

2016 5th Brazilian Conference on Intelligent Systems

A method for automatic determination of the feature vector size for text categorization

Rogério C. P. Fragoso, Roberto H. W. Pinheiro, George D. C. Cavalcanti*
Universidade Federal de Pernambuco (UFPE), Centro de Informática (CIn)
Av. Jornalista Anibal Fernandes s/n, Cidade Universitária 50740-560, Recife, PE, Brazil
http://www.cin.ufpe.br/~viisar
{rcpf,rhwp,gdcc}@cin.ufpe.br
*Telephone: +55 81 2126-8430 Ext.4346 Fax: +55 81 2126-8438

***Abstract*—In this paper, we propose a feature selection method for text categorization based on the filtering approach named Automatic Feature Subsets Analyzer (AFSA). The AFSA extends the Class-dependent Maximum Features per Document (cMFDR) algorithm and automatically defines the best number of features per document. In the cMFDR algorithm, the number of features is selected after a repetitive application of the methods which is a time-consuming strategy. In contrast, AFSA finds the best number of features in a data-driven way which is faster than cMFDR. The experiments with the Naïve Bayes Multinomial classifier, using four benchmark datasets, and three Feature Evaluation Function showed that the AFSA outperforms or presents similar results when compared with the cMFDR.**

I. Introduction

The increasing availability of text documents in digital form is leading to a need for easy, flexible and automated ways of accessing their contents. In this context, text categorization (TC) is a crucial tool for content organization and information retrieval. Text categorization can be defined as the task of determining the category or class that a text document belongs to.

Most of the text categorization techniques use the Bag of Words for representation. In this approach, each word of a textual document is considered a feature. Thereby, it is common for a medium sized dataset have tens of thousands of features, what makes text categorization a high dimensional problem [1], [2], [3]. Computational costs of categorization are directly affected by the dataset's dimensionality. Furthermore, the excessive number of features can negatively impact on the classification accuracy, especially in datasets having a small number of instances. As most of the features in text documents are redundant or irrelevant for classification [4], these issues can be addressed by restricting the number of features in the dataset, approach known as dimensionality reduction. Reducing the number of features in the dataset can improve computational efficiency while maintaining or enhancing classification performance [5], [6].

Two main techniques are used for dimensionality reduction: feature extraction and feature selection [7], [4]. Both techniques aim to reduce the number of features in the dataset maintaining or improving the discriminatory power. Feature extraction consists of creating a new set of features composed of synthetic features extracted from the combination of the original features [7], [8]. Feature selection is the process of generating a new subset of features containing the most discriminatory features of the original set [9].

Feature selection may be made through wrapper methods, filter methods or a hybrid strategy [10]. Wrapper methods [11] consist in creating, randomly or rule-based, several subsets of the original feature space and determining the best one by testing and analyzing the classification accuracy rates of each subset. In other words, using wrapper methods, the classifier is executed as many times as subsets are built. Since every possible subset may be tested, it is theoretically possible, for wrapper methods to achieve the best result. However, this approach leads to a very high computational cost and in high dimensional problems, such as TC, this process can be infeasible. Filter methods [3], [12] use statistical metrics and deterministic algorithms to select the most discriminatory features. Then, a subset containing the selected features is built. Thereby, filter methods do not require interaction with classifiers during the feature selection process. For this reason, they demand lower computation efforts selecting a satisfactory feature subset. Thus, filter methods represent the most commonly adopted strategy for dimensionality reduction in TC problems. The hybrid strategy [13], [14] uses filter methods to diminish the space of solutions, i.e., all the possible subsets, then apply a wrapper method to search for the best solution in this reduced space of solution. This mixed strategy aims to combine the low execution cost of filter methods with the good results of wrapper methods. This work concentrates on feature selection methods based on filtering.

The classical filter method for feature selection, known as Variable Ranking [15], [16], uses Feature Evaluation Function (FEF) to measure the degree of significance for each feature in the dataset. Based on this information, the features are globally ranked, and the top features (m provided by the user) are selected to be present in the new subset. So, the method requires an FEF and a positive integer m as parameters. Lewis and Ringuette [16] introduced this strategy using Information Gain [3] as FEF. Many other FEFs of general purposes, such as Chi-Squared [3] and Gini Index [17] were also combined with Variable Ranking reporting good results. Some other FEFs, such as Bi-Normal Separation [18], Improved Gini Index [17], Class Discriminating Measure [5],

978-1-5090-3566-3/16 $31.00 © 2016 IEEE
DOI 10.1109/BRACIS.2016.45

were designed specifically for text categorization problems.

In this work, we present a feature selection method based on filtering approach, called Automatic Feature Subsets Analyzer (AFSA). The proposed method extends the cMFDR method introducing an automatic procedure to determine the best value for the parameter f that defines the number of features selected per document. In the cMFDR method, the best value for the f parameter is selected after repetitive application of the method with different values of f. This is a manual and time-consuming approach. On the other hand, the AFSA method uses an automatic procedure that searches for the best value of f using a validation set. AFSA generates few subsets using cMFDR, where each subset has a different value for the f parameter. After, the algorithm assesses the performance of each subset and selects the most promising one. This strategy has proved to be more efficient regarding computational effort than the non-informed search done by the cMFDR.

This paper is organized as follows. Section II reviews the previous feature selection algorithms that inspired the proposed method. Section III presents the proposed method. Section IV describes the experiments and discusses the results. Section V closes the paper with conclusions and future works.

II. Previous Works

This section describes the feature selection methods that preceded AFSA: At Least one FeaTure (ALOFT) [19], Maximum Features per Document (MFD) [20], Maximum Features per Document - Reduced (MFDR) [20] and Class-dependent Maximum Features per Document (cMFDR) [21]. All these methods use the text representation known as Bag-of-Words, in which every word in the dataset is considered a feature. Each document is represented as a features vector, where each feature is the number of occurrences of the given feature in the document. Since the number of features in a typical dataset is far bigger than the number of features present in a typical text document, most of the features in a document have zero occurrences. Therefore, a document in Bag-of-Words representation is a sparse vector. The methods described in this section focus on selecting valued features, meaning feature present in a certain document, to avoid sparsity.

At Least one FeaTure [19] is a filter method that determines the number of selected features in a data-driven way, assuring the contribution of every document in the dataset. ALOFT uses a Feature Evaluation Function (FEF) to rank the features and then, for each document, chooses the valued feature with the highest rank. The method achieves a high dimensionality reduction because many documents usually present the same top-ranked feature. Thus, the final subset contains way fewer features than the number of documents. The performance of ALOFT is better than the classical Variable Ranking method [19]. However, the restriction of selecting only one feature per document may hinder the performance since the number of features may be insufficient to discriminate all categories.

After ALOFT, Pinheiro et al. [20] proposed Maximum Features per Document (MFD). MFD requires a parameter f, that stands for the number of features to be selected in a document. This parameter addresses the limitation of selecting only one feature per document, present in ALOFT. The algorithm computes and ranks the FEF value for each feature in the dataset. Then, the method selects the f top-ranked valued features for each document to compose the final feature set. MFD selects more features per document, achieving better results than ALOFT. However, in some cases, the number of features required to achieve its best effectiveness may be high, depending on the combination of dataset and FEF used.

Thus, Maximum Features per Document - Reduced (MFDR) is proposed to restrict the number selected features [20]. MFDR adopts a threshold to determine which documents are important to the feature selection process. For the threshold computation, first, the algorithm calculates the document relevance (DR) as the sum of the FEF of its valued features. Then, the threshold is computed as the mean DR. Only documents with DR above the threshold are considered in the feature selection algorithm. Then, the f valued features with highest FEF are selected for each document that overcomes the threshold. In this way, MFDR excludes documents containing low FEF valued features and reduce the size of the final feature set. MFDR achieves similar results in comparison to MFD with fewer features. However, MFDR may disregard documents belonging to categories in which documents present low FEF valued features, leading to under-representation of such categories in the final feature set.

Fragoso et al. [21] introduced Class-dependent Maximum Features per Document (cMFDR) method to treat the issues of MFDR. cMFDR improves the way the threshold is calculated to take into account particularities of categories. cMFDR computes a threshold for each category as the mean DR of the documents belonging to it. Therefore, the documents are analyzed according to the specifications of the category it belongs to. Another improvement introduced by cMFDR is the calculation of DR. The original formula to calculate DR [20] benefits large documents, i.e., containing a big number of valued features. Thus, features with high FEF value present in small documents may be ignored. To address this, cMFDR computes DR as the mean FEF values of valued features in the document. This assures a fair comparison basis. Fragoso et al. [21] reported that cMFDR selects more features than MFDR for the same value for f parameter, but cMFDR achieves better performance than its predecessor even with lower f, when it selects a similar number of features compared with the best subsets generated with MFDR.

III. Proposed Method

In this work, we propose a feature selection method based on filtering approach that dynamically determines the best value for the f parameter, i.e., the number of features to be selected per document. cMFDR method presents good results [21], however, as its predecessors [20], it requires a

Algorithm 1 Automatic Feature Subsets Analyzer

Require: $n \geqslant 1 \in \mathbb{N}$

1: Load dataset $\mathcal{D}$
2: Split documents into Training ($\mathcal{D}_{tr}$), Validation ($\mathcal{D}_{val}$) and Test ($\mathcal{D}_{tt}$)
3: Set $Results$ an empty vector
4: **for** $f = 1$ to n **do**
5: Select features using cMFDR with f
6: Build $\mathcal{D}_{tr}(f)$ and $\mathcal{D}_{val}(f)$ subsets based on the selected features
7: Build classifier $\mathcal{C}_f$ using $\mathcal{D}_{tr}(f)$
8: Assess precision using $\mathcal{D}_{val}(f)$ and store it in $Results_f$
9: **end for**
10: $bestScore$ = 0
11: **for** $f = 1$ to n **do**
12: **if** $Results_f > bestScore$ **then**
13: $bestScore = Results_f$
14: $bestF = f$
15: **end if**
16: **end for**
17: Build $\mathcal{D}_{tt}(bestF)$
18: Assess final precision using $\mathcal{C}_f$ and $\mathcal{D}_{tt}(bestF)$

value for the parameter f. In a production environment, as new documents are presented, it is necessary, from time to time, to reanalyze the dataset including the new documents in the feature selection process. Every time this analysis is performed, the user needs to test manually the subsets generated by cMFDR and choose the best one. So, in a production environment, several values of the parameter f must be evaluated to determine which one generates the best subset. This process is time-consuming and requires human interaction. Automatic Feature Subsets Analyzer (AFSA), the proposed method, aims to establish, in a data-driven way, the best value for f parameter and, hence, the number of features to be selected. The proposed method adopts a validation set strategy to generate a series of subsets using cMFDR, pre-assesses the performance of each set and then choose the value for f that produces the most effective set. So, the best value of f is provided as input for cMFDR to build the final feature set, using the test set. Algorithm 1 shows the pseudo-code of the AFSA method.

AFSA method requires a value for the parameter n, which is the number of subsets to be generated and pre-assessed. The provided dataset $\mathcal{D}$ is partitioned into $\mathcal{D}_{tr}$, $\mathcal{D}_{val}$ and $\mathcal{D}_{tt}$, standing for training, validation, and test sets, respectively (line 2). The method uses 80% of the documents for training, 10% for validation and 10% for testing. cMFDR receives as input $\mathcal{D}_{tr}$, performs the feature selection, and generates a new subset for each value of the f parameter provided, f varying from 1 to n (lines 5 and 6). $\mathcal{D}_{val}$ is used to assess the n generated subset and to store the result in the $Results$ vector (line 8). After, each value in the $Results$ vector is analyzed to determine if the generated subset has the best classification results. The f parameter that produced the best accuracy using the validation set is stored in $bestF$ (lines 11 to 16). AFSA method performs this task in an automatic fashion without human interaction. The algorithm generates a test set using $bestF$ (line 17). After, the classifier $\mathcal{C}_{bestF}$ is used to evaluate the documents in the test set (line 18).

IV. Experiments

A. Experimental Settings

In this subsection, we describe the configurations of the experiments carried out to assess the performance of the proposed method. The performance of AFSA method is examined against cMFDR using Naïve Bayes Multinomial, that is acknowledged as a good classifier for text categorization [22] and is very influenced by the selected features [19].

Micro Averaged F1 and Macro Averaged F1 [7], [23] were the performance measures used. Macro Averaged F1 is computed giving the same weight to all categories in the dataset. Thus, averaging favors the performance of rare categories. On the other hand, Micro Averaged F1 weights each category according to its size and, hence is dominated by the performance of larger categories.

Since AFSA and cMFDR methods require a Feature Evaluation Function, three FEFs that reported good results for multi-class problems [19] were used. They are described as follows.

Bi-Normal Separation [18]:

$$\mathrm{BNS}(w) = \sum_{j=1}^{C} \left| F^{-1}(P(w|c_j)) - F^{-1}(P(w|\bar{c_j})) \right|, \quad (1)$$

Class Discriminating Measure [5]:

$$\mathrm{CDM}(w) = \sum_{j=1}^{C} \left| \log \frac{P(w|c_j)}{P(w|\bar{c_j})} \right|, \quad (2)$$

and Chi-Squared [3]:

$$\mathrm{CHI}(w) = \sum_{j=1}^{C} \frac{\left[P(w|c_j)P(\bar{w}\bar{c_j}) - P(w|\bar{c_j})P(\bar{w}|\bar{c_j})\right]^2}{P(w)P(\bar{w})P(c_j)P(\bar{c_j})}, \quad (3)$$

where w is a feature, c_j is the jth category and $P(\cdot)$ is the probability of occurrence.

The experiments were conducted on four benchmark datasets: a subset of WebKB corpus, composed of the four top categories; a subset of Reuters-21578 Collection formed by the 10 larger categories; 20 Newsgroup corpus; and a subset of TDT2 containing the top 30 categories. These datasets present distinct characteristics concerning the subject, the size and the distribution of categories, the number of words etc. These differences are important to analyze the behavior of the proposed method in different environments. The subset of WebKB corpus used in the experiments is composed of 4199 documents, 4 categories and 7770 words and. Reuters 10, the subset of Reuters-21578 used in this work, contains 10987 words and 9980 documents distributed among 10 categories. 20 Newsgroup is formed by 70216 words, 18821 documents

and 20 categories. TDT2 top 30 contains 9394 documents, 36093 words and 30 categories.

The experiments were performed with 10-fold stratified cross-validation. For AFSA we used one fold for validation, one fold for test and eight folds for training while for cMFDR we used one fold for test and nine folds for training. Thus, AFSA used less training data than cMFDR.

cMFDR method requires a value for the parameter f, which means the number of features to be selected by document. AFSA requires the range of f (parameter n). Thus, to make a fair comparison, in the experiments carried out in this work, we examined cMFDR with f varying from 1 to n. Fragoso et al. [21] demonstrated that cMFDR achieves satisfactory results using f parameter ranging from 1 to 10. Thus, we adopt $n = 10$ in the experiments. AFSA is only compared with cMFDR because a previous work showed that cMFDR obtains better results than state-of-the-art methods [21].

B. Experimental Results

In this subsection, we present the results of the experiments. Table I presents the results of experiments with AFSA method in four datasets using three FEFs. The performance is measured with Micro and Macro-averaged F1. The results are compared with the best results of cMFDR method for each combination of FEF and dataset. Note that in AFSA, the value of f is dynamically computed.

The observation of the results shows that the performances of both methods are similar, regarding Micro-F1, Macro-F1 and number of selected features, for every combination of dataset and FEF. In most cases, AFSA selected a lower number of features than the best result of cMFDR with BNS and CHI for the four datasets analyzed, except using CHI in TDT2. With CDM, AFSA selected more features than the best result of cMFDR, except in 20 Newsgroup. cMFDR presented a slightly better performance both for Micro-F1 and Macro-F1 with most of the combinations of dataset and FEF, except with Reuters and CHI, when AFSA performed better regarding Macro-F1.

Figures 1 and 2 exhibit the performance of AFSA concerning Micro and Macro F1, respectively, for each dataset using the three FEFs, compared with the results of cMFDR with f ranging from 1 to 10. As AFSA automatically determines the best value for the f parameter, only the final result is plotted as a dashed line.

AFSA performed better than cMFDR with most of the values of f parameter experimented for both Micro and Macro F1. For instance, in 20 Newsgroup dataset, the result of AFSA using CHI as FEF is better than cMFDR in 9 of 10 values of f parameter, both for Micro and Macro F1. The best results were obtained with BNS while CDM was the worst FEF for AFSA. In 20 Newsgroup, AFSA achieved the biggest advantage. WebKB was the dataset that presented the most evenly matched results.

We used the statistical t-test of combined variance for Micro-F1 and Macro-F1 to compare the performance of AFSA against cMFDR. The t-test is applied on the average and the standard deviation of the Micro-F1 and Macro-F1 obtained by each method in the test set on a 10-fold cross-validation experiment [19]. Table II presents the results of t-test executed comparing AFSA against cMFDR. For this experiment, we adopt the following convention for P-value: symbols $>$ and $<$ mean that the P-value is lesser than or equal to 0.05, indicating some evidence that AFSA presents a greater or minor value for the effectiveness measure than cMFDR; symbol $\sim$ means that the P-value is greater than 0.05, indicating no significant difference between effectiveness of both methods.

The results presented in Table II show that the performance of AFSA is similar or better than its predecessor in 100% of the cases. The results were equivalent in 22 out of 24 (91.7%) cases. AFSA performs better in 2 out of 24 (8.3%) cases.

V. Conclusion

This work proposed a filtering feature selection method named Automatic Feature Subsets Analyzer (AFSA). The method is based on cMFDR method and introduces the improvement of the automatically determination of the number of selected features (m), through the cMFDR's f parameter. The method builds n validation subsets using cMFDR algorithm, adopting f from 1 to n. Then, AFSA pre-assesses the performance of these sets and chooses the value for f in which the best performance is achieved. The best value of f is provided as input for cMFDR to build the final feature set.

The proposed method achieves similar or better results than cMFDR in all the tested cases, regardless of dataset or feature evaluation function. It is important to emphasize that AFSA achieves these results using less training data than cMFDR since it needs a validation set. Furthermore, AFSA selects a similar number of features of the best subsets of its predecessor, cMFDR.

Acknowledgment

This research was partially supported by the following Brazilian agencies: CNPq (#446831/2014-0) and FACEPE (APQ-0192-1.03/14).

References

[1] G. Feng, J. Guo, B.-Y. Jing, and T. Sun, "Feature subset selection using naive bayes for text classification," *Pattern Recognition Letters*, vol. 65, pp. 109–115, 2015.

[2] D. Fragoudis, D. Meretakis, and S. Likothanassis, "Best terms: an efficient feature-selection algorithm for text categorization," *Knowledge and Information Systems*, vol. 8, no. 1, pp. 16–33, 2005.

[3] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," in *International Conference on Machine Learning*, 1997, pp. 412–420.

[4] H. Uğuz, "A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm," *Knowledge-Based Systems*, vol. 24, no. 7, pp. 1024–1032, 2011.

[5] J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with Naive Bayes," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5432–5435, 2009.

[6] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[7] F. Sebastiani, "Machine learning in automated text categorization," *Association for Computing Machinery Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

TABLE I

MICRO-F1 AND MACRO-F1 OF AFSA AND CMFDR'S BEST CONFIGURATION WITH THE NUMBER OF SELECTED FEATURES (m) AND PARAMETER f. STANDARD DEVIATION IN PARENTHESES.

Dataset	FEF	cMFDR				AFSA			
		f	m	Micro-F1	Macro-F1	f	m	Micro-F1	Macro-F1
WebKB	BNS	9	382 (9.69)	84.19 (0.78)	83.07 (0.99)	9	366 (13.92)	84.04 (1.01)	83.00 (1.45)
	CDM	4	862 (11.86)	86.45 (2.05)	84.98 (2.14)	5	991 (25.01)	86.07 (1.47)	84.58 (1.83)
	CHI	3	142 (3.33)	84.47 (1.68)	83.68 (1.88)	3	137 (5.58)	84.35 (1.84)	83.44 (1.91)
Reuters	BNS	10	825 (36.82)	81.39 (1.03)	66.26 (1.41)	10	781 (25.15)	81.05 (0.95)	65.60 (1.36)
	CDM	5	680 (9.72)	82.43 (0.72)	67.68 (1.08)	6	775 (9.97)	82.09 (0.71)	66.84 (1.77)
	CHI	10	982 (13.06)	81.44 (1.15)	66.22 (2.16)	8	683 (9.03)	81.23 (0.98)	66.34 (2.03)
20 Newsgroup	BNS	10	5937 (70.39)	85.85 (0.63)	85.44 (0.64)	10	5544 (93.81)	85.42 (0.67)	85.03 (0.67)
	CDM	10	5081 (21.86)	86.07 (0.45)	85.72 (0.51)	10	4707 (29.56)	85.44 (0.62)	85.07 (0.71)
	CHI	10	2621 (19.04)	83.34 (0.61)	82.93 (0.66)	10	2521 (25.42)	83.09 (0.62)	82.67 (0.70)
TDT2	BNS	10	1440 (19.69)	96.61 (0.44)	96.65 (0.35)	9	1089 (86.12)	96.38 (0.58)	96.23 (0.30)
	CDM	9	1354 (10.12)	96.55 (0.40)	96.52 (0.41)	10	1456 (10.77)	96.39 (0.38)	96.28 (0.38)
	CHI	9	1640 (15.26)	96.72 (0.36)	96.73 (0.49)	10	1805 (28.56)	96.68 (0.42)	96.65 (0.49)

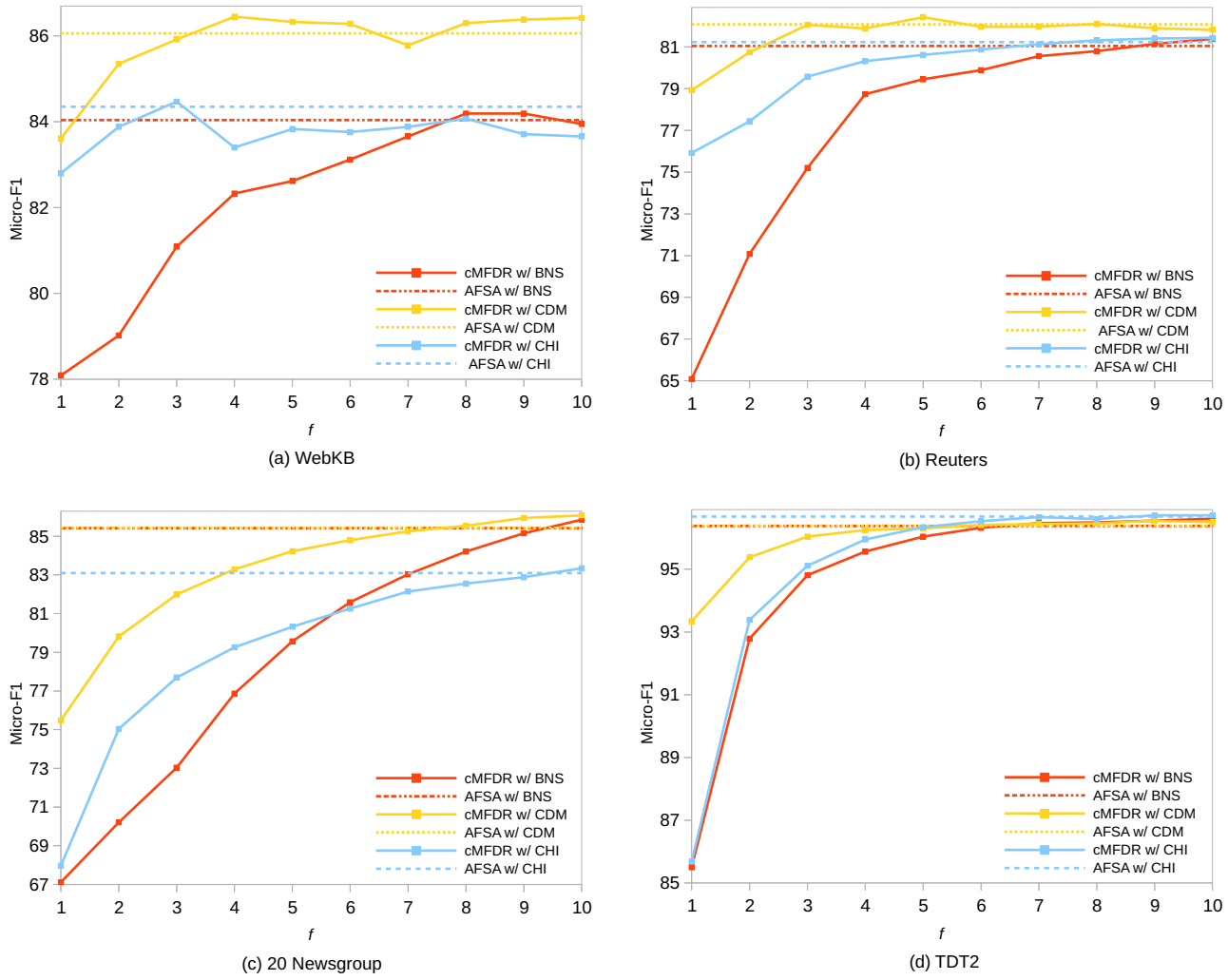


Fig. 1. Results in terms of Micro-averaged F1 for AFSA and cMFDR, with f parameter varying from 1 to 10.

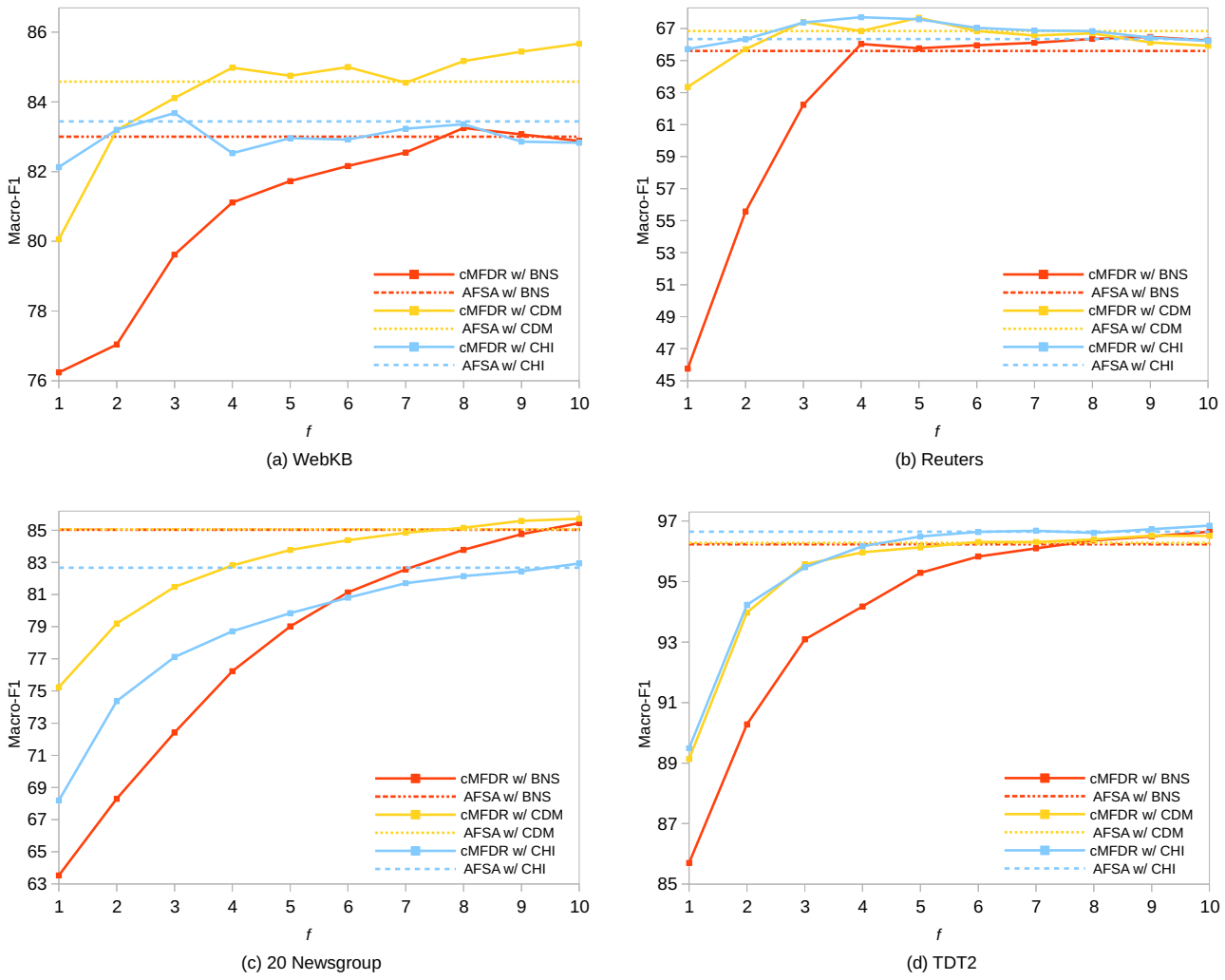
[8] J. Yang, Y. Liu, X. Zhu, Z. Liu, and X. Zhang, "A new feature selection based on comprehensive measurement both in inter-category and intra-category for text categorization," *Information Processing & Management*, vol. 48, no. 4, pp. 741–754, 2012.

[9] X. Sun, Y. Liu, M. Xu, H. Chen, J. Han, and K. Wang, "Feature selection using dynamic weights for classification," *Knowledge-Based Systems*, vol. 37, pp. 541–549, 2013.

[10] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[11] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[12] L. Yu and H. Liu, "Feature selection for high-dimensional data: a fast correlation-based filter solution," in *International Conference on*



(a) WebKB (b) Reuters

(c) 20 Newsgroup (d) TDT2

Fig. 2. Results in terms of Macro-averaged F1 for AFSA and cMFDR, with f parameter varying from 1 to 10.

TABLE II
AFSA VS. CMFDR: T-TEST RESULTS

Dataset	Measure	FEF		
		BNS	CDM	CHI
WebKB	Micro-F1	$\sim$	$\sim$	$\sim$
	Macro-F1	$\sim$	$\sim$	$\sim$
Reuters	Micro-F1	$\sim$	$\sim$	$\sim$
	Macro-F1	$\sim$	$\sim$	$\sim$
20 Newsgroup	Micro-F1	$\sim$	$>$	$\sim$
	Macro-F1	$\sim$	$>$	$\sim$
TDT2	Micro-F1	$\sim$	$\sim$	$\sim$
	Macro-F1	$\sim$	$\sim$	$\sim$

Machine Leaning, vol. 20, no. 2, 2003, pp. 856–863.

[13] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in *International Conference on Machine Learning*, vol. 1, 2001, pp. 74–81.

[14] E. P. Xing, M. I. Jordan, R. M. Karp *et al.*, "Feature selection for high-dimensional genomic microarray data," in *International Conference on Machine Learning*, vol. 1, 2001, pp. 601–608.

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

[16] D. Lewis and M. Ringuette, "A comparison of two learning algorithms for text categorization," in *Annual Symposium on Document Analysis and Information Retrieval*, vol. 33, 1994, pp. 81–93.

[17] W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang, "A novel feature selection algorithm for text categorization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 1–5, 2007.

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

[19] R. Pinheiro, G. Cavalcanti, R. Correa, and T. Ren, "A global-ranking local feature selection method for text categorization," *Expert Systems with Applications*, vol. 39, no. 17, pp. 12 851–12 857, 2012.

[20] R. Pinheiro, G. Cavalcanti, and T. Ren, "Data-driven global-ranking local feature selection methods for text categorization," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1941–1949, 2015.

[21] R. Fragoso, R. Pinheiro, and G. Cavalcanti, "Class-dependent feature selection algorithm for text categorization," in *International Joint Conference on Neural Networks*, 2016.

[22] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Workshop on Learning for Text Categorization*, 1998, pp. 41–48.

[23] J. Zhang and Y. Yang, "Robustness of regularized linear classification methods in text categorization," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003, pp. 190–197.

Combinando Classificadores Binários no Espaço de Dissimilaridade para Categorização de Documentos

Roberto H. W. Pinheiro[1], **George D. C. Cavalcanti**[1], **Tsang Ing Ren**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Av. Jornalista Anibal Fernandes s/n, Cidade Universitária – 50.740-560 – Recife – PE – Brazil

{rhwp,gdcc,tir}@cin.ufpe.br

Abstract. *Text Categorization is a high-dimensional Machine Learning problem. An alternative to deal with high-dimensionality is the Random Subspaces method, which divides the feature space to several classifiers. However, this division is completely random, leading to loss of performance even with the combination of the classifiers. In this paper is proposed a Combination of Classifiers method which maintains the randomness without reducing the number of features. This method is based on the Dichotomization approach, which is possible to generate random sets to train the classifiers in different dissimilarity spaces. Experiments over 16 databases show that the proposed method, especially using similarity, achieves good results in comparison to other methods of classifiers' combination.*

Resumo. *Categorização de Documentos é um problema de Aprendizagem de Máquina com alta dimensionalidade. Uma alternativa capaz de lidar com a alta dimensionalidade é o método Random Subspace (Subespaços Aleatórios), pois esse algoritmo divide o espaço de característica entre vários classificadores. Entretanto, essa divisão é feita de modo completamente aleatório, podendo causar prejuízos à classificação mesmo com a combinação de classificadores. Nesse artigo é proposto um método de Combinação de Classificadores que mantém a aleatoriedade sem reduzir a quantidade de características. Esse método baseia-se em uma abordagem chamada Dicotomização, no qual é possível gerar conjuntos aleatoriamente para treinar os classificadores em diferentes espaços de dissimilaridade. Os experimentos realizados sobre 16 bases de dados mostram que o método proposto, especialmente utilizando similaridade, atinge bons resultados em comparação a outros métodos de combinação de classificadores.*

1. Introdução

Dado o crescimento do acesso à informação textual por meio da *Internet*, sistemas voltados para facilitar a organização dessas informações são importantes. A Categorização de Documentos é um desses sistemas, cujo objetivo é atribuir uma classe, pertencente a um universo finito e pré-estabelecido, a um determinado texto. Algumas aplicações relevantes de Categorização de Documentos são: organização de documentos em tópicos, detecção de *spam* em *e-mails*, sistemas de recomendação e análise de sentimento.

A Categorização de Documentos utiliza Aprendizagem de Máquina para atingir seus objetivos, sendo o *Support Vector Machine* (SVM), considerado o classificador

estado-da-arte [Pang and Jiang, 2013, Rossi et al., 2014]. Entretanto, nenhum classificador é capaz de resolver os problemas em qualquer base de dados [Wolpert, 2002]. Portanto, várias aplicações passaram a combinar múltiplos classificadores [Campos et al., 2012, Yu and Ni, 2014, Ahn et al., 2007, Pathical and Serpen, 2010, Piao et al., 2014], para atingir maior robustez e eficiência [Yang et al., 2000]. A combinação de classificadores baseia-se na premissa de unir classificadores com capacidades diferentes, se possível complementares, pois a união de classificadores distintos pode realçar suas qualidades individuais e reduzir suas fraquezas. Logo, espera-se que uma combinação melhore o desempenho do sistema como um todo [Bi et al., 2004].

Como os problemas de Categorização de Documentos possuem milhares de características, o *Random Subspace* [Ho, 1998] é a abordagem mais adotada, pois é efetiva para problemas com alta dimensionalidade [Piao et al., 2014, Yu and Ni, 2014]. No *Random Subspace* cada classificador é treinado com um subconjunto aleatório das características originais. Assim, cada classificador poderá lidar com um diferente espaço de características reduzido e, ao final, combinar a resposta de todos os classificadores. Gangeh et al. [2010] constatou que a utilização do *Random Subspace* em problemas de Categorização de Documentos pode melhorar o desempenho com relação à classificadores individuais estado-da-arte, como o *Support Vector Machines*.

Entretanto, o *Random Subspace* trabalha de modo completamente aleatório. A aleatoriedade é necessária para atingir a diversidade, fator chave para o funcionamento adequado de uma combinação de classificadores [Dietterich, 2000b]. Entretanto, aleatoriedade não é sinônimo de diversidade, pois é possível gerar classificadores que mesmo combinados obtenham resultados inferiores em comparação a classificadores individuais [Shipp and Kuncheva, 2002].

Neste artigo, é apresentado um método de Combinação de Classificadores que mantém a aleatoriedade sem reduzir a quantidade de características. Esse sistema baseia-se em uma abordagem chamada Dicotomização, no qual um problema inicialmente multiclasse é transformado para um problema binário (duas classes). Essa abordagem requer um conjunto de representação, cuja composição é de documentos referência do conjunto de treinamento original, isto é, documentos de todas as classes que cubram todo o espaço do problema. No método proposto, os Conjuntos de Representação foram gerados com amostragens aleatórias do conjunto de treinamento, a fim dos classificadores serem treinados com dados diferentes. A Combinação de Classificadores foi utilizada para evitar a busca pelo melhor conjunto de representação, pois ele supera o problema do ótimo local ao combinar condições iniciais diferentes [Dietterich, 2000a]. Espera-se que a combinação de classificadores treinados em diferentes espaços de dissimilaridade incremente o desempenho de todo o sistema.

Este artigo é organizado da seguinte maneira: Seção 2 apresenta o método proposto. Seção 3 descreve a metodologia dos experimentos, reporta e analisa os resultados experimentais. A Seção 4 conclui o artigo.

2. Método Proposto

O método proposto é um sistema de Categorização de Documentos que utiliza Combinação de Classificadores gerados por Dicotomizadores. Para realizar a tarefa desejada, o método proposto utiliza-se dois principais conjuntos: o conjunto de treinamento

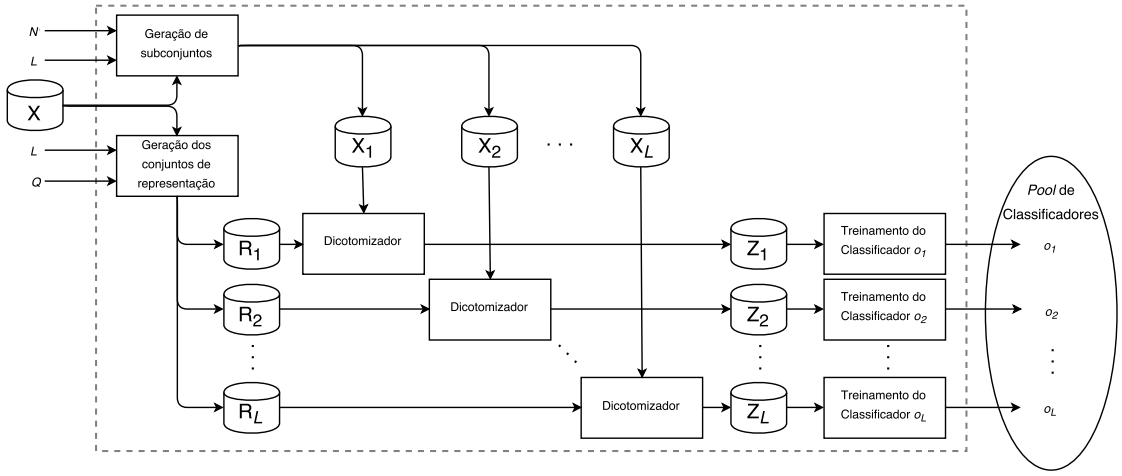


Figure 1. Fase de treinamento do método proposto. $\mathbf{X}$ é o conjunto de treinamento, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$ são os subconjuntos do conjunto de treinamento, $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$ são os conjuntos de representação, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_L$ são os conjuntos dicotomizados, L é a quantidade de classificadores no *pool*, N' é a quantidade de documentos dos subconjuntos de treinamentos, Q é a quantidade de documentos do conjunto de representação e $o_1, o_2, \ldots, o_L$ são os classificadores do *pool*.

$\mathbf{X}$, utilizado como base para o treinamento dos classificadores; e o conjunto de teste $\mathbf{Y}$, utilizado para verificar o desempenho do sistema.

No conjunto de treinamento, cada documento $x_i \in \mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ é representado por um par $\langle w_i, class(x_i)\rangle$, no qual $w_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,V}]^{\mathrm{T}}$ é o vetor de termos utilizando a representação *Bag-of-Words* e $class(x_i) \in \mathbf{C} = \{c_1, c_2, \ldots, c_C\}$ é a classe do documento. Os documentos do conjunto de teste também utiliza a representação *Bag-of-Words*, sendo composto por M documentos $y_j \in \mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$.

A Figura 1 mostra a fase de treinamento do método proposto. Nesta fase temos, como entrada, um conjunto de treinamento $\mathbf{X}$, o tamanho do *pool* de classificadores (L), o tamanho dos subconjuntos de treinamento (N') e o tamanho dos conjuntos de representação (Q); e, como saída, L classificadores treinados. Tanto N' como Q possuem o mesmo valor em todos os L classificadores. São utilizados quatro módulos distintos nessa fase: Geração de Subconjuntos, Geração dos Conjuntos de Representação, Dicotomizador e Treinamento dos Classificadores.

O módulo Geração de Subconjuntos recebe como entrada o conjunto de treinamento $\mathbf{X}$, o tamanho do *pool* de classificadores (L) e o tamanho dos subconjuntos de treinamento (N'). Este módulo trabalha com uma amostragem sem reposição utilizada para reduzir a quantidade de documentos do conjunto inicial. Todos os L subconjuntos terão o mesmo tamanho (N') e este será menor do que o conjunto de treinamento original ($N' < N$). As saídas deste módulo serão L subconjuntos de treinamento ($\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_L}$). Detalhamento na Seção 2.1.

Em paralelo ao módulo Geração de Subconjuntos, existe o módulo Geração dos Conjuntos de Representação que recebe como entrada o conjunto de treinamento $\mathbf{X}$, o tamanho do *pool* de classificadores (L) e o tamanho dos conjuntos de representação (Q).

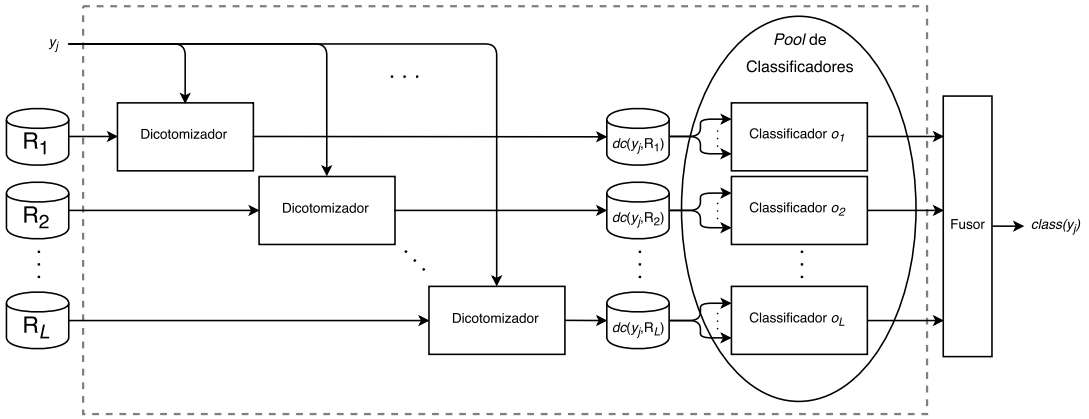


Figure 2. Fase de teste do método proposto. $y_j \in \mathbf{Y}$ é um documento do conjunto de teste, $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$, são os conjuntos de representação, $dc(y_j, \mathbf{R}_s)$ é a dicotomização de y_j para cada conjunto de representação, $class(y_j)$ é a classe resultante de $y_j \in \mathbf{Y}$ e L é a quantidade de classificadores no *pool*.

A saída deste módulo serão L subconjuntos do conjunto de treinamento ($\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_L}$) cada um com Q documentos. Deseja-se obter $Q \ll N$, pois esse conjunto, juntamente com os subconjuntos de treinamento ($\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_L}$) definem a quantidade de documentos gerados pelo Dicotomizador. Detalhamento na Seção 2.2.

Até o momento, foram gerados $2 \times L$ subconjuntos a partir do conjunto de treinamento. Na primeira metade ($\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$) cada subconjunto contém N' documentos; e na segunda metade ($\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$) cada subconjunto contém Q documentos. Cada par $\langle \mathbf{X}_s, \mathbf{R}_s \rangle$ é usado como entrada para o módulo Dicotomizador. Este módulo irá transformar o problema atual em um problema de duas classes com base nesses dois subconjuntos. Deste modo, são gerados L conjuntos dicotomizados ($\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_L$,). Cada conjunto dicotomizado terá $N' \times Q$ documentos, pois será gerado um novo documento para cada par de documentos dos subconjuntos de treinamento com os conjuntos de representação. Os documentos destes conjuntos possuem a mesma quantidade de características do conjunto original (V), mas apenas duas classes: sendo a classe positiva os documentos no subconjunto $\mathbf{Z}_s^{\oplus} \subset \mathbf{Z}_s$ e a classe negativa os documentos no subconjunto $\mathbf{Z}_s^{\ominus} \subset \mathbf{Z}_s$. Detalhamento na Seção 2.3.

Finalmente, cada um dos L conjuntos dicotomizados é apresentado ao respectivo módulo Treinamento do Classificador. O Classificador é binário, pois agora o problema possui duas classes, independentemente da quantidade de classes originais (C). Deste modo, obtemos a saída final da fase de treinamento que são os L classificadores treinados ($o_1, o_2, \ldots, o_L$).

Após a conclusão da fase de treinamento, inicia-se a fase de teste. A Figura 2 mostra a fase de teste do método proposto. Nesta fase temos, como entrada, um documento do conjunto de teste $y_j \in \mathbf{Y}$, L conjuntos de representação ($\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_L}$) e L classificadores; e, como saída, uma resposta $class(y_j)$. São utilizados três módulos distintos nessa fase: Dicotomizador, Classificador e Fusor.

O módulo Dicotomizador deve ser o mesmo utilizado na fase de treinamento. En-

tretanto, os dados de entrada do módulo são: um documento y_j do conjunto de teste $\mathbf{Y}$ e um conjunto de representação $\mathbf{R}_s$ obtido na fase de treinamento. Deste modo, cada classificador recebe Q documentos, pois o Dicotomizador transforma y_j em $1 \times Q$ documentos. Assim, a classificação não será resultado de uma resposta direta do classificador para um documento y_j, mas sim um resultado combinado das respostas obtidas para os Q documentos. Detalhamento na Seção 2.4.

As respostas dos L classificadores são combinadas pelo Fusor que dará a resposta final $class(y_j)$. Foi utilizado o voto majoritário [Li and Jain, 1998] para combinar as respostas dos L classificadores, de modo que a classe resultante será aquela que conseguir a maior quantidade de votos em comparação com as outras [Woźniak et al., 2014].

As seções a seguir detalham os módulos do método proposto.

2.1. Geração dos Subconjuntos

O módulo Geração dos Subconjuntos (Figura 1) recebe o conjunto de treinamento $\mathbf{X}$ e gera L subconjuntos $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L$, chamados de subconjuntos de treinamento. Esses subconjuntos possuem N' documentos, de modo que a quantidade de documentos desses subconjuntos seja inferior à quantidade de documentos do conjunto de treinamento $\mathbf{X}$, isto é $N' < N$. A utilização de subconjuntos diferentes, em vez da replicação do conjunto de treinamento, variam os conjuntos utilizados para treinar os classificadores, potencialmente aumentando a diversidade obtida no *pool* de classificadores.

Qualquer técnica de amostragem poderia ser utilizada nesse módulo. Entretanto, os subconjuntos de treinamento serão utilizados pelo Dicotomizador, que aumenta a quantidade de exemplos devido à transformação para duas classes (Seção 2.3). Portanto, é desejável reduzir a quantidade de documentos nesses subconjuntos treinamento. Para tal, utilizaremos um *Undersampling* aleatório, isto é, uma amostragem aleatória sem reposição de N' documentos para cada subconjunto de treinamento.

2.2. Geração dos Conjuntos de Representação

O módulo Geração dos Conjuntos de Representação (Figura 1) recebe o conjunto de treinamento $\mathbf{X}$ e gera L subconjuntos $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L$, chamados de conjuntos de representação. O conjunto de representação é um conjunto formado por documentos de referência que serão utilizados pelo Dicotomizador (Seção 2.3). Um conjunto de representação tem por objetivo englobar todos, ou a maioria, dos espaços no qual existam documentos do conjunto de treinamento.

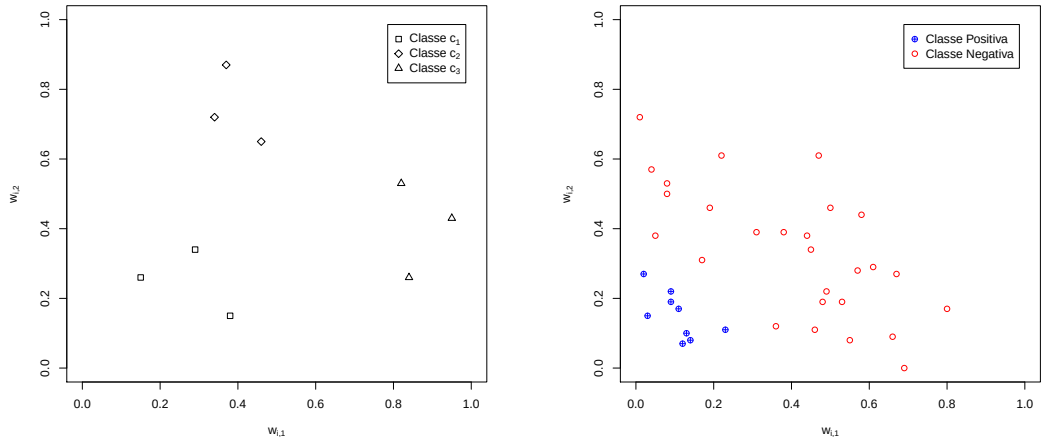
Diversas abordagens podem ser utilizadas para gerar um conjunto de representação: utilizar todos os documentos do conjunto de treinamento, selecionar alguns documentos aleatoriamente ou selecionar os documentos mais dissimilares [Pekalska and Duin, 2002]. Dado que os conjuntos de representação precisam ter a mesma quantidade de documentos por classe, foi utilizada uma seleção aleatória por classe.

A seleção aleatória por classe é uma amostragem sem reposição, isto é, nenhum documento é selecionado duas vezes; com a restrição de que a quantidade de documentos por classe é igual para todas as classes. Portanto, selecionaremos Q documentos sendo Q/C documentos para cada classe. É desejável, também, que a quantidade de documen-

tos selecionados seja menor que a quantidade de documentos originais, pois um módulo posterior (Dicotomizador) aumentará a quantidade de documentos disponíveis.

2.3. Dicotomizador

O módulo Dicotomizador recebe dois conjuntos de dados e gera um novo conjunto de dados com apenas duas classes denominado conjunto dicotomizado. Portanto, o propósito do Dicotomizador é transformar um problema multi-classe em um problema binário (duas classes), facilitando o treinamento de problemas com muitas classes e aumentando a quantidade de exemplos disponíveis para treinamento. No método proposto, esse módulo está presente na fase de treinamento (Figura 1) recebendo os subconjuntos de treinamento $\mathbf{X}_s$ e os conjuntos de representação $\mathbf{R}_s$ ($s = 1, 2, \ldots, L$) para gerar os conjuntos dicotomizados $\mathbf{Z}_s$ com $N' \times Q$ documentos, e na fase de teste (Figura 2) recebendo um documento de teste y_j e os conjuntos de representaçao $\mathbf{R}_s$ ($s = 1, 2, \ldots, L$) para gerar Q documentos. Esses conjuntos dicotomizados são baseados na Transformação de *Polychotomizer* para Dicotomizador proposto por Cha and Srihari [2000].



(a) Subconjunto $\mathbf{X}_s$ (b) Conjunto dicotomizado $\mathbf{Z}_s$

Figure 3. Gráficos mostrando a transformação de um subconjunto $\mathbf{X}_s$ com três classes para o conjunto dicotomizado $\mathbf{Z}_s$, resultando em duas classes. Neste exemplo, o conjunto de representação $\mathbf{R}_s$ é igual ao conjunto $\mathbf{X}_s$, ignorando os pares de documentos idênticos; e os documentos são representados por apenas duas características ($w_{i,1}$ e $w_{i,2}$).

Deste modo, a geração dos conjuntos dicotomizados é realizada calculando $dc(x_i, p_k)$ de cada $x_i \in \mathbf{X}_s$ para cada $p_k \in \mathbf{R}_s$. O modo mais comum dessa função ser calculada é pela diferença modular:

$$dc(x_i, p_k) = \begin{bmatrix} |w_{i,1} - wr_{k,1}| \\ |w_{i,2} - wr_{k,2}| \\ \vdots \\ |w_{i,V} - wr_{k,V}| \end{bmatrix} \quad (1)$$

no qual $w_{i,h}$ e $wr_{k,h}$ são as h-ésimas características dos documentos x_i e p_k respectivamente, e V é a quantidade de características.

Entretanto, o cálculo da diferença modular tem um comportamento de um cálculo de distância, que não funciona tão bem para problemas de características esparsas, como Categorização de Documentos. Os melhores resultados em Categorização de Documentos são atingidos com métricas de similaridade, como a similaridade do cosseno. Portanto, foi elaborado um cálculo para a função $dc(\cdot,\cdot)$ voltado para similaridade:

$$dc(x_i, p_k) = \begin{bmatrix} \frac{min(w_{i,1}, wr_{k,1})}{max(w_{i,1}, wr_{k,1})} \\ \frac{min(w_{i,2}, wr_{k,2})}{max(w_{i,2}, wr_{k,2})} \\ \vdots \\ \frac{min(w_{i,V}, wr_{k,V})}{max(w_{i,V}, wr_{k,V})} \end{bmatrix}, \tag{2}$$

em caso de valores zero na função $max(\cdot,\cdot)$ é assumido o valor 0 naquela característica. Assim, as características nos conjuntos dicotomizados serão no intervalo $[0, 1]$, sendo mais próximas do valor 1 quando as características forem mais similares.

Deste modo, ao calcular $dc(\mathbf{X}_s, \mathbf{R}_s)$ são gerados $\mathbf{Z}_s$ com $N' \times Q$ documentos. Esses documentos são divididos em dois conjuntos: $\mathbf{Z}_s^{\oplus}$ são os documentos da classe positiva, quando $class(x_i) = class(p_k)$, e $\mathbf{Z}_s^{\ominus}$, são os documentos da classe negativa, quando $class(x_i) \neq class(p_k)$.

A Figura 3 exemplifica a transformação de um conjunto com três classes ($\mathbf{X}_s$) para um conjunto com duas classes ($\mathbf{Z}_s$) utilizando a Equação (1), isto é, dissimilaridade. No conjunto dicotomizado $\mathbf{Z}_s$ os documentos da classe positiva, obtidos por pares de documentos da mesma classe, ficam próximos à origem; enquanto que documentos da classe negativa, obtidos por pares de documentos de classes diferentes, ficam distantes da origem.

Na fase de teste (Figura 2) o procedimento é similar. Temos um documento do conjunto de teste $y_j \in \mathbf{Y}$ e o conjunto de representação $\mathbf{R}$ para dicotomização, obtendo Q documentos:

$$dc(y_j, \mathbf{R}) = \left\{ \begin{bmatrix} |w'_{j,1} - wr_{1,1}| \\ |w'_{j,2} - wr_{1,2}| \\ \vdots \\ |w'_{j,V} - wr_{1,V}| \end{bmatrix}, \begin{bmatrix} |w'_{j,1} - wr_{2,1}| \\ |w'_{j,2} - wr_{2,2}| \\ \vdots \\ |w'_{j,V} - wr_{2,V}| \end{bmatrix}, \cdots, \begin{bmatrix} |w'_{j,1} - wr_{Q,1}| \\ |w'_{j,2} - wr_{Q,2}| \\ \vdots \\ |w'_{j,V} - wr_{Q,V}| \end{bmatrix} \right\} \tag{3}$$

no qual $w'_{j,h}$ e $wr_{k,h}$ são as h-ésimas características dos documentos y_j e $p_k \in \mathbf{R}$ respectivamente, e V é a quantidade de características. Considerando o cálculo da função $dc(\cdot,\cdot)$ de acordo com a Equação (1).

2.4. Classificador

Cada classificador o_s é treinado utilizando os documentos dos conjuntos dicotomizados $\mathbf{Z}_s$. Cada conjunto possui $N' \times Q$ documentos divididos em duas classes: os documentos

pertencentes ao conjunto $\mathbf{Z}_s^{\oplus}$ e os documentos pertencentes ao conjunto $\mathbf{Z}_s^{\ominus}$. O treinamento é realizado com base nessas duas classes, obtendo um classificador binário treinado ao final da fase de treinamento. O diferencial encontra-se na fase de teste. Prosseguindo com o exemplo apresentado na Figura 3, o classificador é treinado obtendo uma superfície de separação das duas classes (Figura 4).

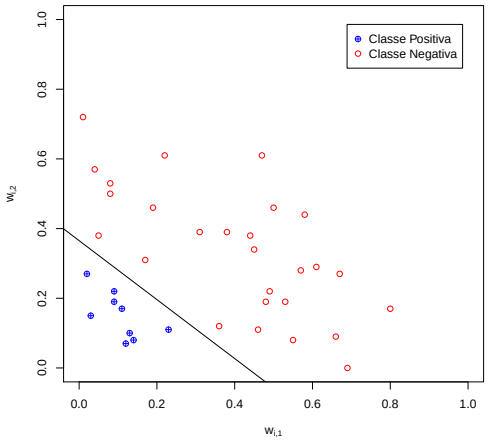


Figure 4. Classificador o_s treinado com os dados $\mathbf{Z}_s$.

Na fase de teste, cada documento do conjunto de teste $y_j \in \mathbf{Y}$ é transformado em Q documentos. Essa transformação é feita com base nos conjuntos de representação $\mathbf{R}_s$. Portanto, a classificação de y_j dependerá da classificação de Q representações diferentes dele próprio.

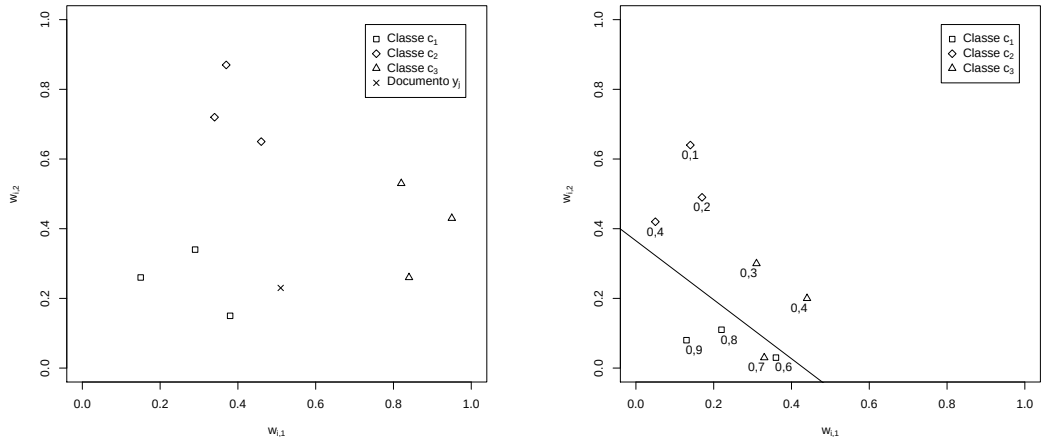
Dado que em cada $\mathbf{R}_s$ existem Q documentos distribuídos igualmente entre as C classes originais, as Q representações de y_j estarão ligadas diretamente com as classes dos protótipos $p_k \in \mathbf{R}_s$. Cada representação de y_j passará por um classificador o_s, sendo a classe resultante obtida pela seguinte equação:

$$class(y_j) = \underset{l=\{1,2,...,C\}}{\operatorname{argmax}} \sum_{\substack{p_k \in \mathbf{R}_s \\ class(p_k)=c_l}} p(dc(y_j, p_k)|\mathbf{Z}_s^{\oplus}), \tag{4}$$

isto é, calcula-se a probabilidade cada documento transformado $dc(y_j, p_k) \forall p_k \in \mathbf{R}_s$ ser classificado como pertencente a classe positiva ($\mathbf{Z}_s^{\oplus}$). Essas probabilidades são somadas por classe, de modo que y_j será classificado como pertencente à classe que as representações de y_j tiveram maior somatório das probabilidades de serem classificados positivamente.

A Figura 5 mostra um exemplo hipotético de classificação do documento y_j, que é classificado como pertence à classe c_1, pois $0,9 + 0,8 + 0,6 = 2,3$ (classe c_1) é maior que $0,4 + 0,2 + 0,1 = 0,7$ (classe c_2) e $0,6 + 0,4 + 0,3 = 1,3$ (classe c_3).

Uma alternativa mais simples seria uma contagem de votos dos documentos transformados classificados positivamente. Entretanto, o potencial de empates para Conjuntos de Representação com poucos documentos por classe torna essa alternativa inviável.



(a) y_j e $\mathbf{X}_s$ (b) $dc(y_j, \mathbf{R}_s)$ com relação ao classificador o_s

Figure 5. Documento y_j é apresentado na figura à esquerda, passa pelo Dicotomizador e gera $Q = 9$ documentos (figura à direita). Os números são as probabilidades $p(dc(y_j, p_k)|\mathbf{Z}_s^{\oplus})$. Assim, temos $class(y_j) = c_1$.

3. Experimentos

Todos os experimentos foram realizados utilizando validação cruzada estratificada com 10 *folds*. A quantidade de documentos dos subconjuntos de treinamento (N') foi igual à quantidade de documentos dos conjuntos de representação (Q). Essa quantidade foi definida como 10% da quantidade de documentos do conjunto de treinamento original ($0.1 \times N$). Entretanto, devido à restrição imposta sobre o conjunto de representação: todas as classes devem ter a mesma quantidade de documentos, é possível que $Q \leq 0.1 \times N$, pois existem classes com poucos documentos, inviabilizando a exatidão dos 10%. A avaliação dos métodos foram realizadas com as métricas macro-F1 e micro-F1 [Sebastiani, 2002].

Dezesseis (16) bases de dados foram utilizadas nos experimentos. A Tabela 1 mostra as características de cada base de dados: quantidade de classes, quantidade de documentos total e quantidade de características. Todas as bases de dados foram obtidas no seguinte endereço eletrônico: `http://sites.labic.icmc.usp.br/text_collections/`.

Dado que o método proposto possui dois cálculos para a dicotomização, eles serão identificados daqui por diante como: DicoDiff quando utiliza a Equação (1) e DicoSim quando utiliza a Equação (2).

Nos experimentos, o método proposto (DicoDiff e DicoSim) foi comparado com *SVM Single*, um classificador individual de SVM; *Bootstrap AGGregatING* [Breiman, 1996], também conhecido como *Bagging*, uma combinação de SVMs no qual cada classificador é treinado com um subconjunto de mesmo tamanho do conjunto de treinamento original obtidos por uma amostragem aleatória com reposição; *Random Subspace* [Ho, 1998], uma combinação de SVMs no qual cada classificador é treinado em um espaço de característica diferente, isto é, treinamento de cada classificador é realizado sobre os

Table 1. Descritores das bases de dados: número de classes, número de documentos e número de características

Bases de Dados	Classes	Documentos	Características
CSTR	4	299	1726
Oh0	10	1003	3183
Oh5	10	918	3013
Oh10	10	1050	3239
Oh15	10	913	3103
Re0	13	1504	2887
Re1	25	1657	3759
Syskill	4	334	4340
Tr11	9	414	6430
Tr12	8	313	5805
Tr21	6	335	7903
Tr23	6	204	5833
Tr31	7	927	10129
Tr41	10	878	7455
Tr45	10	690	8262
WAP	20	1560	8461

Table 2. Melhores resultados, macro-F1 e micro-F1, para cada base de dados e seus respectivos tamanhos do *ensemble* (L) dos métodos Bagging, Random Subspace, Dico Diff e Dico Sim. Em negrito os melhores resultados em cada base de dados.

	Single SVM		*Bagging*			*Random Subspace*			*DicoDiff*			*DicoSim*		
Bases	macro	micro	L	macro	micro	L	macro	micro	L	macro	micro	L	macro	micro
CSTR	80.56 (10.69)	79.21 (8.63)	90	79.09 (10.12)	77.90 (8.59)	50	**84.39 (6.95)**	**80.92 (8.04)**	30	68.43 (7.74)	71.37 (7.34)	100	82.15 (9.06)	79.95 (4.97)
Oh0	85.64 (2.77)	86.55 (2.55)	50	86.38 (3.79)	87.41 (3.42)	70	87.37 (3.56)	88.42 (3.17)	100	89.14 (2.20)	87.57 (2.13)	60	**91.53 (1.84)**	**91.52 (1.60)**
Oh5	86.55 (3.41)	86.73 (3.45)	60	86.78 (3.41)	86.83 (3.40)	90	87.09 (3.66)	86.90 (3.55)	10	87.98 (3.40)	87.34 (3.40)	70	**89.66 (3.67)**	**90.05 (3.27)**
Oh10	74.91 (3.78)	77.59 (3.77)	30	77.69 (3.46)	79.97 (3.71)	100	80.26 (4.70)	81.82 (3.30)	50	82.65 (4.75)	83.39 (4.07)	100	**83.17 (4.58)**	**84.48 (4.28)**
Oh15	80.56 (3.95)	81.00 (4.05)	60	80.76 (3.12)	81.01 (3.45)	100	82.67 (3.74)	82.67 (3.73)	100	85.04 (3.15)	84.05 (3.25)	90	**85.73 (3.43)**	**85.01 (3.64)**
Re0	78.03 (7.05)	83.65 (3.24)	80	**78.62 (8.27)**	**84.16 (2.83)**	80	70.74 (10.71)	80.09 (3.65)	20	70.91 (7.83)	74.61 (2.96)	80	65.49 (7.00)	74.18 (2.15)
Re1	74.43 (6.57)	83.99 (2.45)	10	73.64 (5.75)	**84.54 (2.77)**	80	67.90 (5.18)	81.38 (2.48)	90	**75.31 (4.08)**	84.19 (1.47)	80	73.58 (5.06)	83.74 (1.64)
Syskill	92.73 (4.58)	92.77 (5.11)	30	93.35 (5.23)	93.06 (5.78)	100	92.73 (5.11)	92.98 (5.08)	50	**94.78 (3.96)**	**94.87 (3.81)**	40	93.49 (3.95)	93.05 (4.15)
Tr11	73.91 (9.34)	87.20 (5.70)	10	74.23 (7.63)	88.12 (4.51)	10	77.50 (5.70)	**90.32 (4.40)**	100	**78.53 (7.74)**	88.86 (4.34)	70	67.84 (7.74)	85.68 (4.35)
Tr12	83.21 (7.82)	87.23 (5.84)	30	84.30 (6.25)	88.16 (5.02)	100	84.12 (5.16)	86.96 (3.31)	100	**87.65 (6.53)**	**88.61 (6.26)**	100	83.31 (6.55)	84.39 (4.38)
Tr21	**65.53 (16.88)**	89.92 (4.86)	50	63.88 (16.47)	89.64 (4.66)	30	65.51 (17.63)	**90.84 (5.63)**	80	41.20 (7.15)	83.17 (5.70)	100	28.39 (8.07)	74.56 (6.19)
Tr23	**74.00 (12.43)**	84.19 (4.60)	30	68.54 (14.27)	82.28 (6.51)	90	68.95 (15.83)	84.04 (7.18)	20	69.96 (20.10)	83.15 (6.86)	80	53.73 (9.36)	**84.23 (5.88)**
Tr31	83.36 (1.73)	97.74 (1.49)	60	83.38 (1.79)	97.74 (1.56)	80	**83.46 (1.78)**	**97.95 (1.39)**	100	67.23 (10.39)	89.94 (4.87)	100	58.81 (10.19)	85.18 (8.75)
Tr41	89.04 (5.47)	96.02 (1.98)	40	87.21 (6.42)	95.21 (2.31)	50	89.14 (5.90)	96.32 (2.31)	40	86.97 (5.93)	95.80 (2.01)	40	**91.41 (8.26)**	**97.31 (2.21)**
Tr45	85.97 (6.45)	91.59 (2.67)	50	85.34 (5.85)	90.86 (2.26)	60	88.03 (4.94)	93.26 (2.82)	80	**91.55 (6.27)**	94.58 (2.14)	90	91.45 (4.57)	**96.26 (1.56)**
WAP	**74.08 (5.35)**	**85.69 (2.57)**	50	68.35 (4.27)	84.24 (3.43)	40	70.46 (6.39)	85.67 (2.72)	100	64.86 (5.13)	81.15 (3.12)	90	62.72 (5.64)	80.69 (2.42)

mesmos documentos mas cada um deles utilizando apenas uma parte das características, sendo estas selecionadas aleatoriamente. A utilização do SVM é devido ao seu desempenho exemplar em problemas de Categorização de Documentos, do *Bagging* é por ser bastante utilizado como *baseline* para outros métodos de combinação de classificadores [Qian et al., 2014, Islam et al., 2003, Kim and Kang, 2012], e do Random Subspace é por conta da sua capacidade de lidar com problemas de alta dimensionalidade. Todos os SVMs utilizados, sejam combinações ou não, utilizam *kernel* linear. A quantidade de classificadores para cada combinação foi executada com 10 valores diferentes ($L = \{10, 20, \ldots, 100\}$). No *Random Subspace* cada classificador foi treinado com 20% das características originais, valor previamente estabelecido em outros trabalhos [Campos et al., 2012, Gangeh et al., 2010].

A Tabela 2 mostra uma comparação dos melhores resultados obtidos por cada método estudado. Podemos observar que o DicoSim obtém o melhor resultado mais vezes do que qualquer outro método. Algumas das bases de dado sofrem por conta da restrição dos conjuntos de representação precisar ter a mesma quantidade de documentos por classe (Tr21, Tr23 e Tr31), dado o alto grau de desbalanceamento. Uma alternativa para tratar desse problema seria a aplicação de técnicas de *Oversampling*.

4. Conclusão

Neste artigo foi apresentado um sistema de Categorização de Documentos que utiliza Combinação de Classificadores com Dicotomização. O método proposto é modular, possibilitando a substituição dos métodos utilizados em cada módulo, seja com relação à geração dos subconjuntos de treinamento, geração dos conjuntos de representação, classificador ou fusor. Além da variação de parâmetros, como N', L e Q. Devido à quantidade de variações possíveis e viabilidade dos experimentos, alguns parâmetros foram fixados.

Nos experimentos, o método proposto é comparado com outros métodos da literatura, como *Bagging*, *Random Subspace* e até mesmo SVM individual. No método proposto destacou-se a opção com o cálculo utilizando similaridade em vez da diferença modular, como era o esperado devido à natureza esparsa dos problemas de Categorização de Documentos.

Entre os trabalhos futuros incluem, avaliação mais aprofundada dos parâmetros do método proposto, modificação na geração dos subconjuntos para tratar problemas de classes desbalanceadas, avaliação do método proposto em mais bases de dados e substituição do módulo Classificador por alguma heurística simples.

References

Hongshik Ahn, Hojin Moon, Melissa J Fazzari, Noha Lim, James J Chen, and Ralph L Kodell. Classification by ensembles from random partitions of high-dimensional data. *Computational Statistics & Data Analysis*, 51(12):6166–6179, 2007.

Yaxin Bi, David Bell, Hui Wang, Gongde Guo, and Kieran Greer. Combining multiple classifiers using dempster's rule of combination for text categorization. In *Modeling Decisions for Artificial Intelligence*, pages 127–138. Springer, 2004.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Yoisel Campos, Carlos Morell, and Francesc J Ferri. A local complexity based combination method for decision forests trained with high-dimensional data. In *International Conference on Intelligent Systems Design and Applications*, pages 194–199. IEEE, 2012.

Sung-Hyuk Cha and Sargur N Srihari. Writer identification: statistical analysis and dichotomizer. In *Advances in Pattern Recognition*, pages 123–132. Springer, 2000.

Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, 2000a.

Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000b.

Mehrdad J Gangeh, Mohamed S Kamel, and Robert PW Duin. Random subspace method in text categorization. In *Proceedings of the International Conference on Pattern Recognition*, pages 2049–2052, 2010.

Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Md Monirul Islam, Xin Yao, and Kazuyuki Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4): 820–834, 2003.

Myoung-Jong Kim and Dae-Ki Kang. Classifiers selection in ensembles using genetic algorithms for bankruptcy prediction. *Expert Systems with Applications*, 39(10):9308–9314, 2012.

Yong H Li and Anil K. Jain. Classification of text documents. *The Computer Journal*, 41 (8):537–546, 1998.

Guansong Pang and Shengyi Jiang. A generalized cluster centroid based classifier for text categorization. *Information Processing & Management*, 49(2):576–586, 2013.

Santhosh Pathical and Gursel Serpen. Comparison of subsampling techniques for random subspace ensembles. In *International Conference on Machine Learning and Cybernetics*, volume 1, pages 380–385. IEEE, 2010.

Elzbieta Pekalska and Robert PW Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23(8):943–956, 2002.

Yongjun Piao, Hyun Woo Park, Cheng Hao Jin, and Keun Ho Ryu. Ensemble method for classification of high-dimensional data. In *International Conference on Big Data and Smart Computing*, pages 245–249. IEEE, 2014.

Yun Qian, Yanchun Liang, Mu Li, Guoxiang Feng, and Xiaohu Shi. A resampling ensemble algorithm for classification of imbalance problems. *Neurocomputing*, 143:57–67, 2014.

Rafael Geraldeli Rossi, Alneu de Andrade Lopes, Thiago de Paulo Faleiros, and Solange Oliveira Rezende. Inductive model generation for text classification using a bipartite heterogeneous network. *Journal of Computer Science and Technology*, 29 (3):361–375, 2014.

F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002. ISSN 0360-0300.

Catherine A Shipp and Ludmila I Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information fusion*, 3(2):135–148, 2002.

David H Wolpert. The supervised learning no-free-lunch theorems. In *Soft Computing and Industry*, pages 25–42. Springer, 2002.

Michał Woźniak, Manuel Graña, and Emilio Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17, 2014.

Yiming Yang, Tom Ault, and Thomas Pierce. Combining multiple learning strategies for effective cross validation. In *Proceedings of International Conference on Machine Learning*, pages 1167–1174. Citeseer, 2000.

Hualong Yu and Jun Ni. An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data. *IEEE Transactions on Computational Biology and Bioinformatics*, 11(4):657–666, 2014.

Expert Systems with Applications 42 (2015) 1941–1949

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Data-driven global-ranking local feature selection methods for text categorization

Roberto H.W. Pinheiro, George D.C. Cavalcanti *, Tsang Ing Ren

Universidade Federal de Pernambuco (UFPE), Centro de Informática (CIn), Av. Jornalista Anibal Fernandes s/n, Cidade Universitária, 50740-560 Recife, PE, Brazil

ARTICLE INFO

Article history:
Available online 17 October 2014

Keywords:
Text classification
High dimensionality
Feature selection
Filtering method
Variable Ranking
ALOFT

ABSTRACT

Bag-of-words is the most used representation method in text categorization. It represents each document as a feature vector where each vector position represents a word. Since all words in the database are considered features, the feature vector can reach tens of thousands of features. Therefore, text categorization relies on feature selection to eliminate meaningless data and to reduce the execution time. In this paper, we propose two filtering methods for feature selection in text categorization, namely: Maximum *f* Features per Document (MFD), and Maximum *f* Features per Document – Reduced (MFDR). Both algorithms determine the number of selected features *f* in a data-driven way using a global-ranking Feature Evaluation Function (FEF). The MFD method analyzes all documents to ensure that each document in the training set is represented in the final feature vector. Whereas MFDR analyzes only the documents with high FEF valued features to select less features therefore avoiding unnecessary ones. The experimental study evaluated the effectiveness of the proposed methods on four text categorization databases (20 Newsgroup, Reuters, WebKB and TDT2) and three FEFs using the Naïve Bayes classifier. The proposed methods present better or equivalent results when compared with the ALOFT method, in all cases, and Variable Ranking, in more than 93% of the cases.

1. Introduction

The majority of text categorization studies – also known as topic spotting or text classification – use bag-of-word (Manning, Raghavan, & Schtze, 2008) to represent a document as a vector. In this representation, each word or term that appears in a document is considered a feature. Consequently, the representation can reach tens of thousands of features (Lewis, 1992). The high dimensionality of the feature vector increases storage requirement, execution time and error rate. The error rate increases due to the presence of noisy and irrelevant features. Therefore, text categorization relies on feature selection to require less storage space, streamline the runtime and achieve better accuracy.

The objective of feature selection is to reduce the size of the feature vector without sacrificing the performance of the categorization (Sebastiani, 2002). Feature selection methods are divided into two main approaches: filtering (Yu & Liu, 2003), and wrapper (Bermejo, Gámez, & Puerta, 2014; John, Kohavi, & Pfleger, 1994). Both approaches select a subset of features using an evaluation function (Yang, Liu, Liu, Zhu, & Zhang, 2011). In the wrapper approach, a classifier evaluates many different subsets and selects the one with the highest accuracy rate. On the other hand, in filtering approaches, the construction of the final subset is not classifier dependent.

Filtering techniques are the most popular and fast approach (Fragoudis, Meretakis, & Likothanassis, 2005). Despite good reports on the results of wrapper techniques (Meiri & Zahavi, 2006), they are very computationally expensive and even prohibitive if used with more complex classifiers. Consequently, due to the high dimensionality of text categorization problems, this paper focuses on filtering techniques.

Many filtering methods (Baccianella, Esuli, & Sebastiani, 2013; Chen, Huang, Tian, & Qu, 2009; Debole & Sebastiani, 2003) require an input parameter that defines how many features must be selected. An example of a method that does not require such parameter is the At Least One FeaTure (ALOFT) method (Pinheiro, Cavalcanti, Correa, & Ren, 2012). However, ALOFT may not select enough features to discriminate well the categories, decreasing the accuracy of the classifier. This problem occurs in 20 Newsgroup with Bi-Normal Separation (Pinheiro et al., 2012).

Here, we propose two methods to address the drawbacks of ALOFT and to improve the overall accuracy rate. The first proposed method is a generalization of ALOFT called MFD (Maximum *f*

* Corresponding author. Tel.: +55 81 2126 8430x4346; fax: +55 81 2126 8438.
E-mail addresses: rhwp@cin.ufpe.br (R.H.W. Pinheiro), gdcc@cin.ufpe.br (G.D.C. Cavalcanti), tir@cin.ufpe.br (T.I. Ren).
URL: http://www.cin.ufpe.br/~viisar (G.D.C. Cavalcanti).

http://dx.doi.org/10.1016/j.eswa.2014.10.011
0957-4174/

Features per Document) that selects f valued features for every document in the training set. Using more than one feature per document, we expect to increase the capability of the method in dealing with different databases. The second proposed method is MFDR (Maximum f Features per Document – Reduced) that analyzes a subset of the whole set of documents in order construct the subset of features. Since only documents with high FEF valued features are used, we expect to eliminate irrelevant documents in terms of discriminative power.

This paper is organized as follows. Section 2 reviews the classical approach of feature selection by filtering methods. It includes a description of Variable Ranking and ALOFT methods. Section 3 presents the proposed methods. Section 4 shows an evaluation of the proposed methods using a toy dataset. Section 5 describes the methodology of the experiments, databases, classifier, Feature Evaluation Functions, metrics and methods to evaluate the text categorization effectiveness. Section 6 reports the experimental results and analysis. Finally, Section 7 concludes the paper.

2. Feature selection methods

Feature selection can be defined as a search problem where each state in the search space specifies a subset of the possible features. Exhaustive evaluation of the feature subsets is usually intractable, especially if the feature vector has high dimensionality such as in text categorization problems.

To avoid exhaustive search, several feature selection methods used in text categorization relies on Feature Evaluation Function (FEF) (Rogati & Yang, 2002) to calculate a degree of significance of each feature. FEF calculates how important a feature is given a training set. So, instead of evaluating many different subsets, it is common to select the features that have higher degree of significance to compose the final subset of features.

The next subsections show two algorithms that use FEF in the feature selection process.

2.1. Variable Ranking

Variable Ranking (VR) (Guyon & Elisseeff, 2003) is a classic feature selection method used for text categorization tasks (Forman, 2003; Yang & Pedersen, 1997). It is a filtering method that usually serves as bias to evaluate proposed feature selection algorithms (Pinheiro et al., 2012; Xue & Zhou, 2009). VR has two parameters: an FEF and a positive integer n. The algorithm selects the subset of features as follows: (i) a degree of significance is calculated for each feature using the given FEF; (ii) the features are globally ranked based on the FEF outputs; and, (iii) the top n features are selected.

Several FEFs have been proposed, such as: Chi-Squared (Yang & Pedersen, 1997), Information Gain (Quinlan, 1986), Odds Ratio (Mladenic & Grobelnik, 1998) and Gini Index (Breiman, Friedman, Stone, & Olshen, 1984). These are classical FEFs. However, there are FEFs that were proposed specifically for text categorization, such as: Bi-Normal Separation (Forman, 2003), Orthogonal Centroid Feature Selection (Yan et al., 2005), Improved Gini Index (Shang et al., 2007), Class Discriminating Measure (Chen et al., 2009), Ambiguity Measure (Mengle & Goharian, 2009), Comprehensively Measure Feature Selection (Yang, Liu, Zhu, Liu, & Zhang, 2012) and Distinguishing Feature Selector (Uysal & Gunal, 2012).

Despite the amount of proposed FEFs, VR has some drawbacks: the parameter n is generally chosen in a try-and-error methodology (Yang & Pedersen, 1997); the optimization to define the parameter n has high computational cost since several tests are performed with different values of n in order to find the best one; if the value n is too small, some documents will have no representation at all, making its classification random. Therefore, defining a good value for n is not a trivial task, especially because the value is different for each combination of database and FEF.

2.2. At Least One FeaTure

The At Least One FeaTure (ALOFT) algorithm is a filtering method that ensures the contribution of each document to the final feature subset (Pinheiro et al., 2012). Thus, all documents in the training set possess at least one non-zero feature.

ALOFT selects the subset of feature as follows: (i) each feature is scored using a given FEF; (ii) the features are globally ranked based on the FEF outputs; (iii) every document in the training set is visited; (iv) in each document is chosen the feature with the highest FEF value and weight different of zero; and, (v) if the chosen feature was already selected by a previously visited document, no new feature is selected in that document.

The main advantages of ALOFT over VR are: the number of features is automatically determined after sweeping each document only once; it requires only one parameter: an FEF; given an FEF, it finds the smallest possible number of features to cover all documents in the training set.

3. Proposed methods

ALOFT has an intrinsic restriction of considering only one feature per document. However, if the FEF privileges the same feature in several documents, probably those documents will not have enough features to discriminate the categories.

A possible solution for these cases is to select more features per document. Despite of this change which increases the number of selected features, the quality of the selection can be improved selecting more features. That is the main idea of both proposed methods (MFD and MFDR) which are presented in this section. The next subsections describes the algorithms of the proposed methods.

3.1. Maximum f Features per Document (MFD)

Algorithm 1 shows the pseudo-code of the MFD method. The algorithm requires a specific value for f that must be a positive integer greater than zero.

The algorithm is described as follows:

- Line 1: A training set $\mathcal{D}_{tr}$ is loaded. The set is composed of $d \in \mathbb{N}^V$ documents, V is the size of the vocabulary (number of features).
- Lines 2–4: For each feature w_h, the FEFs values are calculated and stored in S_h. Thus, S_h represents the importance of the hth feature. Note that $S \in \mathbb{R}^V$.
- Lines 5–26: The new set of features FS is computed. The hth feature is inserted in FS (S_h is the highest value among all features). However, if this feature is already in FS, it is ignored and the algorithm increments nf to search for another feature or continues to the next document (if $nf > f$). At Line 20, $S_{bestfeature}$ assumes a negative value. This avoids future selection of this feature in the document d_i under analysis. After, the S vector has its values restored (Lines 23–25) because the next document must deal with the original values. At the end of this phase, FS should be a vector with m values, and these values represent the indexes of the selected features.
- Lines 27–32: The new training set $\mathcal{D}_{nv}$ is constructed. It is composed of $d' \in \mathbb{N}^m$ documents, m represents the number of selected features. The test set is generated using the same procedure.

Algorithm 1. MFD

Require: $f \geqslant 1 \in \mathbb{N}$
1: Load training set $\mathcal{D}_{tr}$
2: **for** $h = 1$ to V **do** {Calculate FEF for each term}
3: $S_h = \text{FEF}(w_h)$
4: **end for**
5: $m = 0$
6: FS is an empty vector
7: **for all** $d_i \in \mathcal{D}_{tr}$ **do** {Select f valued features with the highest FEF per document}
8: **for** $nf = 1$ to f **do**
9: $bestscore = 0.0$
10: $bestfeature = null$
11: **for** $h = 1$ to V **do**
12: **if** $w_{h,i} > 0$ and $S_h > bestscore$ **then**
13: $bestscore = S_h$
14: $bestfeature = h$
15: **end if**
16: **end for**
17: **if** $bestfeature \notin FS$ and $bestfeature \neq null$ **then**
18: $m = m + 1$
19: $FS_m = bestfeature$
20: $S_{bestfeature} = -1 \times S_{bestfeature}$ {Negativates to avoid select the feature again}
21: **end if**
22: **end for**
23: **for** $h = 1$ to V **do**
24: $S_h = \mid S_h \mid$ {Restores the original FEF value}
25: **end for**
26: **end for**
27: $\mathcal{D}_{nv}$ an empty database
28: **for all** $d \in \mathcal{D}_{tr}$ **do**
29: Create d' in $\mathcal{D}_{nv}$
30: **for** $h = 1$ to m **do**
31: $d'_h = d_{FS_h}$
32: **end for**
33: **end for**

3.2. Maximum f Features per Document – Reduced (MFDR)

In the MFDR method, the first step is to calculate the relevance of each document given by

$$\text{DR}(d_i) = \sum_{h=1}^{V} \big(\text{FEF}(w_h) \times \text{bin}(w_{h,i})\big), \tag{1}$$

where $d_i \in \mathbb{N}^V$ is a document, V is the size of the vocabulary (number of features), w_h is the hth feature and $w_{h,i}$ is the hth feature of the ith document. bin$(\cdot)$ returns a binary value indicating if a specific feature is valued or not.

$$\text{bin}(w_{h,i}) = \begin{cases} 1 & w_{h,i} \neq 0, \\ 0 & w_{h,i} = 0. \end{cases} \tag{2}$$

So, DR(d_i) is the sum of the FEF values of valued features in d_i. The DR value indicates the importance or the representative power of a document.

Algorithm 2 presents the pseudo-code of MFDR. When compared with its predecessor, MFD, it has three major modifications:

(i) Lines 2–4: This loop calculates the values of the DR vector. The vector stores the importance of each document d_i;
(ii) Line 5: The threshold mDR is calculated;
(iii) Line 12: Documents having DR smaller than the threshold are discarded.

After these three modifications, the algorithm selects equal or less features than MFD. The restriction applied by MFDR decreases the probability of selecting irrelevant or noisy features because a document is considered for feature selection only if its DR value is greater than a threshold (mDR). So, we expect that documents with DR value below average to be equivalent to documents with poor discriminative features.

Algorithm 1. MFDR

Require: $f \geqslant 1 \in \mathbb{N}$
1: Load training set $\mathcal{D}_{tr}$
2: **for all** $d_i \in \mathcal{D}_{tr}$ **do** {Calculate DR for each document}
3: $DR_i = \text{DR}(d_i)$
4: **end for**
5: $mDR = \text{mean}(DR)$;
6: **for** $h = 1$ to V **do** {Calculate FEF for each term}
7: $S_h = \text{FEF}(w_h)$
8: **end for**
9: $m = 0$
10: FS an empty vector
11: **for all** $d_i \in \mathcal{D}_{tr}$ **do** {Select f valued features with the highest FEF per document}
12: **if** $DR_i > mDR$ **do** {Ignore all documents that has DR smaller than the average}
13: **for** $nf = 1$ to f **do**
14: $bestscore = 0.0$
15: $bestfeature = null$
16: **for** $h = 1$ to V **do**
17: **if** $w_{h,i} > 0$ and $S_h > bestscore$ **then**
18: $bestscore = S_h$
19: $bestfeature = h$
20: **end if**
21: **end for**
22: **if** $bestfeature \notin FS$ and $bestfeature \neq null$ **then**
23: $m = m + 1$
24: $FS_m = bestfeature$
25: $S_{bestfeature} = -1 \times S_{bestfeature}$ {Negativates to avoid select the feature again}
26: **end if**
27: **end for**
28: **for** $h = 1$ to V **do**
29: $S_h = \mid S_h \mid$ {Restores the original FEF value}
30: **end for**
31: **end if**
32: **end for**
33: $\mathcal{D}_{nv}$ an empty database
34: **for all** $d \in \mathcal{D}_{tr}$ **do**
35: Create d' in $\mathcal{D}_{nv}$
36: **for** $h = 1$ to m **do**
37: $d'_h = d_{FS_h}$
38: **end for**
39: **end for**

4. Toy examples

In order to illustrate how the proposed methods work, a hypothetical training set (Table 1) composed of 13 documents represented by 9 boolean features (presence or absence of the word) was constructed. We define S to represent the importance of each feature as in Algorithms 1 and 2. However, for simplicity, S should assume only integer values. The column DR is only used for the reduced version (MFDR).

Table 1
A toy example. The first column (D) represents an index to identify each document, the second-to-last column (C) represents the category of the document and the columns between (w_i) are the features. The last column (DR) represents the importance of each document. Each line represents one document, except the last one, which represents the S vector.

D	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	C	DR
d_1	1	1	1	0	0	0	1	1	0	A	36
d_2	1	0	0	1	1	0	0	1	1	A	53
d_3	1	0	0	1	0	0	0	1	1	A	43
d_4	0	1	0	0	0	0	0	0	0	A	14
d_5	0	1	0	0	0	0	1	1	0	A	21
d_6	0	0	0	1	0	1	0	0	0	B	23
d_7	0	0	0	1	0	1	0	0	1	B	35
d_8	1	0	1	1	1	0	0	0	0	B	40
d_9	0	0	0	1	0	1	1	0	0	B	25
d_{10}	0	0	0	1	0	1	1	0	1	B	37
d_{11}	1	0	1	1	1	0	1	0	0	B	42
d_{12}	0	0	0	1	1	0	0	0	1	B	37
d_{13}	1	0	0	1	0	1	0	0	0	B	34
S	11	14	4	15	10	8	2	5	12	-	-

4.1. A toy example for MFD

The first step of MFD is to calculate the values of the S vector; any FEF can be used. For this example, the values in the table are all hypothetical.

The second step is to select the best features. For each document, the f best valued features are selected based on the S vector; consider $f = 2$. The first document (d_1) presents five valued features: w_1, w_2, w_3, w_7, and w_8. Since $f = 2$, only two features are selected. The first selected feature is w_2 because $S_2 > S_1 > S_8 > S_3 > S_7$, and the second selected feature is w_1 because $S_1 > S_8 > S_3 > S_7$. So, the index of these words are inserted in the vector $FS = \{2, 1\}$.

For the second document (d_2), there are five valued features: w_1, w_4, w_5, w_8, and w_9. The first and the second selected feature are w_4 and w_9 because they presents the highest values in the S vector among all the valued features; now, $FS = \{2, 1, 4, 9\}$. The next document is d_3 and the selected words are w_4 and w_9. Since both words are already in FS, no action is performed. For d_4, there is only one valued feature (w_2), so only w_2 is selected. However, $w_2 \in FS$, then the algorithm continues to the next document. For d_5, the first selection is w_2, but $w_2 \in FS$. Although, the second best feature is w_8, since $w_8 \notin FS$, now $FS = \{2, 1, 4, 9, 8\}$. For d_6, there is only two valued features: w_4 and w_6, since $w_4 \in FS$, only w_6 is inserted to FS, then $FS = \{2, 1, 4, 9, 8, 6\}$. From d_7 to d_{13}, no feature is added to FS.

The third step is to create the new training set based on the index present in FS, as shown in Table 2. It is important to note that even when $f = 2$, there are some documents that contribute with only one feature to the FS vector (such as d_5 and d_6), or no new feature (such as d_3 and d_7).

Table 2
The selected features for the toy example (Table 1) using ALOFT, MFD and MFDR.

D	ALOFT		MFD						MFDR				C
	w_2	w_4	w_2	w_1	w_4	w_9	w_8	w_6	w_2	w_1	w_4	w_9	
d_1	1	0	1	1	0	0	1	0	1	1	0	0	A
d_2	0	1	0	1	1	1	1	0	0	1	1	1	A
d_3	0	1	0	1	1	1	1	0	0	1	1	1	A
d_4	1	0	1	0	0	0	0	0	1	0	0	0	A
d_5	1	0	1	0	0	0	1	0	1	0	0	0	A
d_6	0	1	0	0	1	0	0	1	0	0	1	0	B
d_7	0	1	0	0	1	1	0	1	0	0	1	1	B
d_8	0	1	0	1	1	0	0	0	0	1	1	0	B
d_9	0	1	0	0	1	0	0	1	0	0	1	0	B
d_{10}	0	1	0	0	1	1	0	1	0	0	1	1	B
d_{11}	0	1	0	1	1	0	0	0	0	1	1	0	B
d_{12}	0	1	0	0	1	1	0	0	0	0	1	1	B
d_{13}	0	1	0	1	1	0	0	1	0	1	1	0	B

4.2. A toy example for MFDR

The first step of MFDR is to calculate the values of the vectors S and DR. The S vector is calculated as described in Section 4.1 and the DR vector is calculated using Eq. (1). The threshold mDR is the mean of the DR vector. For the toy example shown in Table 1, $mDR \approx 33.8$.

The second step is to select the best features. For each document that has DR higher than mDR, the f best valued features are selected based on the S vector; consider $f = 2$.

Since $DR(d_1) > mDR$, the document d_1 is verified and the algorithm selects the two words that has the highest values in S; in this case: w_2 and w_1. After, FS is updated to the set $\{2, 1\}$ and the next document is analyzed.

For d_2, again its DR value is higher than mDR, so w_4 and w_9 are selected; now, $FS = \{2, 1, 4, 9\}$. For d_3, the two selected words (w_4 and w_9) are already in FS, so no update occurs. The DR of d_4 is smaller than mDR. This indicates that the document should be ignored and the algorithm proceeds to the next document. d_5 and d_6 are also ignored since their DR values are smaller than mDR. From d_7 to d_{13}, despite their higher values of DR, no feature is added to FS.

The third step is to create the new training set based on the index of the four select features present in FS as shown in Table 2.

4.3. Comparing the proposed methods using the toy example

Table 2 shows the features selected by ALOFT, MFD, and MFDR using the toy dataset showed in Table 1. ALOFT selected only two features and these two features are not enough to discriminate the categories. Thus, documents d_2 and d_3 are misclassified because their feature vectors are equal to the feature vector of any document of the category B. This happens because ALOFT selects only one feature per document. So, if a feature has a high degree of significance, it is expected that this feature should be chosen by many documents. Therefore, an alternative to address this problem is to increase the number of features selected per document. Such solution is presented by the proposed MFD method that selects six features for the toy problem as shown in Table 2. The selected features discriminate well both categories, as shown by the following rules: if w_2 or w_8 are equal to "1", then classify

Table 3
Databases description.

Database	# Categories	# documents	# words
20 Newsgroup	20	19,997	46,834
Reuters 10	10	9,980	10,987
WebKB	4	4,199	21,324
TDT2	30	9,394	36,771

Table 4
Feature Evaluation Functions. w is the feature (word or term), c_j is the jth class and $P(\cdot)$ is the probability.

FEF	Function
Bi-Normal Separation (Forman, 2003)	$\mathrm{BNS}(w) = \sum_{j=1}^{C} \left\lvert F^{-1}(P(w \mid c_j)) - -F^{-1}(P(w \mid \bar{c}_j)) \right\rvert$
Class Discriminating Measure (Chen et al., 2009)	$\mathrm{CDM}(w) = \sum_{j=1}^{C} \left\lvert \log \frac{P(w\lvert c_j)}{P(w\lvert \bar{c}_j)} \right\rvert$
Chi-Squared (Debole & Sebastiani, 2003)	$\mathrm{CHI}(w) = \sum_{j=1}^{C} \frac{[P(w\lvert c_j)P(\bar{w}\bar{c}_j) - P(w\lvert \bar{c}_j)P(\bar{w}\lvert c_j)]^2}{P(w)P(\bar{w})P(c_j)P(\bar{c}_j)}$

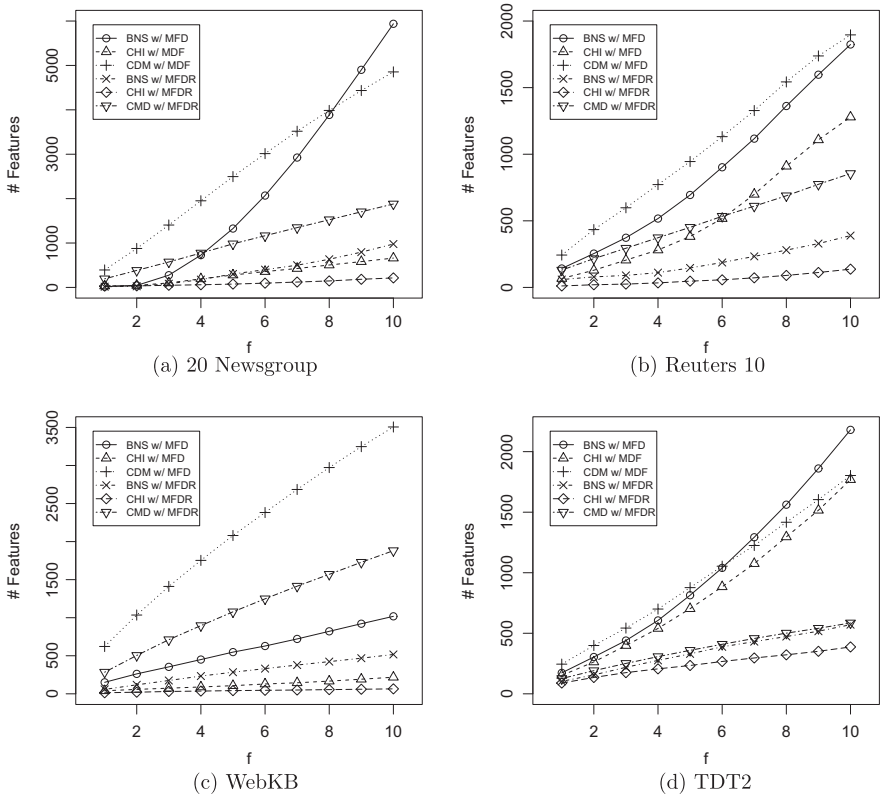


Fig. 1. Number of selected features for each database, proposed method and FEF.

as category *A*; otherwise classify as *B*. However, among the selected features are irrelevant ones.

In general, irrelevant features only hinder the classifier performance if they are misleading. In the presented toy dataset, the irrelevant features are not problematic. However, the chance of selecting misleading features increases when more features are selected per document because features with low FEF are added to the final subset. A proposal to avoid selecting features with low FEF is to ignore the documents that have a small number of relevant features, since these documents are more likely to present unnecessary features. As shown in Table 2, the proposed method MFDR selects less features than MFD and the selected features still discriminate both categories, for example: if w_2 is equal to "1", classify as category *A*; or if w_1 is equal to w_9, classify as *A*; otherwise, classify as *B*.

5. Experimental settings

The proposed methods (MFD and MFDR) were compared with ALOFT and VR. In order to have a fair comparison, the number of selected features in VR was defined as the same number found in the proposed methods. The experiments were performed with 10-fold stratified cross-validation, Naïve Bayes Multinomial Event Model classifier (McCallum & Nigam, 1998), four databases, three FEFs and ten *f* values (1 to 10).

Preliminary experiments were performed to verify the best values of *f*. In those experiments, *f* ranges from 1 to 15. However, we observed that for values of *f* greater than 10, the gain was minimal or the results do not show any relevant improvement. So, we decided to present only the results for the *f* values that ranges from 1 to 10.

The proposed techniques are evaluated using Micro averaged and macro averaged F1 evaluation measures. Micro averaged F1 gives an overall performance, whereas macro averaged F1 focuses on rare categories since it uses the same weight for each class regardless of its size (Taşcı & Güngör, 2013). Both measures are calculated as:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{(\text{Recall} + \text{Precision})}. \quad (3)$$

However, Precision and Recall are calculated for micro averaged F1 using a global contingency table, while macro averaged F1 uses a contingency table for each class and takes the average. A detailed explanation and equations of both measures are found in Sebastiani (2002).

A subset of the Reuters-21578 Collection,[1] 20 Newsgroup corpus,[2] a subset of the WebKB corpus2 and a subset of TDT2[3] (Table 3) were the databases used for the experiments. In Reuters, only the 10 categories with more documents were considered to generate the subset Reuters 10 (Chang, Chen, & Liau, 2008; Chen et al., 2009; Shang et al., 2007). While in WebKB, only four categories (course, faculty, project and student) were used to generate the subset (Bekkerman, El-Yaniv, Tishby, & Winter, 2003; Nigam, McCallum, Thrun, & Mitchell, 1998; Xue & Zhou, 2009). For the TDT2 database, a subset having the 30 categories that present more documents was considered (Cai, Mei, Han, & Zhai, 2008). Bi-Normal Separation (BNS), Chi-Squared (CHI), and Class Discriminating Measure (CDM) (Table 4) were the evaluated FEFs. Those FEFs were the most effective and diverse among the five evaluated in our previous work (Pinheiro et al., 2012).

6. Analysis of experiments

Fig. 1 shows the number of selected features for the proposed approaches (MFD and MFDR) when the parameter *f* varies from 1

[1] Available at http://disi.unitn.it/moschitti/corpora.htm.
[2] Available at http://www.cs.cmu.edu/~textlearning/.
[3] Available at http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html.

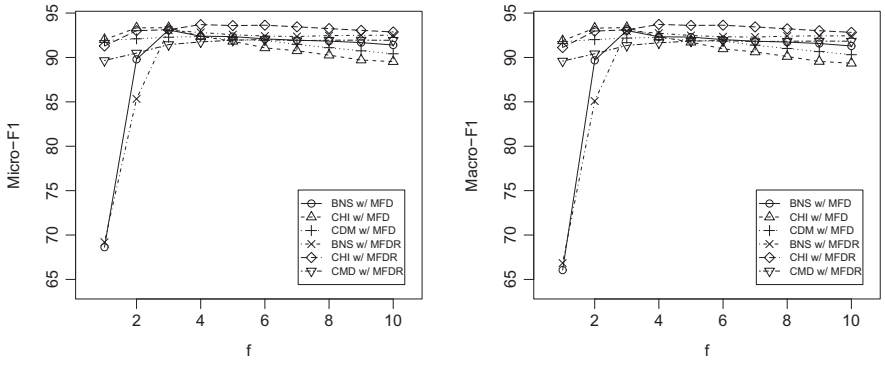


Fig. 2. Results for 20 Newsgroup with MFD and MFDR using each FEF.

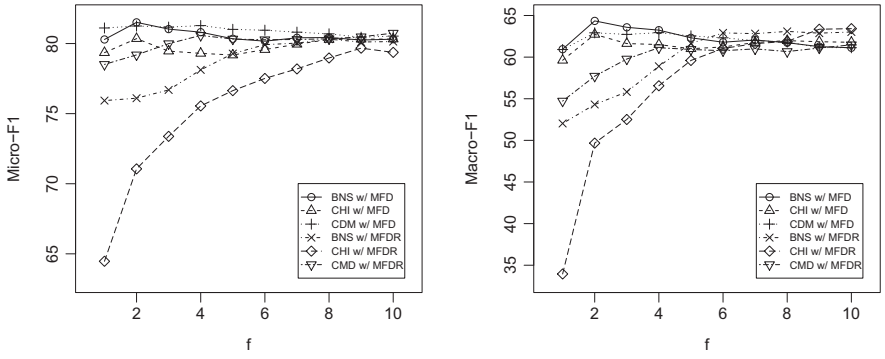


Fig. 3. Results for Reuters 10 with MFD and MFDR using each FEF.

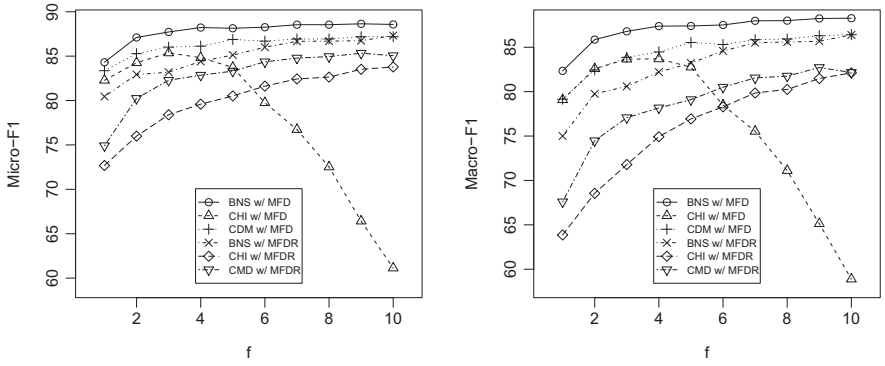


Fig. 4. Results for WebKB with MFD and MFDR using each FEF.

to 10. Four databases (20 Newsgroup, Reuters 10, WebKB and TDT2) are evaluated using three different FEFs (BNS, CHI, and CDM). Regardless of the database, CHI selects the smallest number of features. In general, CDM selects more features than other FEFs. Two exceptions were 20 Newsgroup and TDT2 because BNS presented an exponential increase in the number of selected features.

Figs. 2–5 show the results of the proposed methods in terms of Macro-F1 and Micro-F1 (Sebastiani, 2002). In general, MFDR presents a slower grow in performance when the number of features increase compared with MFD because it selects less feature than MFD. Therefore, for smaller values of f and comparing the same FEF, the performance of MFD is better than MFDR.

In 20 Newsgroup, the results of Macro-F1 and Micro-F1 (Fig. 2) are equivalent because this database has almost the same number of instances per class. For both proposed methods, BNS presents low performance when $f = 1$ because the number of selected features ($\approx$17) are not enough to discriminate all the 20 categories. The best f value for the proposed methods, MFD and MFDR, oscillates between 3 and 4 depending on FEF.

Reuters 10 is an unbalanced database where the largest category has 8.37 times more documents than the smallest category. The results of Macro-F1 and Micro-F1 (Fig. 3) show that the proposed methods present a similar behavior. MFDR with CHI presented the worst performance when $f = 1$ because the number of selected features ($\approx$11) was not enough to discriminate all the 10 categories. The best f value for MFD is between 2 and 4, and for MFDR between 8 and 10.

In the WebKB dataset (Fig. 4), the behavior of the macro-F1 and micro-F1 measures are almost the same. We observed that the results in WebKB continues to grow when the f value grows (except for CHI that have the best performance with $f = 3$). So, we run more experiments with different values of $f = \{11, 12, 13, 14, 15\}$, and observed a stabilization of the results

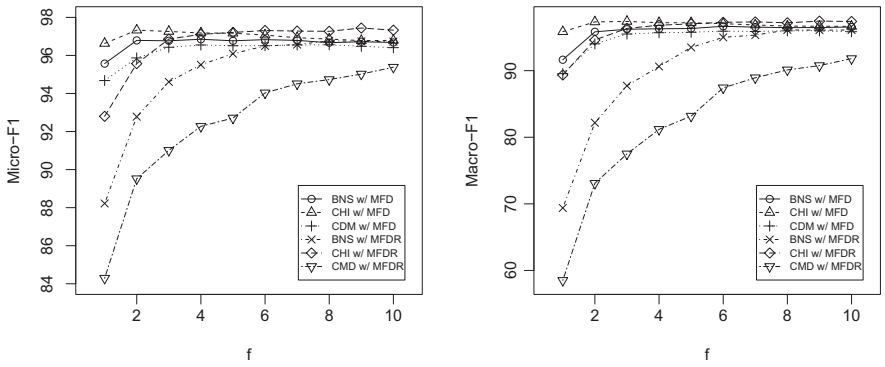


Fig. 5. Results for TDT2 with MFD and MFDR using each FEF.

Table 5
Best Micro-F1 results for ALOFT, MFD, and MFDR. Standard deviation is in parentheses and the best results for each database is in bold. Underlined results are significantly better then ALOFT.

Database	FEF	ALOFT	MFD	MFDR
20 Newsgroup	BNS	68.63 (4.71)	93.09 (0.69)	93.07 (0.91)
	CHI	92.01 (1.02)	93.35 (0.56)	**93.70 (0.94)**
	CDM	91.90 (0.65)	92.36 (0.66)	92.02 (0.70)
Reuters 10	BNS	80.29 (1.49)	**81.51 (1.39)**	80.31 (1.84)
	CHI	79.35 (2.17)	80.35 (2.00)	79.67 (1.83)
	CDM	81.11 (2.74)	81.28 (2.19)	80.74 (1.88)
WebKB	BNS	84.31 (4.94)	**88.64 (2.62)**	86.74 (3.12)
	CHI	82.28 (4.18)	85.36 (3.46)	83.78 (2.42)
	CDM	83.36 (2.22)	87.19 (3.29)	85.33 (3.05)
TDT2	BNS	95.57 (1.46)	96.85 (0.92)	96.77 (0.77)
	CHI	96.63 (0.97)	97.33 (0.97)	**97.44 (0.87)**
	CDM	94.68 (1.23)	96.56 (0.94)	95.38 (0.97)

Table 6
Number of selected features (m) and its respective f value for the Best Micro-F1 results for ALOFT, MFD, and MFDR (matches Table 5). Standard deviation of number of selected features is in parentheses. Value of f is not necessary in ALOFT.

Database	FEF	ALOFT	MFD		MFDR	
			f	m	f	m
20 Newsgroup	BNS	18 (1.00)	3	275 (9.04)	3	83 (4.00)
	CHI	32 (0.00)	3	101 (3.56)	4	60 (1.41)
	CDM	393 (8.10)	4	1951 (21.97)	6	1164 (17.65)
Reuters 10	BNS	142 (3.74)	2	254 (5.23)	8	280 (9.40)
	CHI	67 (2.89)	2	129 (5.29)	10	855 (11.78)
	CDM	243 (7.42)	4	772 (21.89)	9	112 (3.70)
WebKB	BNS	151 (11.22)	9	921 (42.28)	9	468 (15.28)
	CHI	40 (3.21)	3	74 (4.12)	10	65 (2.65)
	CDM	622 (13.66)	9	3248 (53.65)	9	1728 (33.59)
TDT2	BNS	173 (7.53)	4	606 (12.23)	8	473 (14.47)
	CHI	152 (4.65)	2	263 (3.32)	9	351 (5.32)
	CDM	245 (9.47)	8	1417 (20.27)	10	585 (13.99)

or a decrease depending on FEF. The MFDR method does not suffer from this problem even when selecting more features. When $f = 15$, MFDR selects 92 features and its result is better than MFD using any f.

In the TDT2 database (Fig. 5), macro-F1 and micro-F1 also presented a similar behavior of growth, changing only the range of the values: 58–97 and 84–97, respectively. MFD obtained the maximum performance with $f = 2$ or $f = 3$ depending on FEF. Increasing the f value, the performance decreased. Whereas, MFDR continues to grow even when $f > 10$ for CMD.

Our analysis show that macro-F1 and micro-F1 have almost the same behavior. So, the next analysis only considers micro-F1. Table 5 shows the best Micro-F1 for each method; standard deviations are in parentheses. The proposed methods achieved better or similar results compared with ALOFT. The best results are presented in bold and the underlined results indicate that they are better than ALOFT based on the t-test ($\alpha = 0.05$). So, for the four evaluated databases, the proposed approach presented significantly better results than ALOFT. Table 6 shows the number of selected features (m) and its respective f value for the Best Micro-F1 results presented in Table 5.

Table 7 compares MFD and MFDR against VR with a statistical t-test of combined variance. The leverage of MFD over VR decreases when f increases. Since MFD selects only valued features in each document, there is a chance of selecting features with low FEF values that should negatively impact the classifier. Despite the proposed methods decreasing performance, they are better or similar in 95% of cases.

Table 7
T-test results comparing MFD x VR and MFDR x VR. MFD results for $f = 1$ are reported in (Pinheiro et al., 2012). The code adopted for the *P*-value: "$\gg$" and "$\ll$" for *P*-values less than or equal to 0.01, this indicates a strong evidence of that a method has a greater or minor value of effectiveness measure than another one respectively; ">" and "<" for *P*-values greater than 0.01 and less than or equal to 0.05, this indicates a weak evidence that a method has a greater or minor value of effectiveness measure than another one respectively; "$\sim$" for *P*-values greater than 0.05, this indicates that the difference between both methods are not significant.

Database	f	MFD × VR			MFDR × VR		
		BNS	CHI	CDM	BNS	CHI	CDM
20 Newsgroup	$f = 1$	$\gg$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$
	$f = 2$	$\gg$	$\sim$	$\gg$	$\gg$	$\sim$	$\gg$
	$f = 3$	$\gg$	$\sim$	$\gg$	$\gg$	$\sim$	$\gg$
	$f = 4$	$\gg$	<	$\sim$	$\gg$	$\sim$	$\gg$
	$f = 5$	$\gg$	$\ll$	$\sim$	$\gg$	$\sim$	$\gg$
	$f = 6$	$\gg$	$\ll$	$\sim$	$\gg$	$\sim$	$\gg$
	$f = 7$	$\gg$	$\ll$	$\sim$	$\gg$	$\sim$	>
	$f = 8$	$\gg$	$\ll$	$\sim$	$\gg$	$\sim$	>
	$f = 9$	$\gg$	$\ll$	$\sim$	$\gg$	$\sim$	>
	$f = 10$	>	$\ll$	$\sim$	$\gg$	$\sim$	$\sim$
Reuters10	$f = 1$	$\gg$	$\sim$	>	>	$\sim$	$\gg$
	$f = 2$	$\gg$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 3$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 4$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 5$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 6$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 7$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 8$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 9$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
	$f = 10$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
WebKB	$f = 1$	$\sim$	>	$\gg$	$\gg$	$\sim$	$\gg$
	$f = 2$	>	$\sim$	$\gg$	$\gg$	$\sim$	$\gg$
	$f = 3$	>	$\sim$	$\gg$	$\sim$	$\sim$	$\gg$
	$f = 4$	>	$\sim$	$\gg$	$\sim$	$\sim$	$\gg$
	$f = 5$	$\sim$	$\sim$	$\gg$	$\sim$	$\sim$	$\gg$
	$f = 6$	$\sim$	$\ll$	$\gg$	$\sim$	$\sim$	$\gg$
	$f = 7$	$\sim$	$\ll$	>	$\sim$	$\sim$	$\gg$
	$f = 8$	$\sim$	$\ll$	>	$\sim$	$\sim$	$\gg$
	$f = 9$	$\sim$	$\ll$	>	$\sim$	$\sim$	$\gg$
	$f = 10$	$\sim$	$\ll$	$\sim$	$\sim$	$\sim$	$\gg$
TDT2	$f = 1$	$\gg$	$\gg$	$\gg$	$\gg$	$\sim$	$\gg$
	$f = 2$	$\gg$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$
	$f = 3$	$\gg$	$\sim$	$\gg$	$\gg$	>	$\gg$
	$f = 4$	$\sim$	$\sim$	$\gg$	$\gg$	$\sim$	$\sim$
	$f = 5$	$\sim$	$\sim$	>	$\gg$	$\sim$	$\sim$
	$f = 6$	$\sim$	$\sim$	$\sim$	$\gg$	$\sim$	$\gg$
	$f = 7$	$\sim$	$\sim$	$\sim$	$\gg$	$\sim$	>
	$f = 8$	$\sim$	$\sim$	$\sim$	$\gg$	$\sim$	>
	$f = 9$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	>
	$f = 10$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\gg$

In regards of FEF, CHI brings the worst contribution to the proposed methods, losing in 12 cases out of 80. The most difficult base for CHI was 20 Newsgroup, where VR wins in 35% of the comparisons. BNS and CDM show 0% loses against VR in all databases. The margin is greater in 20 Newsgroup for BNS with 100% victory and in WebKB for CDM with 95% of victory. MFDR is better than MFD when compared with VR. Because MFDR ignores some documents, probably avoiding low FEF features of those ignored documents; and selects less features, preventing VR from progress.

7. Conclusion

This paper presented two feature selection methods: Maximum f Features per Document (MFD), and Maximum f Features per Document – Reduced (MFDR). MFD and MFDR obtain better or similar performance when compared with the well established Variable Ranking (VR) method (Forman, 2003; Yang & Pedersen, 1997) and the ALOFT method (Pinheiro et al., 2012).

Both proposed methods have a parameter (f) that requires much less computational effort to be adjusted than the parameter (n) of the VR method. In our experiments, the parameter f (MDF and MFDR) varied from 1 to 10, while the parameter n (VR) varied from 11 to 5938. So, the proposed methods requires less time to find a better subset of features than the VR method.

The best values for f, independent of the database, were: (i) $f < 8$ for MFD, because when using higher values of f, more features with low FEF values are selected hindering the quality of the subset of features; and (ii) $7 < f \leqslant 10$ for MFDR, because when using lower values of f, a small number of features is selected and some important features are not included in the final subset.

Despite the good results of the proposed methods, it is important to select an FEF that is suitable for the database under evaluation. So, as future work, we plan to develop an automatic algorithm: (i) to determine the best f value for the proposed methods; (ii) to remove redundant features; and, (iii) to select the best FEF per database.

Acknowledgment

This research was partially supported by the Brazilian agency FACEPE.

References

Baccianella, S., Esuli, A., & Sebastiani, F. (2013). Using micro-documents for feature selection: The case of ordinal text classification. *Expert Systems with Applications, 40*, 4687–4696.

Bekkerman, R., El-Yaniv, R., Tishby, N., & Winter, Y. (2003). Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research, 3*, 1183–1208.

Bermejo, P., Gámez, J., & Puerta, J. (2014). Speeding up incremental wrapper feature subset selection with Naive Bayes classifier. *Knowledge-Based Systems, 55*, 140–147.

Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). Classification and regression trees. Wadsworth International Group

Cai, D., Mei, Q., Han, J., & Zhai, C. (2008). Modeling hidden topics on document manifold. In *Proceeding of the 17th ACM conference on information and knowledge management (CIKM'08)* (pp. 911–920).

Chang, Y., Chen, S., & Liau, C. (2008). Multilabel text categorization based on a new linear classifier learning method and a category-sensitive refinement method. *Expert Systems with Applications, 34*, 1948–1953.

Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with Naive Bayes. *Expert Systems with Applications, 36*, 5432–5435.

Debole, F., & Sebastiani, F. (2003). Supervised term weighting for automated text categorization. In *Proceedings of the ACM symposium on applied computing* (pp. 784–788). ACM.

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

Fragoudis, D., Meretakis, D., & Likothanassis, S. (2005). Best terms: An efficient feature-selection algorithm for text categorization. *Knowledge and Information Systems, 8*, 16–33.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157–1182.

John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the international conference on machine learning* (pp. 121–129).

Lewis, D. D. (1992). Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on speech and natural language, association for computational linguistics* (pp. 212–217).

Manning, C., Raghavan, P., & Schtze, H. (2008). *Introduction to information retrieval.* Cambridge University Press.

McCallum, A., & Nigam, K. (1998) . A comparison of event models for Naive Bayes text classification. In *Workshop on learning for text categorization* (pp. 41–48).

Meiri, R., & Zahavi, J. (2006). Using simulated annealing to optimize the feature selection problem in marketing applications. *European Journal of Operational Research, 171*, 842–858.

Mengle, S., & Goharian, N. (2009). Ambiguity measure feature-selection algorithm. *Journal of the American Society for Information Science and Technology, 60*, 1037–1050.

Mladenic, D., & Grobelnik, M. (1998). Feature selection for classification based on text hierarchy. In *Text and the web, conference on automated learning and discovery, Citeseer*.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *Proceedings of the national conference on artificial intelligence* (pp. 792–799).

Pinheiro, R., Cavalcanti, G., Correa, R., & Ren, T. (2012). A global-ranking local feature selection method for text categorization. *Expert Systems with Applications, 39*, 12851–12857.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Rogati, M., & Yang, Y. (2002). High-performing feature selection for text classification. In *Proceedings of the international conference on information and knowledge management* (pp. 661). ACM.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*, 1–47.

Shang, W., Huang, H., Zhu, H., Lin, Y., Qu, Y., & Wang, Z. (2007). A novel feature selection algorithm for text categorization. *Expert Systems with Applications, 33*, 1–5.

Taşcı, Ş., & Güngör, T. (2013). Comparison of text feature selection policies and using an adaptive framework. *Expert Systems with Applications, 40*, 4871–4886.

Uysal, A., & Gunal, S. (2012). A novel probabilistic feature selection method for text classification. *Knowledge-Based Systems, 36*, 226–235.

Xue, X., & Zhou, Z. (2009). Distributional features for text categorization. *IEEE Transactions on Knowledge and Data Engineering, 21*, 428–442.

Yang, Y., & Pedersen, J. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the international conference on machine learning* (pp. 412–420).

Yang, J., Liu, Y., Liu, Z., Zhu, X., & Zhang, X. (2011). A new feature selection algorithm based on binomial hypothesis testing for spam filtering. *Knowledge-Based Systems, 24*, 904–914.

Yang, J., Liu, Y., Zhu, X., Liu, Z., & Zhang, X. (2012). A new feature selection based on comprehensive measurement both in inter-category and intra-category for text categorization. *Information Processing and Management: An International Journal, 48*, 741–754.

Yan, J., Liu, N., Zhang, B., Yan, S., Chen, Z., Cheng, Q., et al. (2005). OCFS: Optimal orthogonal centroid feature selection for text categorization. In *Proceedings of the ACM SIGIR conference on research and development in information retrieval* (pp. 122–129). ACM.

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the international conference on machine leaning* (pp. 856–863).

Text categorization based on dissimilarity representation and prototype selection

Roberto H. W. Pinheiro, George D. C. Cavalcanti and Tsang Ing Ren
Universidade Federal de Pernambuco (UFPE) - Centro de Informática (CIn)
Av. Jornalista Anibal Fernandes s/n, Cidade Universitária 50740-560 - Brazil
Emails: {rhwp, gdcc, tir}@cin.ufpe.br
www.cin.ufpe.br/~viisar

***Abstract*—Bag-of-Words is the most used representation in text categorization, however it has some problems because its representation produces sparse high-dimensional feature vectors and have high feature-to-instance ratio. Feature selection is the most common approach to alleviate these problems. However, feature selection does not solve all the problems and information is lost in the process. In this paper, we propose a method based on dissimilarity representation and prototype selection to address these problems. Dissimilarity representation reduces the problems of Bag-of-Words and prototype selection is used to select a smaller representation set, increasing the benefits of using dissimilarity representation. The experimental study evaluated the effectiveness of the proposed method on four text categorization databases (RCV1, Reuters, TDT2, and WebKB) using Support Vector Machines. The proposed method reduces the number of features in 79% on average and presents better, or similar, results in 84% of the cases when compared with the Bag-of-Words approach.**

I. Introduction

Bag-of-Words (BoW) is the most used representation for text categorization problems [1]. In BoW, each document is represented by a feature vector, where each feature is a term that appears in any document of the *Corpus* (database of documents). Each feature can be i) binary, presence or absence; ii) integer, term frequency; or iii) real, term frequency-inverse document frequency (TF-IDF). Regardless of the feature type, terms that do not appear in a given document have value zero.

Despite the large number of papers using BoW and achieving good results [2], [3], it faces some difficulties: i) high-dimensionality feature vector [4], since all terms of the *Corpus* are considered; ii) sparse matrix, because every documents has only a small percentage of all terms; and, iii) high feature-to-instance ratio, because every document adds a whole new set of features to the vector. Those difficulties can be alleviated – not removed – with feature selection. However, a very aggressive feature selection is necessary to address those three difficulties and such approach degrades the accuracy rate of the system [5].

In this paper, we propose a text categorization method based on dissimilarity representation [6] and prototype selection [7]. Instead of using terms as features, in the proposed approach, the features are defined as the cosine similarity between two documents. Therefore, a document is represented by a feature vector of similarities. Those similarities are calculated between the original document and documents of a representation set obtained by a prototype selection algorithm. In this manner, all three previously pointed difficulties of text categorization are addressed: i) the proposed method works as a feature extraction method, reducing the high-dimensionality of the task; ii) since the features are similarities between documents, the number of features with zero weight are smaller in comparison to BoW – this reduces the sparseness; and, iii) high feature-to-instance ratio cannot be higher than 1 (one), which would be the case of using all training set as the representation set. The prototype selection algorithm finds a smaller representation set and increases the benefits of using dissimilarity representation.

This paper is organized as follows. Section II describes the dissimilarity representation and introduces prototype selection. Section III presents the proposed method. Section IV describes the methodology of the experiments, reports and analyses the experimental results. Finally, Section V concludes this paper.

II. Basic concepts

A. Dissimilarity Representation

The dissimilarity representation [6] is a mapping defined by calculating the dissimilarity of a document (or set of documents) to every document of a representation set. The representation set $\mathbf{R}$ is a subset of the training set ($\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$) composed of Q prototypes ($\mathbf{R} = \{p_1, \ldots, p_Q\}$). The number of prototypes (Q) is defined by the user or automatically with a prototype selection algorithm. Figure 1 shows an example of transition, from the original representation ($w_1, w_2, \ldots, w_V$) composed of V different terms to the dissimilarity representation, obtained by calculating the distance $d(\cdot,\cdot)$ from all documents of $\mathbf{X}$ to all documents of $\mathbf{R}$, resulting in the dissimilarity matrix $D(\mathbf{X}, \mathbf{R})$.

The main reason for using dissimilarity representation is to take advantage of distances properties [8]. Distances between objects from the same class should be small, while distance between objects from different classes should be large. Therefore, having a prototype p_k, representing well one specific class, generates a powerful feature, because the distances from p_k probably will have a good discriminative power.

Other reason for using dissimilarity representation is to reduce the problems in the original feature space. The Bag-of-Words representation, for text categorization, suffers from sparseness, high-dimensionality and high feature-to-instance

978-1-5090-0016-6/15 $31.00 © 2015 IEEE
DOI 10.1109/BRACIS.2015.28

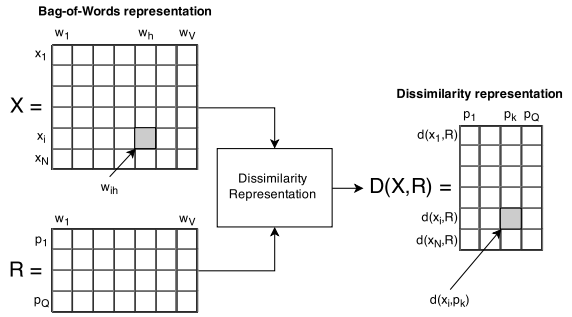


Fig. 1. Transition from the Bag-of-Words representation to the dissimilarity representation. $\mathbf{X}$ is the training set, $\mathbf{R}$ is the representation set and $D(\mathbf{X}, \mathbf{R})$ is the resulting dissimilarity matrix. The feature w_{ih} is the term frequency of the hth term of the ith document. The feature $d(x_i, p_k)$ is the dissimilarity from the document x_i and the prototype from a representation set p_k

ratio. In the dissimilarity space all of those problems are reduced. Sparseness, because $d(x_i, p_k)$ is only zero when both documents are equal, thus rarely occurs; dimensionality, because Q is usually smaller than V in text categorization problems; and feature-to-instance ratio, because $Q \leq N$.

B. Prototype Selection

Prototype selection algorithms reduce the size of the original training set. This reduction is obtained by selecting a subset of the patterns that improves or at least maintains the discrimination power of the original set. There are three types of prototype selection: edition, condensation and hybrid [7].

The edition prototype selection algorithms aim to remove noisy examples. It is expected to increase the performance after removing noisy data, because these examples confuse the classifier training. Condensation methods aim to remove irrelevant examples. The irrelevant examples are those that are redundant or simply do not impact on the classification. The idea behind removing irrelevant examples is simply to reduce the size of the training set without hindering the classification task. Finally, hybrid methods aim to remove noisy and irrelevant examples. The expected result is a small subset with some gain in performance.

The main reasons for using prototype selection are to improve the performance and reduce storage. The performance is improved because noisy data and redundant examples can mislead the classifier. The storage reduction is obvious, since only a subset of the training set is used. Additionally, prototype selection also increases the processing speed, since fewer examples are used to train the classifier.

Some prototype selection algorithms are used in this paper. The criterion of choice are diversity of types. The algoriths used are: CNN (Condensation Nearest Neighbor) [9], ENN (Edition Nearest Neighbor) [10], All-kNN (All k Nearest Neighbor) [11], DROP3 (Decremental Reduction Optimization Procedure 3) [12] and Atisa2 (Adaptive Threshold-based Instance Selection Algorithm 2) [13]. Which, according to the taxonomy [7] are condensation, edition decremental, edition batch and hybrid, respectively. The taxonomy was created before Atisa2 was published, thus we classified as a hybrid method. Atisa2 is the more balanced (reduction and accuracy) of the three Atisa algorithms [13] and it is a recent prototype selection algorithm.

III. Proposed Method

Figure 2 shows the proposed method that is composed of three main modules: Prototype Selection, Cosine Representation, and Classification. The first module uses a prototype selection algorithm to select a subset $\mathbf{R}$ of the original training set $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ ($\mathbf{R} \subseteq \mathbf{X}$); this module is described in Section III-A. The subset $\mathbf{R}$ is used to generate a new representation that aims to reduce the dimensionality, sparseness and feature-to-instance ratio of the original database. This is performed by the Cosine Representation module (Section III-B). The classifier is trained using the similarity matrix $S(\mathbf{X}, \mathbf{R})$. Given an unlabeled document y_j that belongs to the test set $\mathbf{Y} = \{y_1, y_2, \ldots, y_M\}$, the output of the classifier is the predicted class c'_j of the query document.

This section also explains why cosine is used in the proposed system (Section III-C) and presents a toy example of the proposed method (Section III-D).

A. Prototype Selection

There are several alternatives to obtain the representation set $\mathbf{R}$, such as: use all documents in the training set, select some documents at random or select the most dissimilar documents [6]. However, the alternative that uses all documents ($\mathbf{R} = \mathbf{X}$) includes irrelevant and/or redundant documents, random selection reduces the size of the representation but gives no guarantee about avoiding irrelevant document, and the last alternative may produce a representation set composed of outliers and/or noisy data. Herein, we use a prototype selection algorithm to deal with this task. It is expected that the prototype selection algorithm reduces and improves the representation set by removing redundant and irrelevant documents [7]. Moreover, the number of eliminated documents depends on the database under study, in other words, it is not necessary to determine the size of the representation set $\mathbf{R}$ in advance.

Each document of the training set $x_i \in \mathbf{X}$ is represented by a pair (w_i, c_i), where $w_i = [w_{i1}, w_{i2}, \ldots, w_{iV}]^{\mathrm{T}}$ is a vector of words using the BoW representation and $c_i \in \mathbf{C} = \{1, 2, \ldots, C\}$ is the class of the document. The documents of the training set $\mathbf{X}$ are given as input to a prototype selection algorithm that aims to select a subset $\mathbf{R} = \{p_1, p_2, \ldots, p_Q\}$ ($\mathbf{R} \subseteq \mathbf{X}$) of the original database.

B. Cosine Representation

During the training phase, $\mathbf{R}$ and $\mathbf{X}$ are used as input to the Cosine Similarity module to calculate the similarity matrix $S(\mathbf{X}, \mathbf{R})$. Each element $\cos(x_i, p_j)$ in $S(\cdot, \cdot)$ is the cosine similarity between a document $x_i \in \mathbf{X}$ and a document

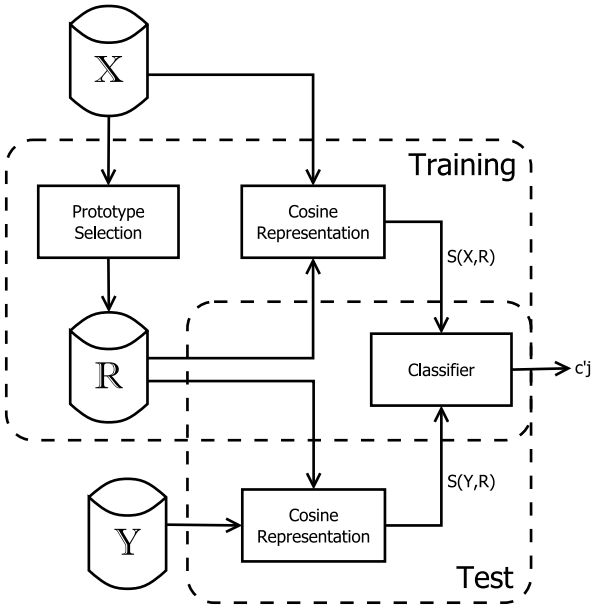


Fig. 2. Flowchart of the proposed method. $\mathbf{X}$ is the training set, $\mathbf{R}$ is the representation set, $\mathbf{Y}$ is the test set, $S(\cdot,\cdot)$ are similarity matrices and c'_j is the predicted class of a query document y_j

$p_j \in \mathbf{R}$. Therefore,

$$S(\mathbf{X},\mathbf{R}) = \begin{bmatrix} \cos(x_1,p_1) & \cos(x_1,p_2) & \dots & \cos(x_1,p_Q) \\ \cos(x_2,p_1) & \cos(x_2,p_2) & \dots & \cos(x_2,p_Q) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(x_N,p_1) & \cos(x_N,p_2) & \dots & \cos(x_N,p_Q) \end{bmatrix}$$

is a matrix with $N \times Q$ dimensions and the cosine similarity is defined as:

$$\cos(x_a,x_b) = \frac{\sum_{h=1}^{V} w_{ah} w_{bh}}{\sqrt{\sum_{h=1}^{V} w_{ah}^2}\sqrt{\sum_{h=1}^{V} w_{bh}^2}}, \qquad (1)$$

where x_a and x_b are any two documents, w_{ah} and w_{bh} are the hth features of x_a and x_b respectively.

The original documents are, now, represented in a space of similarity with Q dimensions. In other words, each document x_i is represented by a feature vector $S(x_i,\mathbf{R}) = [\cos(x_i,p_1),\cos(x_i,p_2),\dots,\cos(x_i,p_Q)]^{\mathrm{T}}$ and by its class c_i.

In the test phase, the procedure is quite similar. The representation set $\mathbf{R}$ obtained by a prototype selection algorithm during the training phase is used to generate the similarity matrix

$$S(\mathbf{Y},\mathbf{R}) = \begin{bmatrix} \cos(y_1,p_1) & \cos(y_1,p_2) & \dots & \cos(y_1,p_Q) \\ \cos(y_2,p_1) & \cos(y_2,p_2) & \dots & \cos(y_2,p_Q) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(y_M,p_1) & \cos(y_M,p_2) & \dots & \cos(y_M,p_Q) \end{bmatrix}$$

with dimensions $M \times Q$. However, the classifier is already trained. Thus, the classifier will response a class c'_j for each document $y_j \in \mathbf{Y}$.

C. Euclidean Distance versus Cosine Similarity

Defining a well-discriminating (dis)similarity measure can be a challenge [8]. The Euclidean distance is commonly used for Dissimilarity Representation [6]. However, the Euclidean distance behaves poorly for the feature distribution of text categorization problems.

The BoW representation generates a high dimensional sparse matrix. The Euclidean distance is not suitable for comparing two documents in this representation. Here follows an example to illustrate this issue: given two documents from the training set $x_1 = \{3,3,0,4,0,0,0,0,0,0\}$ of class A and $x_2 = \{0,0,0,0,0,0,0,3,2,0\}$ of class B and a query document $y_1 = \{3,0,2,0,0,0,0,0,0,0\}$ of class A. y_1 is classified as the training document closer to it (smaller distance). Consider $\mathrm{dist}(x_a,x_b) = \sqrt{\sum_{h=1}^{V}(w_{ah}-w_{bh})^2}$, then y_1 is classified as class B because $\mathrm{dist}(x_1,y_1) = 5.38$ is higher than $\mathrm{dist}(x_2,y_1) = 5.09$. In this case, y_1 is misclassified.

However, when using the cosine similarity [14] on the same example, y_1 is correctly classified as class A because $\cos(x_1,y_1) = 0.42$ is higher than $\cos(x_2,y_1) = 0$. This occurs because the cosine similarity concentrates only on the resemblance of values different than zero, avoiding the excess of zeros present in text categorization problems. Therefore, it is better to adopt cosine similarity instead of Euclidean distance.

D. Toy example

In order to illustrate how the proposed method works, a hypothetical training set (Table I) composed of 13 documents represented with 9 features was constructed. There is only two classes for this example ($\mathbf{C} = \{A,B\}$).

TABLE I
A TOY EXAMPLE. THE FIRST COLUMN (D) REPRESENTS AN IDENTIFIER TO EACH DOCUMENT, THE LAST COLUMN (C) REPRESENTS THE CATEGORY OF THE DOCUMENT AND THE COLUMNS BETWEEN $(w_1, w_2, \ldots, w_9)$ ARE THE FEATURES. EACH LINE REPRESENTS ONE DOCUMENT

D	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	C
d_1	2	1	5	0	0	0	1	5	0	A
d_2	3	0	0	3	2	0	0	6	4	A
d_3	1	0	0	1	0	0	0	5	2	A
d_4	0	6	0	0	0	0	0	0	0	A
d_5	0	2	0	0	0	0	1	1	0	A
d_6	0	0	0	6	0	6	0	0	0	B
d_7	0	0	0	3	0	3	0	0	6	B
d_8	1	0	1	4	2	0	0	0	0	B
d_9	0	0	0	2	0	2	1	0	0	B
d_{10}	0	0	0	2	0	2	2	0	7	B
d_{11}	5	0	1	3	1	0	5	0	0	B
d_{12}	0	0	0	4	2	0	0	0	3	B
d_{13}	2	0	0	4	0	3	0	0	0	B

The first step is to select some prototypes to obtain the representation set $\mathbf{R}$. For this example, consider that a prototype selection algorithm selected 4 prototypes: d_2, d_4, d_6, d_{11} ($Q = 4$). Therefore, $d_2 \rightarrow p_1$, $d_4 \rightarrow p_2$, $d_6 \rightarrow p_3$, $d_{11} \rightarrow p_4$ and $\mathbf{R} = \{p_1, p_2, p_3, p_4\}$.

The second step is to calculate the similarity matrix (Cosine Representation) using $\mathbf{X}$ and $\mathbf{R}$. The matrix is presented in Table II.

TABLE II
COSINE REPRESENTATION FOR THE TOY EXAMPLE

D	$p_1(d_2)$	$p_2(d_4)$	$p_3(d_6)$	$p_4(d_{11})$	C
d_1	0.55	0.43	0.00	0.34	A
d_2	1.00	0.28	0.24	0.38	A
d_3	0.91	0.36	0.12	0.18	A
d_4	0.00	0.81	0.00	0.00	A
d_5	0.28	1.00	0.00	0.26	A
d_6	0.24	0.00	1.00	0.27	B
d_7	0.52	0.00	0.57	0.15	B
d_8	0.47	0.00	0.60	0.54	B
d_9	0.23	0.13	0.94	0.46	B
d_{10}	0.50	0.10	0.36	0.26	B
d_{11}	0.38	0.26	0.27	1.00	B
d_{12}	0.60	0.00	0.52	0.33	B
d_{13}	0.38	0.00	0.91	0.52	B

The original matrix (Table I) has 72 zeros ($72/117 \approxeq 61\%$), while the Cosine Representation (Table II) has only 10 zeros ($10/52 \approxeq 19\%$). Despite reducing the number of features to 4, instead of the original 9, this is not the reason for the severe reduction of sparseness. If a feature selection is used with the premise to select the features with less zeros, the best reduction is to 26 zeros ($26/52 \approxeq 50\%$) with 4 features. Therefore, Cosine Representation gives a double benefit: reduce the number of features and reduce the sparseness.

IV. EXPERIMENTS AND RESULTS

The experiments were performed using the 10-fold stratified cross-validation method and the classifier was SVM [15] with linear kernel. Four databases of text categorization were used:

- The Reuters Corpus Volume 1 (RCV1)[1] contains news wire stories of Reuters from 1996 to 1997. The original dataset has more than 800,000 documents organized in a hierarchical structure of categories. We adopted a subset with 9,625 documents divided in 4 categories: ECAT, C15, GCAT and MCAT [16]. The vocabulary contains 29,992 words;
- The Reuters-21578 Collection[2] is a well-known text classification benchmark containing news wire stories of Reuters of 1987. The original dataset has 135 categories. We adopted the top ten categories subset with 9,980 documents [17], [18], [19]. The vocabulary contains 10,987 words;
- The TDT2 Corpus[1] is a collection created in 1998 using news wire, radio programs and television programs with 11,201 documents which were classified into 96 categories. We adopted the top thirty categories and only used documents with one class, resulting in 9,394 documents. The vocabulary contains 36,771 words;
- The WebKB corpus[3] is composed of web pages from four universities. The original dataset has 7 categories and 8,282 documents. We adopted a subset with only 4 categories: course, faculty, project and student [20]. This subset contains 4,199 documents. The vocabulary contains 21,324 words.

Macro averaged and micro averaged F1 were the evaluation measures [1].

The proposed method was tested with five different configurations: one without prototype selection and five with different prototype selection algorithms. The prototype selection algorithms were: CNN [9], ENN [10], All-kNN [11], DROP3 [12] and Atisa2 [13]. All algorithms performed with $k = 5$ and used cosine representation to find the nearest neighbors.

Table III shows the results of the proposed method and BoW. Based on a statistical t-test of combined variance, the proposed method won in 35% of the cases, lost only in 16% of the cases and it was considered similar in the rest of the cases. Also, the proposed method reduces the number of features. For instance, the proposed method with DROP3 in Reuters 10 reduced 95.05% of the features and obtained the best result in this database.

Figure 3 shows a plot with the average reduction and micro-F1 values for all evaluated datasets. CNN, DROP3, and Atisa2 were the most aggressive prototype selection algorithms with a reduction rate around 95%. Note that reduction occurs even without prototype selection because the dimensionality using cosine representation is, at most, equal to the number of documents in the training set. Since all databases have the number of documents higher than the number of features, the final number of features of the proposed approach, even without prototype selection, is smaller than in the BoW representation.

[1] Available at http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html.
[2] Available at http://disi.unitn.it/moschitti/corpora.htm.
[3] Available at http://cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

TABLE III

RESULTS OF BOW AND PROPOSED METHOD IN FOUR DATABASES. NAMES IN BOLD INDICATES THE BEST METHOD IN EACH DATABASE. THE REDUCTION COLUMN SHOWS HOW MUCH THE FEATURE VECTOR IS REDUCED. THE TABLE COMPARES BOW WITH PROPOSED METHOD USING T-TEST ($\alpha = 0.05$). THE COMPARISON IS SHOWED IN PARENTHESIS AFTER THE RESULTS AND IT IS BASED ON THE FOLLOWING ICONOGRAPHIC: WHEN P-VALUE IS LESS THAN OR EQUAL TO 0.01 ("$\gg$" AND "$\ll$") REPRESENTS A STRONG EVIDENCE OF THAT A METHOD HAS A GREATER OR MINOR VALUE OF EFFECTIVENESS MEASURE THAN ANOTHER ONE RESPECTIVELY; WHEN P-VALUES IS GREATER THAN 0.01 AND LESS THAN OR EQUAL TO 0.05 ("$>$" AND "$<$") REPRESENTS A WEAK EVIDENCE THAT A METHOD HAS A GREATER OR MINOR VALUE OF EFFECTIVENESS MEASURE THAN ANOTHER ONE RESPECTIVELY; WHEN P-VALUE IS GREATER THAN 0.05 ("$\sim$") REPRESENTS THAT THE DIFFERENCE BETWEEN BOTH METHODS ARE NOT SIGNIFICANT

Database	Method	Reduction (%)	macro-F1	micro-F1
RCV1	BoW	0.00	94.67	94.78
	Proposed without PS	71.12	94.84 ($>$)	94.94 ($>$)
	Proposed w/ CNN	95.09	94.62 ($\sim$)	94.74 ($\sim$)
	Proposed w/ ENN	72.69	94.89 ($\gg$)	94.99 ($\gg$)
	Proposed w/ All-kNN	73.97	94.77 ($>$)	94.89 ($>$)
	Proposed w/ DROP3	94.43	93.56 ($\ll$)	93.71 ($\ll$)
	Proposed w/ Atisa2	95.42	94.79 ($>$)	94.91 ($\gg$)
Reuters 10	BoW	0.00	54.01	79.36
	Proposed without PS	18.25	55.56 ($\sim$)	80.28 ($>$)
	Proposed w/ CNN	72.89	57.54 ($>$)	81.58 ($\gg$)
	Proposed w/ ENN	37.00	56.20 ($\sim$)	80.97 ($\gg$)
	Proposed w/ All-kNN	40.97	57.33 ($\sim$)	81.97 ($\gg$)
	Proposed w/ DROP3	95.05	62.77 ($\gg$)	85.58 ($\gg$)
	Proposed w/ Atisa2	88.99	61.69 ($\gg$)	85.05 ($\gg$)
TDT2	BoW	0.00	97.90	98.22
	Proposed without PS	77.01	98.03 ($\sim$)	98.27 ($\sim$)
	Proposed w/ CNN	97.12	97.57 ($<$)	97.97 ($\ll$)
	Proposed w/ ENN	77.72	97.92 ($\sim$)	98.22 ($\sim$)
	Proposed w/ All-kNN	78.08	97.86 ($\sim$)	98.19 ($\sim$)
	Proposed w/ DROP3	95.62	96.79 ($\ll$)	97.35 ($\ll$)
	Proposed w/ Atisa2	97.20	97.66 ($\sim$)	98.04 ($\sim$)
WebKB	BoW	0.00	90.66	91.55
	Proposed without PS	82.28	92.78 ($\sim$)	93.26 ($\sim$)
	Proposed w/ CNN	92.92	92.48 ($\sim$)	93.05 ($\sim$)
	Proposed w/ ENN	86.10	92.27 ($\sim$)	92.88 ($\sim$)
	Proposed w/ All-kNN	88.16	92.07 ($\sim$)	92.67 ($\sim$)
	Proposed w/ DROP3	93.68	86.94 ($<$)	87.97 ($\ll$)
	Proposed w/ Atisa2	93.73	92.15 ($\sim$)	92.90 ($\sim$)

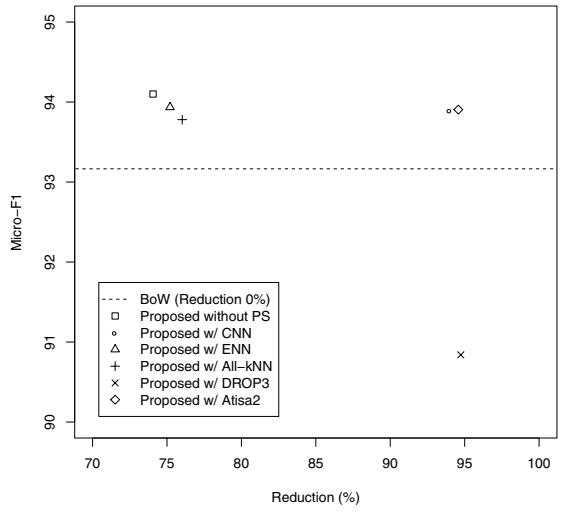


Fig. 3. Average reduction and micro-F1 values for all evaluated datasets

The best configuration of the proposed system was obtained using the Atisa2 algorithm. It reduced more features (except in Reuters 10 where it obtained the second best reduction rate) and obtained the balance of average performance and average reduction when compared with configurations of the proposed method.

V. CONCLUSION

This paper presented a new classification method for text categorization based on dissimilarity representation and prototype selection. The proposed method reduces three well-known problems of BoW: high-dimensionality, sparse matrix and high feature-to-instance ratio.

The experiments show that the proposed method achieved better or similar performance in 84% of the cases when compared with BoW. Furthermore, the proposed method reduces the number of features (79% in average).

One of the modules of the proposed systems requires a prototype selection algorithm. The Atisa2 algorithm was the best prototype algorithm. But, it is important to note that the proposed system without prototype selection obtained better average macro-F1 and micro-F1 values than BoW. So, even without using prototype selection, the proposed system still reduces the size of the feature vector and increases the accuracy rates.

For future work, we intend to verify the behavior of the proposed system in a multiple classifier system where different prototype selection algorithms will be used for different classifiers and these classifiers will be combined in order to increase the overall accuracy rates. Moreover, we plan to select the best prototype selection method per database using meta-learning as described in [21].

ACKNOWLEDGMENT

This research was partially supported by the following Brazilian agencies: FACEPE and CNPq.

REFERENCES

[1] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[2] R. Pinheiro, G. Cavalcanti, R. Correa, and T. Ren, "A global-ranking local feature selection method for text categorization," *Expert Systems with Applications*, vol. 39, no. 17, pp. 12 851–12 857, 2012.

[3] R. Pinheiro, G. Cavalcanti, and T. Ren, "Data-driven global-ranking local feature selection methods for text categorization," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1941–1949, 2015.

[4] D. Lewis, "Feature selection and feature extraction for text categorization," in *Workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 212–217.

[5] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," in *International Conference on Machine Learning*, 1997, pp. 412–420.

[6] E. Pekalska and R. Duin, "Dissimilarity representations allow for building good classifiers," *Pattern Recognition Letters*, vol. 23, no. 8, pp. 943–956, 2002.

[7] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[8] E. Pekalska, P. Paclik, and R. Duin, "A generalized kernel approach to dissimilarity-based classification," *The Journal of Machine Learning Research*, vol. 2, pp. 175–211, 2002.

[9] P. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[10] D. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.

[11] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 6, pp. 448–452, 1976.

[12] D. R. Wilson and T. R. Martinez, "Instance pruning techniques," in *Proceedings of the International Conference on Machine Learning*, vol. 97, 1997, pp. 403–411.

[13] G. Cavalcanti, T. Ren, and C. L. Pereira, "ATISA: Adaptive Threshold-based Instance Selection Algorithm," *Expert Systems with Applications*, vol. 40, no. 17, pp. 6894–6900, 2013.

[14] G. Salton and D. Harman, *Information retrieval*. John Wiley and Sons Ltd., 2003.

[15] I. Steinwart and A. Christmann, *Support vector machines*. Springer, 2008.

[16] D. Cai and X. He, "Manifold adaptive experimental design for text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 707–719, 2012.

[17] J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with Naive Bayes," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5432–5435, 2009.

[18] Y. Chang, S. Chen, and C. Liau, "Multilabel text categorization based on a new linear classifier learning method and a category-sensitive refinement method," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1948–1953, 2008.

[19] W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang, "A novel feature selection algorithm for text categorization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 1–5, 2007.

[20] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Learning to classify text from labeled and unlabeled documents," in *National Conference on Artificial Intelligence*, 1998, pp. 792–799.

[21] S. de Oliveira Moura, M. B. de Freitas, H. A. C. Cardoso, and G. D. C. Cavalcanti, "Choosing instance selection method using meta-learning," in *IEEE International Conference on Systems, Man and Cybernetics*, 2014, pp. 2003–2007.