Knowledge-Based Systems 59 (2014) 173–181

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A multi-instance learning wrapper based on the Rocchio classifier for web index recommendation

Dánel Sánchez Tarragó [a], Chris Cornelis [b,*], Rafael Bello [a], Francisco Herrera [b,c]

[a] Department of Computer Science, Central University of Las Villas, Cuba
[b] Department of Computer Science and Artificial Intelligence, Research Center on Information and Communications Technology (CITIC-UGR), University of Granada, Spain
[c] Faculty of Computing and Information Technology – North Jeddah, King Abdulaziz University, Saudi Arabia

ARTICLE INFO

Article history:
Received 31 July 2013
Received in revised form 23 October 2013
Accepted 8 January 2014
Available online 30 January 2014

Keywords:
Multi-instance learning
Multi-instance wrapper
Recommender system
Rocchio classifier
Web mining

ABSTRACT

Web index recommendation systems are designed to help internet users with suggestions for finding relevant information. One way to develop such systems is using the multi-instance learning (MIL) approach: a generalization of the traditional supervised learning where each example is a labeled bag that is composed of unlabeled instances, and the task is to predict the labels of unseen bags. This paper proposes a multi-instance learning wrapper method using the Rocchio classifier to recommend web index pages. The wrapper implements a new way to relate the instances with the class labels of the bags. The proposed method has low computational cost and the experimental study on benchmark data sets shows that it performs better than the state-of-the-art methods for this problem.

1. Introduction

Multi-instance learning (MIL) is a generalization of traditional supervised learning. Since it was formalized by Dietterich et al. [8] in 1997, the interest of the machine learning scientific community in this approach has grown rapidly because of its suitability for solving certain types of complex problems. Such problems are characterized by the fact that each example has multiple forms of representation or, alternatively, consists of multiple parts [7], or represents multiple samples from a stochastic process [2]. Consequently, an example is not described by a single feature vector, but by a bag of feature vectors which are called instances. The class labels of instances are not known, only those of the bags. The goal is to classify new bags based on the description of its instances.

In this paper, we apply the MIL approach to the development of recommendation systems for web index pages. This is a kind of web mining task aimed at assisting Internet users with web page suggestions. Web index pages are Internet pages that only provide titles or brief summaries while leaving the detailed presentation to their linked pages. The recommendation system learns a model for deciding whether a new web index page is relevant to one user on the basis of other web index pages this user has already browsed and classified as relevant or irrelevant, to provide the user with new potentially interesting web index pages.

The web index recommendation (WIR) problem is difficult because the information available about the user is ambiguous: we know whether the user is interested in a web index, but we do not know why, i.e., we do not know which linked pages make that web index relevant for the user. The relationship between a web index and its linked pages makes this problem a natural candidate for the MIL approach. Concretely, web indexes are considered as bags and their linked pages make up the individual instances.

The WIR problem was introduced in the context of MIL by Zhou et al. [37]. Fretcit-kNN [37], based on the well known kNN classifier, was the first MIL method designed specifically for WIR. More recently, two grammar-guided genetic programming methods, G3P-MI [33] and MOG3P-MI [32], were developed for this problem.

Our paper presents a new MIL method for the WIR problem that develops a wrapper algorithm using the Rocchio classifier [23]. The wrapper algorithm uses a new heuristic to assign class labels to instances and thus transforms the data of the MIL problem so that traditional simple-instance learning methods can be used. The Rocchio classifier is a classic method in the information retrieval field [1,25,26]. which has the advantage to be very fast and suitable for textual problems. We learn a Rocchio classifier from the instances that have been labeled by the wrapper, and use it to classify the instances of a new bag. We compare our proposed method with Fretcit-kNN and MOG3P-MI in an experimental study on nine WIR benchmark data sets introduced in [37], which demonstrates that our method exhibits excellent performance. We also show how our method can be optimized to take into

* Corresponding author. Tel.: +34 958 241774.
E-mail addresses: danels@uclv.edu.cu (D.S. Tarragó), chris.cornelis@decsai.ugr.es (C. Cornelis), rbellop@uclv.edu.cu (R. Bello), herrera@decsai.ugr.es (F. Herrera).

http://dx.doi.org/10.1016/j.knosys.2014.01.008
0950-7051/© 2014 Elsevier B.V. All rights reserved.

account the imbalance between positive and negative bags present in some of the data sets.

The rest of this paper is organized as follows. In the next section, we recall formal aspects that relate to the MIL problem, and give a brief overview of MIL approaches. Section 3 introduces the WIR problem and summarizes representation forms and solution methods that have been used in previous studies to solve this problem. Section 4 presents the details of the method we propose. In Section 5, experimental results and discussions are provided. Finally, we offer concluding remarks in Section 6.

2. Multi-instance learning

This section introduces the fundamentals of multi-instance learning (MIL). Our MIL definition is stated in Section 2.1; assumptions that drive an algorithm to the solution of MIL problems, and that are relevant to this paper, are examined in Section 2.2; and finally, we present a categorization of MIL methods in Section 2.3.

2.1. Multi-instance learning definition

MIL is a kind of supervised learning. In this setting, we consider a training set $D = \{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$ of labeled examples, where x_i is the description of the i-th example and $f(x_i) \in L$ the label attached to it.

In the *traditional* (simple-instance) supervised learning scenario, each example x_i consists of a single instance represented by a fixed-length vector of features, i.e., $x_i \in \mathbb{X}$, where $\mathbb{X} = X_1 \times \cdots \times X_d$ is called the instance space. Conversely, in MIL, each example is a multi-set (or colloquially a bag) $b_i \in \mathbb{N}^{\mathbb{X}}$ of instances with a single label $f(b_i)$. The bag b_i consists of n_i unlabeled instances $x_{ij} \in \mathbb{X}, j = 1, \ldots, n_i$. While in the traditional supervised learning the objective is to approximate the function $f : \mathbb{X} \to L$, in MIL the function to approximate is $f : \mathbb{N}^{\mathbb{X}} \to L$.

In this paper, we restrict ourselves to two-class classification, i.e., $L = \{0, 1\}$, where 0 stands for the negative class and 1 for the positive class.

2.2. Multi-instance learning assumptions

A key element in the design of a MIL algorithm is the hypothesis it assumes regarding the relationship between instances in a bag and bag-level class labels. The first MIL algorithms only considered the assumption that a bag is positive if and only if at least one of its instances is positive. This hypothesis was called *standard MIL assumption* by Weidmann et al. [28]. Under this assumption, the Boolean mapping function f_S is defined as $f_S(b) \iff \exists x \in b : c_I(x)$, where b is a bag and $c_I \in \mathbb{C}$ is a concept from a concept space $\mathbb{C}$. The standard MIL assumption has been applied to various problems such as drug activity prediction [8], image retrieval [3,19,36], and text categorization [3].

Weidmann et al. [28] also presented various generalizations to the standard MIL assumption. According to these, a *set* of underlying concepts $C \subset \mathbb{C}$ is used, each of which contributes to the classification. One of the generalizations is the *threshold based assumption*, in which the mapping function f_T requires a minimum number of instances of each concept; formally, $f_T(b) \iff \forall c_i \in C : \Delta(b, c_i) \geqslant t_i$, where $\Delta : \mathbb{N}^{\mathbb{X}} \times \mathbb{C} \to \mathbb{N}$ is a counting function which counts the members of a given concept in a bag, and t_i is the lower threshold of concept c_i. The standard MIL assumption can be seen as a particular case of the threshold based assumption where $C = \{c_I\}$ and $f_S(b) \iff \Delta(b, c_I) \geqslant 1$.

Other MIL assumptions include those based on bag-level distances [27], instance-level distances [24], the collective assumption [11], and more specific MIL assumptions. A good review of the most important MIL assumptions is presented in [12].

2.3. Categorization of multi-instance learning methods

Table 1 summarizes important MIL methods which have been developed to date, categorizing them according to their most distinctive features. Xu's taxonomy [29], placed in the columns of the table, divides MIL methods into instance-based approaches and metadata-based approaches. Instance-based approaches make the assumption that instances have hidden class labels. The set of hidden instance-level class labels may or may not correspond to the set of bag-level class labels. Methods in this category typically try to estimate a function that assigns class labels to instances, and then use that function to make a prediction at bag level. This category is further subdivided according to the specific MIL assumption used. The most basic division is into those algorithms implementing the standard MIL assumption and those implementing other assumptions. The importance of the standard MIL assumption is that it was the first approach to MIL and there are many algorithms that implement it.

Methods that use the metadata approach rely on the assumption that the class labels of bags are determined by some meta-level information that describes the examples. Algorithms of this type generally apply a transformation that maps bags into a new simple-instance feature space, where the features consist of some metadata extracted from the bag. A simple-instance learning algorithm can then be applied to the instances in this space in order to make predictions. Most metadata algorithms are data-oriented in the sense that the metadata is extracted directly from the data. The alternative to data orientation is the model-oriented approach, where some model is built on the data, and the metadata is extracted from this model rather than from the original data set.

Another way to classify the MIL methods is by the way they relate to traditional learning methods [11]. In this regard, there are three categories which we place in the rows of the table. First are those which are not related to traditional methods, they are *purpose-built* algorithms designed specifically to learn MIL concepts. The second category includes *upgraded* simple-instance learners that have been modified to learn directly from multi-instance data. Finally, there are *wrapper* algorithms that convert MIL problems into simple-instance problems, thereby allowing existing simple-instance learners to be applied directly.

In this work we present a novel instance-based wrapper that implements a variant of the threshold based assumption. Any simple-instance learner can be plugged into this wrapper. As a solution to the WIR problem we propose to combine the wrapper with the Rocchio classifier as this is a very efficient algorithm that performs well in textual domains.

3. Multi-instance learning approaches for web index recommendation

3.1. Web index recommendation

In 2005, Zhou et al. [37] presented the web index recommendation (WIR) problem as a MIL problem. From the standpoint of the MIL approach, in the WIR problem a bag is a web index page that we will call *index page* for short. In turn, instances of a bag are the web pages linked to the index page. We will call them *linked pages*.

Zhou et al. collected and prepared data, described in more detail in Section 5.1, that have become a benchmark to test new methods of solution to WIR. The benchmark data consist only of textual information; hypermedia information such as audio, images and videos was discarded to simplify the analysis. Moreover, the data

Table 1
Categorization of some representative MIL methods.

MIL Methods	Instance-based approaches		Metadata-based approaches	
	Standard MI assumption	Other assumptions	Data-oriented	Model-oriented
Purpose-built Algorithms	APR algorithms [8] Diverse density [17,18] EM-DD [35] ConMIL [22]		GMIL [24]	
Upgraded Simple-Instance Learners	mi-SVM [3] Decision trees [5,7] Decision rules [7] Boosting [4,14]	MI-SVM [3] Logistic regression [30] MIBoosting [30]	Citation-kNN [27] Minimax Kernel [15] MI Kernel [15] MI-Graph and mi-Graph [38]	
Wrappers for Simple-Instance Algorithms	Mi-NB [21]	MI-Wrapper [13]	Simple-MI [29] MILES [6] MICCLLR [10] BARTMIP [34]	Two-Level Classification [28] Constructive clustering Ensemble [39]

was preprocessed using feature selection, removing textual terms with little semantic content. In the benchmark data, trivial terms such as *a*, *the*, *is*, have been discarded.[1]

To solve the WIR problem, Zhou et al. [37] proposed the MIL algorithm Fretcit-kNN. Subsequently, new MIL algorithms from the genetic programming family were introduced in [32,33] to solve it. The following subsections describe these MIL approaches, along with the way in which each of them represents the WIR data.

3.2. Fretcit-kNN

Fretcit-kNN is based on the *Citation-kNN* [27] algorithm, which in turn is an adaptation of the popular kNN to the MIL approach. Instead of the traditional Euclidean metric, Citation-kNN uses a variant of the Hausdorff metric [9] to measure distance between bags. In order to increase robustness with respect to noise, Wang and Zucker [27] defined the minimal Hausdorff distance between bags $b = \{x_1, \ldots, x_{n_b}\}$ and $b' = \left\{y_1, \ldots, y_{n_{b'}}\right\}$ as

$$H_{min}(b, b') = \min_{x \in b, y \in b'} \|x - y\| \tag{1}$$

where $\|x - y\|$ is the distance under some norm between points x and y (usually Euclidean distance). To label a new bag, Citation-kNN takes into account the R-nearest *references* and the C-nearest *citers* to the bag. The reference concept is equivalent to the neighbor concept in classic kNN. Meanwhile, an example (a bag) b is a C-nearest citer of b', if b' is among the C nearest neighbors of b.

Just like classic kNN, Citation-kNN uses a vectorial representation of the instance space. Zhou et al. [37] adapted Citation-kNN to work with a different type of representation used in the data sets built for the WIR problem. They represent each instance (i.e., each linked page) as a term set $T = \{t_1, t_2, \ldots, t_n\}$, where t_i $(i = 1, \ldots, n)$ is one of the n most frequent terms appearing in the corresponding linked page, and they replace the usual Euclidean distance with a new distance measure that takes into account the number of shared terms between instances.[2] The distance between two instances x and y is defined as:

$$\textit{fret-distance}\ (x, y) = 1 - \sum_{\substack{i,j=1 \\ x_i = y_j}}^{n} \frac{1}{n}$$

where x_i represents the i-th term of instance x. Fretcit-kNN then refers to Citation-kNN where $\|x - y\| = \textit{fret-distance}\ (x, y)$.

[1] The list of used stop words is available from: http://lamda.nju.edu.cn/data_MILWEB.ashx.

[2] In this paper *textual terms* always refer to *single words*.

As Fretcit-kNN is a minor variation of Citation-kNN, it maintains the same computational complexity. As members of the lazy learning family, they have the drawback of deferring the main computational burden to the generalization stage. While its training time is $\mathcal{O}(N)$, where N is the number of training examples (bags), the generalization time is $\mathcal{O}\left(N^2 \log N\right)$ because they need to perform a sort operation ($\mathcal{O}(N \log N)$) for every training example in order to compute the C-nearest citer of the example to be classified.

Zhou et al. [37] report results of Fretcit-kNN on the benchmark data with term sets of size 5, 8, 10, 12 and 15. Fretcit-kNN was compared to a simplified version of the Rocchio classifier they call TFIDF [16], as well as with a classic kNN implementation and an updated kNN approach that considers both the references and the citers of an unseen object in prediction, like Citation-kNN but with Euclidean distance. These three algorithms were directly applied to the index pages while ignoring their linked pages. However, since most information of the index pages is delivered by their linked pages, their corresponding performances were very poor [37]. The classic and the updated kNN were therefore modified in order to consider the linked page information, by considering that all the instances in a bag have the label of that bag, and replacing Euclidean distance with *fret-distance* so as to enable the application to textual frequent terms. Fretcit-kNN was superior in all of these comparisons [37].

3.3. G3P-MI and MOG3P-MI

Zafra et al. [33] introduced a grammar-guided genetic programming approach to solve the WIR problem. By contrast to Zhou et al.'s approach, they use a vector whose components correspond to the terms in the document universe. By a document, they refer to the set of pages (instances) linked to an index page. They also apply a further feature selection technique, eliminating those terms appearing in less than 3 documents or more than 40.

Within their approach, they proposed two algorithms, *boolean G3P-MI* and *frequency G3P-MI*, whose differences are in the form used to represent instances. In boolean G3P-MI, each component of the vector is a boolean and indicates the presence of a term in the document, while in frequency G3P-MI, each component of the vector represents the absolute frequency of a term in the document.

Both methods use a context free grammar to evolve an evolutionary program in which individuals are rules that codify the conditions under which a bag is positive. The methods adopt the standard MIL assumption, according to which (considering that class labels 1 and 0 are equivalent to the logical values true and false) the label of a bag is obtained by the disjunction of the labels of its instances. In boolean G3P-MI the generated rules refer to the presence or absence of a term in a given page, while in frequency G3P-MI they indicate whether a term appears in a given page more or less times than a predefined threshold. The fitness function used in both cases tries to maximize the accuracy, seeking a balance between precision and recall by the formula $fitness = accuracy \times recall \times precision$.

However, the method was not able to obtain a good trade-off between recall and precision. Therefore, Zafra et al. [32] updated the former method and proposed MOG3P-MI, a multi-objective grammar-guided genetic programming method, based on SPEA2 [40], able to simultaneously optimize more than one performance measure, namely, sensitivity and specificity. Sensitivity (resp., specificity) measures the proportion of actual positives (resp., negatives) which are correctly classified. The experiments showed that MOG3P-MI indeed improves the classification performance [32].

Like SPEA2 [40], the complexity of MOG3P-MI in its training stage is on average $\mathcal{O}\left(N^2 \log N\right)$, with N equal to number of index pages. On the other hand, MOG3P-MI represents hypotheses as rules, so the generalization time is $\mathcal{O}(1)$.

4. Multi-instance learning wrapper based on the Rocchio classifier for web index recommendation

To solve the WIR problem, we present a new MIL classification method based on plugging a Rocchio classifier into a specific kind of MIL wrapper. The Rocchio classifier has been widely used in the field of information retrieval for its reasonable accuracy and low computational cost. The wrapper transforms the original MIL data into simple-instance data in a process called *propositionalization*, so that the Rocchio classifier can be applied. The integration between the Rocchio classifier and the proposed wrapper provides an efficient learning method well suited to the WIR problem. For convenience, from now on we will refer to this proposal as MI-Rocchio.

Section 4.1 describes the way in which we represent the WIR data, while Section 4.2 explains the simple-instance Rocchio classifier we use. The design of the new wrapper we introduce is presented in Section 4.3, and an evaluation of its run-time efficiency appears in Section 4.4.

4.1. Representation of web index recommendation data

Like Zafra et al.'s frequency GP3MI and MOG3P-MI, [32,33], we use a classical vector representation whose components are term frequencies, and apply the same feature selection preprocessing step. However, while their methods use absolute frequencies in the feature values, we calculate a weighted and normalized value. In particular, we use *tfidf* feature weighting (Term Frequency/Inverse Document Frequency) [25] by the formula

$$tfidf\ (t,x) = tf\ (t,x) \log\left(\frac{P}{df(t)}\right) \quad (2)$$

where $tf\ (t,x)$ is the number of times term t occurs in linked page x, P is the total number of linked pages in the training set, and $df(t)$ denotes the number of linked pages in the training set in which term t occurs. We applied cosine normalization using the formula:

$$w_{xk} = tfidf\ (t_k,x) \Big/ \sqrt{\sum_{i=1}^{d} (tfidf\ (t_i,x))^2} \quad (3)$$

where w_{xk} is the normalization weight of term t_k in linked page x, and d is the dimension of the attribute vector, i.e., the total number of remaining terms after the feature selection process.

4.2. Design of the Rocchio classifier for web index recommendation

The Rocchio algorithm is a two-class classifier which belongs to the family of profile-based classifiers. In profile-based classifiers, the classification function f is based on the similarity between the instance to be classified and the synthetic representation of each class, i.e., the class prototype or class profile. Instance information from both classes is aggregated to build each profile. Then, a new instance is assigned to the class whose profile is more similar.

In the WIR problem, we have the class label set $\mathcal{L} = \{+,-\}$, where $+$ means a *relevant* index page, and $-$ means an *irrelevant* index page. Following [20], the weights of attribute i in the positive profile $\overrightarrow{W^+}$ and the negative profile $\overrightarrow{W^-}$ are calculated by the Rocchio formula as

$$W_i^+ = \max\left\{0, \frac{1}{|R^+|}\sum_{x\in R^+} w_{xi} - \frac{\rho^+}{|R^-|}\sum_{x\in R^-} w_{xi}\right\} \quad (4)$$

$$W_i^- = \max\left\{0, \frac{1}{|R^-|}\sum_{x\in R^-} w_{xi} - \frac{\rho^-}{|R^+|}\sum_{x\in R^+} w_{xi}\right\} \quad (5)$$

where w_{xi} represents the weight of feature i in instance x, R^+ is the set of positive instances, and R^- is the set of negative instances. The Rocchio algorithm implicitly applies feature selection, and parameters ρ^+ and ρ^- control the extent of this selection [20]. The higher the parameter value, the more intense the feature selection.

From the class profiles $W^c, c \in \mathcal{L}$, the class label of a new example b is given by the following expression:

$$f(b) = \arg\max_{c\in\mathcal{L}} S_{\cos}(b, W^c) \quad (6)$$

where $S_{\cos}(x,y)$ represents the cosine similarity between vectors x and y, and is defined as

$$S_{\cos}(x,y) = \frac{xy}{|x||y|} \quad (7)$$

where $|x|$ is the vectorial norm of x.

4.3. Design of the proportion-based wrapper

The function of the wrapper is to interface between the MIL problem and the simple-instance classifier. Instance-based wrappers, like the one we propose, heuristically impute class labels to instances during the propositionalization step of the training stage. In the classification stage, the wrapper computes the class label of a new bag once its instances have been classified by the simple-instance classifier. Before we proceed to explain the details of the imputation scheme and the classification procedure, we first describe the MIL assumption that our wrapper implements.

4.3.1. Proportion-based MIL assumption

We introduce a new MIL assumption which is a variation of the threshold-based assumption [28] described in Section 2.2. We call it *proportion-based assumption* because it requires a minimum proportion (relative to the size of the bag) of instances of each concept. In general, let $C \subset \mathbb{C}$ be the set of underlying concepts used, and $\Delta : \mathbb{N}^{\mathbb{X}} \times \mathbb{C} \to \mathbb{N}$ a counting function which counts the members of a given concept in a bag. A mapping function f_P under the proportion-based assumptionis defined as $f_P(b) \iff \forall c_i \in C : \frac{\Delta(b,c_i)}{n_b} \geqslant t_i$, where t_i is the lower threshold of the proportion of concept c_i and n_b is the number of instances in bag b.

In the WIR problem, this assumption models the behavior of users who consider index pages relevant provided they have a certain minimum proportion of relevant links. In this case, we are only interested in the concept of relevant linked page, i.e., the positive class. Therefore, $C = \{+\}$ and the mapping function can be simplified as

$$f_P(b) \iff g(b) \geqslant t \tag{8}$$

where $g(b) = \frac{\Delta(b,+)}{n_b}$.

4.3.2. Algorithm description

The first step in the training stage is the propositionalization process. Since we have no a priori information about the class labels of instances in training bags, we rely on the standard MIL hypothesis "by default" to obtain an initial imputation: an instance is assigned to the negative class if it appears in any negative bag, otherwise it is assigned to the positive class.

Instances with imputed class labels are then used to feed the Rocchio formula in Eqs. (4) and (5) in order to compute the positive profile $\overrightarrow{W^+}$ and the negative profile $\overrightarrow{W^-}$. Next, the proportion threshold t is calculated based on the values of $g(b)$ for training bags b. In the computation of $g(b)$, a second imputation process is implicit. An instance x will be considered positive if its cosine similarity with the positive profile is greater than the cosine similarity with the negative profile. In the opposite case, it will be considered a negative instance. The calculation of the proportion threshold is described in the next subsection.

In the classification stage, Eq. (8) is used to determine the class label of a new bag. The bag b will be assigned to the positive class if $g(b) \geqslant t$, otherwise it will be assigned to the negative class; $g(b)$ is calculated in the same way as in the training stage.

The outline of the algorithm is shown graphically in Fig. 1. In summary, the proportion-basedwrapper first performs the process of instance label imputation, then constructs the class profiles by the Rocchio formula and finally determines the threshold for positive bags, using a second imputation. The class profiles together with the threshold constitute the learning model.

4.3.3. Assessing the proportion threshold t

Let μ_p and μ_n be the mean of $g(b)$ over all the positive, respectively negative bags in the training set. Also, let V_p and V_n be the variance of $g(b)$ over all the positive, respectively negative bags in the training set. We want the threshold t to adequately separate positive and negative bags. Given that positive and negative bags have different degrees of dispersion we want to find a middle ground between μ_p and μ_n to take into account the sample variance of each class. Therefore, one can use the average of the means weighted by the variances:

$$t_l = \frac{\mu_p V_n + \mu_n V_p}{V_p + V_n} \tag{9}$$

which we call the *linear weighting model*. This equation calculates a threshold t_l situated between μ_p and μ_n, which is located closest to either mean with the lowest variance. However, in some cases this threshold is too close to the sample mean as the population variance is greater than the sample variance. Specifically, when μ_n and V_n approach zero then t gets too close to zero and the classifier will not generalize well.

To alleviate this problem we calculate the average of the means weighted by the variances' exponentials, in what we call the *exponential weighting model*:

$$t_e = \frac{\mu_p e^{-\eta V_p} + \mu_n e^{-\eta V_n}}{e^{-\eta V_p} + e^{-\eta V_n}} \tag{10}$$

where η is a parameter that controls the slope of the curve. Fig. 2 shows how the threshold varies with each weighting model, the linear and the exponential, as the variance V_n moves from 0.1 to zero. Unlike the linear model, the exponential is a convex function, which slows the decrease of the threshold as the variance approaches zero.

4.4. Run-time efficiency

Our intention is not only to obtain a competitive algorithm in terms of classification accuracy, but also in terms of efficiency, as required for this type of online applications. For this reason, in this section we analyze the run-time complexity of our proposal and compare it to that of the algorithms Fretcit-kNN and MOG3P-MI.

As noted above, the simple-instance Rocchio classifier is very efficient; its algorithmic complexity in training is $\mathcal{O}(N)$, where N is the total number of training bags. Therefore, the efficiency of

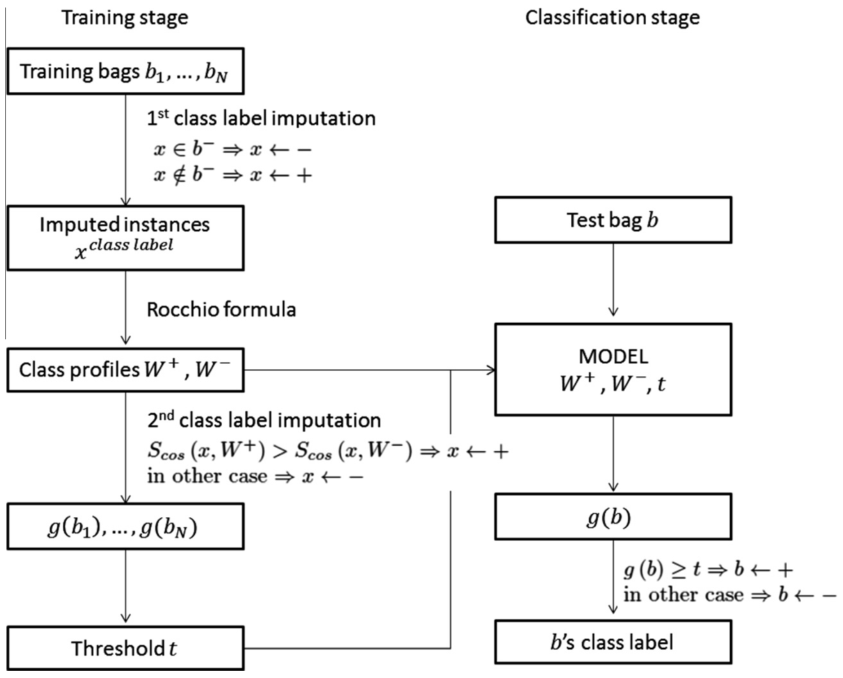
Training stage
Classification stage
Training bags $b_1, \ldots, b_N$
1st class label imputation
$x \in b^- \Rightarrow x \leftarrow -$
$x \notin b^- \Rightarrow x \leftarrow +$
Imputed instances $x^{class\ label}$
Rocchio formula
Class profiles W^+, W^-
2nd class label imputation
$S_{cos}(x, W^+) > S_{cos}(x, W^-) \Rightarrow x \leftarrow +$
in other case $\Rightarrow x \leftarrow -$
$g(b_1), \ldots, g(b_N)$
Threshold t
Test bag b
MODEL W^+, W^-, t
$g(b)$
$g(b) \geq t \Rightarrow b \leftarrow +$
in other case $\Rightarrow b \leftarrow -$
b's class label

Fig. 1. Overview of the MI-Rocchio algorithm; b^- is the set of all instances that appear in at least one negative training bag.

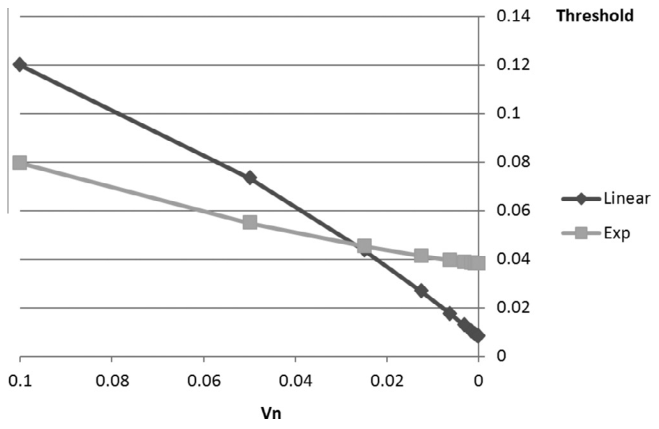

Fig. 2. Threshold value as a function of V_n according to linear and exponential weighting model. The other parameters are fixed at typical values of the WIR benchmark data: $\mu_p = 0.4$, $\mu_n = 0.008$ and $V_p = 0.25$. The value of η is set to 10.

the MI-Rocchio algorithm is determined by the complexity of the initial imputation method which is computationally more expensive because for each instance we need to check if it is in another bag, and the complexity of this step is $\mathcal{O}\left(N^2\right)$.

During classification, instances of the new bag just have to be compared with the class profiles, so the computational complexity of this part is $\mathcal{O}(1)$. Thus, the complexity analysis support the claim that MIRocchio is more efficient than the compared algorithms Fretcit-kNN and MOG3P-MI, which both have complexity $\mathcal{O}\left(N^2 log(N)\right)$.

5. Experimental results and analysis

In this section, we evaluate the performance of our approach and compare it with the state-of-the-art solutions to the WIR problem. Our experiments are aimed at three objectives:

- To compare the linear and the exponential weighting model introduced in Section 4.3.3 to establish the proportion threshold of MI-Rocchio, in order to determine the cases for which each of them are best suited, and to find the overall best variant.
- To verify the benefit of MI-Rocchio's use of the proportion-based MIL assumption introduced in Section 4.3.1 over the standard MIL assumption.
- To compare the MI-Rocchio variants with the state of the art algorithms proposed for the WIR problem, notably Fretcit-kNN and MOG3P-MI (see Section 3), in order to determine the relevance of our proposal.

For this purpose, we first discuss the data sets, performance measures and algorithm parameters used in the experimental study.

5.1. The web index recommendation benchmark data sets

For comparison purposes, we use the benchmark data.[3] introduced by Zhou et al. in [37]. This data consists of nine sets, as a result of 113 web index pages that were labeled by nine volunteers according to their interests. The label of the bag is positive if the web index page interested the user. Otherwise the label is negative. For each data set, 75 web index pages are used as training bags and the remaining 38 as test bags. These partitions were proposed by [37] and subsequently used by [31,32]. The number of positive and negative bags in the data sets is shown in Table 2.

Table 2
Class balance of the experimental data sets.

Data set	Training set		Test set	
	Positive	Negative	Positive	Negative
V1	17	58	4	34
V2	18	57	3	35
V3	14	61	7	31
V4	56	19	33	5
V5	62	13	27	11
V6	60	15	29	9
V7	39	36	16	22
V8	35	40	20	18
V9	37	38	18	20

Note that the balance between positive and negative classes varies considerably among the data sets; indeed, the appearance of unbalanced classes is a very common characteristic in recommendation problems. In particular, the first three volunteers have marked positive only about a third of the bags, the three following volunteers have marked positive about two-thirds of the bags while the last three volunteers have marked approximately equal numbers of positive and negative bags.

5.2. Performance measures

We evaluate the quality of the classification using a total of five performance measures: accuracy, precision, recall/sensitivity, specificity and Area Under the ROC Curve (AUC). The first three measures were proposed by [37] to analyze the behavior of the algorithm Fretcit-kNN, while the first, third and fourth were used by [32] to measure the performance of MOG3P-MI. We have added the AUC measure here to take into account the imbalance characteristic that occurs in 6 out of 9 data sets; indeed, a trivial classifier that assigns all instances to the majority class is able to reach high accuracy, but is not useful in practice. For instance, if all the examples of the test set corresponding to V2 were classified negative, $accuracy = 35/38 = 0.921$ which looks good. However, the classifier would have missed the three positive instances, so it would be useless to make recommendations. By contrast, AUC reflects the trade-off between correctly classified instances in the minority class and a high classification accuracy of instances in the majority class.

The definitions of these measures are based on the different outcomes of a two-class classification problem; a single prediction has the four different possible outcomes shown in Table 3. The measures themselves are defined according to Eqs. (11)–(15).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$Specificity = \frac{TN}{TN + FP} \tag{14}$$

$$AUC = \frac{1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}}{2} \tag{15}$$

5.3. Parameters of MI-Rocchio and comparison algorithms

The execution of MI-Rocchio requires setting the parameters ρ^+ and ρ^- in Eqs. (4) and (5) of the simple-instance Rocchio classifier. In our implementation, the values of these parameters are learned

[3] The data sets are available at http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/annex/milweb-datafile.htm.

Table 3
Different outcomes of a two-class prediction.

		Predicted class	
		Positive	Negative
Actual class	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

in the training stage; in particular, a total of 30 different configurations of ρ^+ ($\rho^+ = 15, 20, 25, 30, 40$) and ρ^- ($\rho^- = 2, 3, 4, 8, 15, 20$) are compared, using training accuracy to guide the search. As a general rule, the value of ρ^+ should be higher than that of ρ^-, for the following reason. According to the initial imputation of the proportion-based assumption, all instances in a negative bag are considered negative while only a certain percentage of the instances in a positive bag are considered positive, and the remainder negative; that is, positive bags involve more uncertainty than negative ones. Therefore, a more intense feature selection (greater ρ^+) in the positive instances benefits the classification accuracy.

On the other hand, we compare the linear weighting model for selecting the proportion threshold with the exponential one. The respective implementations are called MI-Rocchio-L and MI-Rocchio-E. The parameter η for t_e in Eq. (10) is set to 10.

As mentioned before, we also want to investigate the effect of using the proportion-based MIL assumption. To this aim, we compare MI-Rocchio-E and Mi-Rocchio-L with SAMI-Rocchio, a variant that uses the standard MIL assumption instead of the proportion-based assumption. Instead of basing the classification of a new bag on the proportion of positive instances in the bag (as MI-Rocchio does), SAMI-Rocchio assigns the bag to the positive class if at least one positive instance is present in the test bag. Therefore, there is no proportion threshold computation in the training stage, nor is a second imputation needed, and the learning model comprises just the class profiles.

We compare MI-Rocchio-E, Mi-Rocchio-L and SAMI-Rocchio with those algorithms specifically developed and applied to the WIR problem: Fretcit-kNN and MOG3P-MI. We do not compare with G3P-MI because from [32], it is clear that its results are lower than those of MOG3P-MI. These classifiers were described in Section 3. We use the results reported by their authors for the best configuration of each algorithm: 15 frequent terms to describe instances in Fretcit-kNN [37] and boolean features in MOG3P-MI [32].

5.4. Comparative analysis

Table 4 shows the accuracy, precision, recall/sensitivity, specificity and AUC of MI-Rocchio-L/MI-Rocchio-E and their rival classifiers over the nine test data sets V1–V9. The last column displays the performance measure average of each classifier. The best method is highlighted in bold for each data set.

Note that as there are only nine data sets, and ties frequently occur in the values of performance measures, it is not useful to apply statistical significance tests to these results. For the same reason, nor are nonparametric tests recommended by studies like [10,18] helpful. As an alternative way to compare the performance of classifiers, we may count the number of data sets for which an algorithm is the winner, which has been done in Tables 5–7.

5.4.1. Comparison between linear and exponential weighting

Comparing first the two proposed implementations of MI-Rocchio, it is clear that the version using an exponentially weighted proportion threshold performs much better than the one that assumes the linear weighting model. As stated also in Section 4.3.3, this has to do with the fact that when μ_n and V_n get too close to zero, the threshold t_l also becomes very small. This is especially so in the case of V4-V6, for which V_p is very high and the

Table 4
Comparative results.

Measure	Model	V1	V2	V3	V4	V5	V6	V7	V8	V9	Mean
Accuracy											
	MI-Rocchio-L	**0.921**	0.737	**0.868**	**0.947**	0.711	0.842	0.763	0.605	0.816	0.801
	MI-Rocchio-E	**0.921**	**0.947**	0.842	0.895	0.816	0.868	0.816	**0.737**	**0.921**	**0.863**
	SAMI-Rocchio	**0.921**	0.895	**0.868**	**0.947**	0.684	0.816	0.816	0.711	0.816	0.830
	Fretcit-kNN	**0.921**	0.868	**0.868**	0.895	**0.895**	**0.895**	0.816	**0.737**	0.632	0.836
	MOG3P-MI	**0.921**	0.921	**0.868**	0.921	0.842	0.842	**0.842**	0.711	0.763	0.848
Precision											
	MI-Rocchio-L	0.571	0.231	**0.667**	0.943	0.722	0.829	0.706	0.600	0.762	0.670
	MI-Rocchio-E	0.600	**0.600**	**0.667**	**1.000**	0.813	0.875	**0.909**	0.778	**0.941**	**0.798**
	SAMI-Rocchio	0.571	0.429	0.625	0.943	0.703	0.806	0.765	0.696	0.762	0.700
	Fretcit-kNN	0.600	0.333	**0.667**	0.968	**0.897**	**0.903**	0.800	**0.813**	0.667	0.739
	MOG3P-MI	**0.667**	0.500	**0.667**	0.917	0.862	0.829	0.778	0.737	0.737	0.744
Recall/sensitivity											
	MI-Rocchio-L	**1.000**	**1.000**	0.571	**1.000**	**0.963**	**1.000**	0.750	0.750	**0.889**	0.880
	MI-Rocchio-E	0.750	**1.000**	0.286	**1.000**	**0.963**	0.966	0.625	0.700	**0.889**	0.784
	SAMI-Rocchio	**1.000**	**1.000**	**0.714**	**1.000**	**0.963**	**1.000**	0.813	**0.800**	**0.889**	**0.909**
	Fretcit-kNN	0.750	0.667	0.571	0.909	**0.963**	0.966	0.750	0.650	0.444	0.741
	MOG3P-MI	0.500	0.667	0.571	**1.000**	0.926	**1.000**	**0.875**	0.700	0.778	0.780
Specificity											
	MI-Rocchio-L	0.912	0.714	0.935	0.600	0.091	0.333	0.773	0.444	0.750	0.617
	MI-Rocchio-E	0.941	**1.000**	**0.968**	**1.000**	0.455	0.556	**0.955**	0.778	**0.950**	**0.838**
	SAMI-Rocchio	0.912	0.886	0.903	0.600	0.000	0.222	0.818	0.611	0.750	0.634
	Fretcit-kNN	0.941	0.886	0.935	0.800	0.727	**0.667**	0.864	**0.833**	0.800	0.828
	MOG3P-MI	**0.971**	0.943	0.936	0.480	**0.765**	0.426	0.818	0.722	0.750	0.757
AUC											
	MI-Rocchio-L	**0.956**	0.857	0.753	0.800	0.527	0.667	0.761	0.597	0.819	0.749
	MI-Rocchio-E	0.846	**0.971**	0.627	**0.939**	0.709	0.761	0.790	0.739	**0.919**	**0.811**
	SAMI-Rocchio	**0.956**	0.943	**0.809**	0.800	0.481	0.611	0.815	0.706	0.819	0.771
	Fretcit-kNN	0.846	0.776	0.753	0.855	**0.845**	**0.816**	0.807	**0.742**	0.622	0.785
	MOG3P-MI	0.735	0.805	0.753	0.700	0.827	0.722	**0.847**	0.711	0.764	0.763

Table 5
Winners in MI-Rocchio-E vs. SAMI-Rocchio comparison over 9 data sets.

	Wins				
	Accuracy	Precision	Recall/ sensitivity	Specificity	AUC
MI-Rocchio-E	6	9	3	9	6
SAMI-Rocchio	3	0	6	0	3

Table 6
Winners in MI-Rocchio-E vs. Fretcit-kNN comparison over 9 data sets.

	Wins				
	Accuracy	Precision	Recall/sensitivity	Specificity	AUC
MI-Rocchio-E	4	5	4.5	5.5	3.5
Fretcit-kNN	5	4	4.5	3.5	5.5

Table 7
Winners in MI-Rocchio-E vs. MOG3P-MI comparison over 9 data sets.

	Wins				
	Accuracy	Precision	Recall/sensitivity	Specificity	AUC
MI-Rocchio-E	4.5	6.5	4.5	6.5	6
MOG3P-MI	4.5	2.5	4.5	2.5	3

proportion threshold will often be 0, not allowing to separate well between positive and negative bags.

However, for data sets V1–V3, in which the majority of examples are negative, the linear threshold performs quite well, and for V1 and V3 it even achieves the highest AUC value. This behavior can be explained in the following way. The preponderance of negative examples in data sets V1–V3 causes that a large number of instances in positive bags will be more similar to the negative profile than to the positive one, producing lower $g(b)$ values for positive bags. Hence, data sets V1–V3 are best fitted with lower thresholds such as those produced by the linear weighting model. Balanced data sets V7–V9, and data sets V4–V6, where the majority examples are positive, produce higher $g(b)$ values for positive bags, so they are best fitted with higher thresholds like that of the exponentially weighted model. This suggests that a hybrid model can be used for the selection of the proportion threshold, using the linear threshold when the majority examples are negative and the exponential threshold in the other case.

5.4.2. Comparison between proportion-based and standard MIL assumption

When we compare MI-Rocchio-E with SAMI-Rocchio, it is evident that the first algorithm, which uses the proportion-based assumption, performs much better than the variant relying on the standard assumption. Table 5 shows the winners between MI-Rocchio-E and SAMI- Rocchio for each performance measure.

MI-Rocchio-E is better than SAMI-Rocchio with respect to accuracy, precision, specificity and AUC. Only for recall, SAMI-Rocchio obtains more wins than MI-Rocchio-E. This can be explained by the different MIL assumptions they use. The set of test bags in which the proportion of positive instance is above a given threshold is included in the set of test bags containing at least one positive instance. In other words, the standard assumption is more general than the proportion-based assumption. Hence, SAMI-Rocchio assigns an example to the positive class more often than MI-Rocchio-E do. However, the lower precision values of SAMI-Rocchio reveal that this approach results in more false positives. Conversely, MI-Rocchio-E generates fewer false positives and more false negatives, which explains its high precision, high specificity, and moderate recall.

In the context of WIR, the standard assumption models the kind of user who considers an index page relevant if it has at least one relevant link. But the results shown here support the model of users who consider an index page relevant if it has a certain minimum proportion of relevant links, as implemented by the proportion-based assumption.

5.4.3. Comparison with state-of-the-art algorithms

When we compare MI-Rocchio-E with Fretcit-kNN and MOG3P-MI, it is clear that our method shows very good behavior, performing better, on average, for all the selected performance measures. Table 6 shows the winners between MI-Rocchio and Fretcit-kNN for each performance measure. In general, both algorithms have comparable performance, except for specificity for which MI-Rocchio-E obtains more wins, indicating that MI-Rocchio-E is better at avoiding false positives (irrelevant index pages recommended to the user) than Fretcit-kNN; and for AUC, where Fretcit-kNN is better in more cases, which appears to indicate that it is better at dealing with the data imbalance than MI-Rocchio-E. However, if only the data sets that are actually imbalanced (V1–V6) are taken into account, this difference is less pronounced, and if moreover for V1–V3 the linear proportion threshold is used, as was suggested in the previous paragraph, the advantage is actually for MI-Rocchio.

Table 7 shows the winners between MI-Rocchio-E and MOG3P-MI. In this case, the advantage of MI-Rocchio-E over MOG3P-MI regarding precision is very clear, meaning that the former makes much more accurate recommendations (more index pages recommended by MI-Rocchio-E are actually relevant). Similar observations hold for specificity and AUC. In terms of recall, MOG3P-MI and MI-Rocchio-E perform comparably, but the lower precision values of MOG3P-MI indicate that it has a certain predisposition to assign examples to the positive class, resulting in fewer false negatives but more false positives. This is also consistent with its modest specificity. MI-Rocchio-E, however, is more conservative for assigning an example to the positive class. Therefore it generates fewer false positives and more false negatives, which explains its high precision, high specificity, and moderate recall. This can be seen as an additional advantage of our method: in the WIR problem, due to the large number of web index pages on the Internet, the recommendation of an irrelevant page is more expensive than not to recommend a relevant page. In other words, the recommendation for a user of all potentially interesting pages (recall) is not as important as the recommendation of a really interesting page (precision).

6. Concluding remarks

In this paper, we have presented a new multi-instance learning algorithm to be applied to the web index recommendation problem. The algorithm is a wrapper that implements a new MIL assumption that we introduced with the name proportion-based assumption. This assumption assumes that a bag is positive if it contains a certain proportion of positive instances. The wrapper uses the simple-instance Rocchio algorithm as base classifier.

The proposed algorithm has low computational cost and the experimental study on benchmark data sets showed that it outperforms state-of-the-art algorithms applied to the problem, and that in particular its precision is high.

Since the Rocchio classifier is considered useful for text applications, the proposed algorithm may be advantageous in other applications such as text categorization. In addition, the new MIL assumption introduced and its implementation by the wrapper are general enough to be used with any base classifier to other application domains.

Acknowledgment

This work was partially supported by Projects TIN2011-28488 and P10-TIC-6858.

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans Knowl Data Eng 17 (6) (2005) 734–749.

[2] S. Andrews, Learning from Ambiguous Examples, Doctoral thesis, Brown University, Providence, Rhode Island, 2007.

[3] S. Andrews, I. Tsochantaridis, T. Hofmann, Support vector machines for multiple-instance learning, in: Advances in Neural Information Processing Systems 15 (NIPS), 2002, pp. 561–568.

[4] P. Auer, R. Ortner, A boosting approach to multiple instance learning, in: J.F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004, Lecture Notes in Computer Science, vol. 3201, Springer, Berlin/ Heidelberg, 2004, pp. 63–74.

[5] H. Blockeel, D. Page, A. Srinivasan. Multi-instance tree learning, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 57–64.

[6] Y. Chen, J. Bi, J.Z. Wang, MILES: multiple-instance learning via embedded instance selection, IEEE Trans. Pattern Anal. Mach. Intell. (2006) 1931–1947.

[7] Y. Chevaleyre, J.D. Zucker, Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. Application to the mutagenesis problem, in: Canadian Conference on AI, 2001, pp. 204–214.

[8] T.G. Dietterich, R.H. Lathrop, T. LozanoPerez, Solving the multiple instance problem with axis-parallel rectangles, Artif. Intell. 89 (1-2) (1997) 31–71.

[9] G.A. Edgar, Measure, Topology, and Fractal Geometry, third ed., Springer-Verlag, 1995.

[10] Y. El-Manzalawy, V. Honavar, MICCLLR: multiple-instance learning using class conditional log likelihood ratio, in: Discovery Science, 2009, pp. 80–91.

[11] J. Foulds, Learning Instance Weights in Multi-Instance Learning, Master Thesis, University of Waikato, Hamilton, New Zealand, February 2008.

[12] J. Foulds, E. Frank, A review of multi-instance learning assumptions, Knowl. Eng. Rev. 25 (1) (2010) 1–25.

[13] E. Frank, X. Xu, Applying Propositional Learning Algorithms to Multi-Instance Data, Technical report, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2003.

[14] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.

[15] T. Gärtner, P.A. Flach, A. Kowalczyk, A.J. Smola, Multi-instance kernels, in: Proc. 19th International Conf. on Machine Learning, Morgan Kaufmann, 2002, pp. 179–186.

[16] T. Joachims, A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, in: Proc. ICML1997: the Fourteenth Internat, Conf. on Machine Learning, 1997, pp. 143-151.

[17] O. Maron, Learning from Ambiguity. PhD thesis, Massachusetts Institute of Technology, United States, 1998.

[18] O. Maron, T. Lozano-Pérez, A framework for multiple-instance learning, in: Advances In Neural Information Processing Systems (NIPS), MIT Press, 1998, pp. 570–576.

[19] O. Maron, A.L. Ratan, Multiple-instance learning for natural scene classification, in: ICML, 1998, pp. 341–349.

[20] A. Moschitti, A study on optimal parameter tuning for Rocchio text classifier, in: ECIR, LNCS2633, 2003, pp. 420–435.

[21] J.F. Murray, G.F. Hughes, K. Kreutz, Machine learning methods for predicting failures in hard drives: A multiple-instance application, J. Mach. Learn. Res. 6 (2005) 816.

[22] G.J. Qi, X.S. Hua, Y. Rui, T. Mei, J. Tang, H.J. Zhang, Concurrent multiple instance learning for image categorization, in: Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[23] J. Rocchio, Relevance feedback in information retrieval, in: G. Salton (Ed.), The Smart Retrieval System-Experiments in Automatic Document Processing, Prentice Hall, 1971, pp. 313–323.

[24] S. Scott, J. Zhang, J. Brown, On generalized multiple-instance learning, Int. J. Comput. Intell. Appl. 5 (1) (2005) 21–35.

[25] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (2002) 1–47.

[26] A. Vinciarelli, Application of information retrieval techniques to single writer documents, Pattern Recognit. Lett. 26 (14) (2005) 2262–2271.

[27] J. Wang, J-D. Zucker, Solving the multiple-instance problem: a lazy learning approach, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 1119–1126.

[28] N. Weidmann, E. Frank, B. Pfahringer, A two-level learning method for generalized multi-instance problems, Mach. Learn.: Ecml 2003 2837 (2003) 468–479.

[29] X. Xu, Statistical Learning in Multiple Instance Problems, Master Thesis, University of Waikato, Hamilton, New Zealand, June 2003.

[30] X. Xu, E. Frank, Logistic regression and boosting for labeled bags of instances, in: Proc. of the PacificAsia Conf. on Knowledge Discovery and Data Mining, Springer-Verlag, 2004, pp. 272–281.

[31] A. Zafra, Modelos de Programación Genética Gramatical para Aprendizaje con Múltiples Instancias, PhD Thesis, Universidad de Granada, July 2009.

[32] A. Zafra, E.L. Gibaja, S. Ventura, Multiple instance learning with multiple objective genetic programming for web mining, Appl. Soft Comput. 11 (2011) 93–102.

[33] A. Zafra, C. Romero, S. Ventura, E. Herrera-Viedma, Multi-instance genetic programming for web index recommendation, Expert Syst. Appl. 36 (2009) 11470–11479.

[34] M.L. Zhang, Z.H. Zhou, Multi-instance clustering with applications to multi-instance prediction, Appl. Intell. 31 (1) (2009) 47–68.

[35] Q. Zhang, S.A. Goldman, EM-DD: an improved multiple-instance learning technique, Adv. Neural Inf. Process. Syst. 14, Vols. 1 and 2 14 (2002) 1073–1080.

[36] Q. Zhang, S.A. Goldman, W. Yu, J.E. Fritts, Content-based image retrieval using multiple-instance learning, in: Machine Learning-International Workshop Then Conference, 2002, pp. 682–689.

[37] Z.H. Zhou, K. Jiang, M. Li, Multi-instance learning based web mining, Appl. Intell. 22 (2005) 135–147.

[38] Z.H. Zhou, Y.-Y. Sun, Y.-F. Li, Multi-instance learning by treating instances as non-I.I.D. samples, in: Proceedings of the 26th International Conference on Machine Learning, ACM, 2009, pp. 1249–1256.

[39] Z.H. Zhou, M.L. Zhang, Solving multi-instance problems with classifier ensemble based on constructive clustering, Knowl. Inf. Syst. 11 (2) (2007) 155–170.

[40] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.