Expert Systems with Applications 36 (2009) 11470–11479

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Multi-instance genetic programming for web index recommendation

A. Zafra [a], C. Romero [a], S. Ventura [a,*], E. Herrera-Viedma [b]

[a] Dept. of Computer Sciences and Numerical Analysis, University of Córdoba, Campus de Rabanales, edificio Albert Einstein, 14071 Córdoba, Spain
[b] Dept. of Computer Sciences and Artificial Intelligence, University of Granada, Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

ARTICLE INFO

Keywords:
Grammar-guided genetic programming
Multiple instance learning
User modelling
Web mining

ABSTRACT

This article introduces the use of a multi-instance genetic programming algorithm for modelling user preferences in web index recommendation systems. The developed algorithm learns user interest by means of rules which add comprehensibility and clarity to the discovered models and increase the quality of the recommendations. This new model, called G3P-MI algorithm, is evaluated and compared with other available algorithms. Computational experiments show that our methodology achieves competitive results and provide high-quality user models which improve the accuracy of recommendations.

1. Introduction

In the last few years, the quantity of information available on Internet has been growing so rapidly that it now exceeds human processing capabilities. Users feel overwhelmed by the amount of information available and are usually unable to locate really relevant information that suits their individual needs in a limited amount of time. In this situation, there is a pressing need for tools that anticipate the preferences of users and provide recommendations about whether or not a particular item will be of interest to the user. Such systems, referred to in the literature as recommendation systems (Felfernig, Friedrich, & Schmidt-Thieme, 2007), have features similar to traditional information retrieval approaches but differ from them, especially in the use of models that contain information about user tastes, preferences and needs. This information differs according to the type of processing performed by the system. So, in collaborative filtering recommender systems (Schafer, Herlocker, & Sen, 2007) this model reflects similar users' preferences or needs, while in content-based recommender systems (Pazzani & Billsus, 2007) this information maps the relationship between the items to be recommended and the preferences of a given user.

In modelling user preferences, an interesting problem is the classifying of web index pages into two categories (according to whether or not they are pertinent for a user), because this allows us to build a user model for a content-based recommendation system. The main difficulty in this problem lies in training set representation; web index pages are those which contain references or brief summaries of other pages and where there is a different number of references on each page. Moreover, the information available about the user is imprecise. We know if the user is interested in an index page or not, instead of determining exactly which concrete links the user really considers to be of interest. Recently, Zhou, Jiang, and Li (2005) have solved the problem from a multi-instance learning perspective, adapting the well known k-Nearest Neighbor (k-NN) algorithm to this new learning framework. Experimental results show that this approach greatly improves supervised learning algorithm approaches.

In spite of the interesting results reported by Zhou et al. (2005), their proposal presents two major limitations. The first one is related to sparsity and to scalability, as the k-NN algorithm requires computations that grow linearly with the number of items, which makes it hard to scale when the number of items is high and maintain reasonable prediction performance and accuracy. The second one is related to the interpretability of new-found knowledge. The K-NN algorithm is a black box algorithm, that is, it simply classifies web index pages as being "of interest" or "not of interest", without providing additional information about user preferences. This is not a desirable property in recommendation systems, where any information that allows us to learn more about the interest of the user is of outmost interest for facilitating new recommendations.

To overcome the aforementioned drawbacks, we propose the use of G3P-MI, a grammar-guided genetic programming algorithm for multiple instance learning. This algorithm learns prediction rules which provide information on whether any of the links contained on a given web index page are of interest to a given user. Experimental results concerning several benchmarks show that this approach obtains competitive results in terms of accuracy, recall and precision. Moreover, it adds comprehensibility and clarity to the knowledge discovery process which is such an important characteristic for obtaining high predictive accuracy since the

* Corresponding author. Tel.: +34 957212218; fax: +34 957218630.
E-mail addresses: azafra@uco.es (A. Zafra), cromero@uco.es (C. Romero), sventura@uco.es (S. Ventura), viedma@decsai.ugr.es (E. Herrera-Viedma).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.03.059

system's results can be interpreted easily (understandable user models) and this data can be used to obtain further information about the user thus generating even more appropriate recommendations.

The rest of this paper is organized as follows. Section 2 is devoted to introducing the multi-instance learning paradigm, and Section 3 describes the proposed G3P-MI algorithm. Section 4 presents Web Index Recommendation as a multi-instance learning problem. Sections 5 and 6 presents and analyses the experimental results of our system. Finally, Section 6 presents conclusions and future work.

2. Multiple instance learning

The term Multiple Instance Learning was coined by Dietterich, Lathrop, and Lozano-Perez (1997) when investigating a qualitative structure–activity relationship problem. In this problem, the task consisted of determining if a given substance does or does not present pharmacological activity in information about its molecular structure. The difficulty of this task is due to the fact that a substance can present more than one spatial configuration, each of which showing different structural properties. Because conformations can most naturally be represented as fixed-length vectors of attribute values (or instances), the most convenient form for representing these learning examples seems to be in collections of these instances, with one associated class label representing the concept that we can learn. This is called "multiple instance" and contrasts with the manner of representing examples in supervised learning, where each example contains only one labeled instance.

To solve this last problem, Dietterich et al. (1997) proposed as the learning hypothesis that an example should be considered positive (that is, it represents the concept that we can learn) if it contains at least one instance that represents this concept. On the other hand, an example should be considered negative if it does not contain any instances representing the concept of learning. Using this learning hyphothesis, they developed three Axis-Parallel Rectangle (abbreviated as APR) algorithms, which attempt to search for appropriate axis-parallel rectangles constructed by the conjunction of their features. Their best performing algorithm (iterated-discrim) starts with a point in the feature space and "grows" a box with the goal of finding the smallest box that can cover at least one instance from each positive bag and no instances from any negative bags. The resulting box was then expanded (via a statistical technique) to get optimum results.

Following Dietterich et al.'s study, Auer (1997) tries to avoid some potentially difficult computational problems that were required by the heuristics used in the iterated-discrim algorithm, presenting a theoretical algorithm that does not require product distribution, MULTINST. With a new approach, Maron and Lozano-Pérez (1997) proposed one of the most famous multi-instance learning algorithms, Diverse Density (DD). The diverse density of a point, p, in the feature space is defined as a probabilistic measure taking into consideration how many different positive bags have an instance near p, and how far the negative instances are from p. This algorithm was combined with the Expectation Maximization (EM) algorithm, resulting in EM-DD (Zhang & Goldman, 2001), a general-purpose MI algorithm whose basic premise is to show which instance corresponds to the bag labeled as a missing attribute which can be estimated using the EM approach. Recently, Pao, Chuang, Xu, and Fu (2008) have proposed an EM based learning algorithm to provide a comprehensive procedure to maximize the measurement of DD on given multiple instances.

In 1998, Long and Tan (1998) described a polynomial-time theoretical algorithm showing that if the instances in the bags drawn from product distribution are independent, then the APR is PAC-learnable. Continuing with PAC-learnable research, Kalai and Blum (1998) described a reduction in the problem of PAC-learning in the MIL framework as compared to PAC-learning with one-sided random classification noise, and presented a theoretical algorithm with less complexity than the algorithm described in Auer (1997).

The first approaches using lazy learning, decision trees and rule learning were studied during the year 2000. In the lazy learning context, Wang and Zucker (2000) proposed two variants of the K-nearest neighbor algorithm (kNN) that they referred to as Citation-kNN and Bayesian-kNN, these algorithms extending the K-nearest neighbor algorithm for MIL, adopting Hausdorff distance. With respect to decision trees and learning rules, Zucker and Chevaleyre (2000) implemented ID3-MI and RIPPER-MI, which are multi-instance versions of decision tree algorithm ID3 and rule learning algorithm RIPPER, respectively. At that time, Ruffo (2000) presented a multi-instance version of the C4.5 decision tree, which was called RELIC.

There are also many other supervised learning algorithms which have been adapted to MIL. Thus, we can find the contribution of Ramon and De Raedt (2000) which extends standard neural networks to MIL. After this work, further studies appeared improving or extending it (Chai & Yang, 2007; Zhang, Jack, & Nandi, 2005; Zhang & Zhou, 2004, 2006). Another approach that has been adapted to the MIL framework is Support Vector Machines (SVM). There are numerous proposals in these approaches, Gärtner, Flach, Kowalczyk, and Smola (2002) adapted kernel methods to work with MIL data by modifying the kernel distance measures to handle sets. Using a related approach, Chen and Wang (2004) and Chen et al. (2006) adapted SVMs by modifying the form of the data rather than changing the underlying SVM algorithms while Andrews, Tsochantaridis, and Hofmann (2002) adapted the SVM kernels directly to produce one of the best MIL classification systems currently available. Recently, we can also find the proposals by Mangasarian and Wild (2008) and Gu et al. (2008). Finally, there are works such as those by Zhang et al. (2005) and Zhou and Zhang (2007) that show the use of ensembles to enhance multi-instance learners.

3. Grammar-guided genetic programming for multiple instance learning

In this section we introduce G3P-MI, a grammar-guided genetic programming algorithm for multi-instance learning. In the next sections, we will introduce the following design aspects: individual representation, genetic operators, fitness function and evolutionary process.

3.1. Individual representation

In G3P-MI, an individual represents rules that determine if a given pattern should be considered positive (that is, is an example of the concept we want to represent) or negative (if it is not):

if $(\text{cond}_B(b))$ **then**
  pattern b is an instance of the concept.
else (1)
  pattern b is not an instance of the concept.
end if

where cond_B is a condition that is applied to the pattern. Considering the Dietterich hypothesis, which states that a pattern is positive if any of its instances represent the concept that we want to learn and negative otherwise, we could represent cond_B as:

$$\mathrm{cond}_B(b) = \begin{cases} \textbf{true}, & \exists i \in [1, \mathrm{size}(b)] / \mathrm{cond}_I(\mathrm{instance}(i,b)) = \mathrm{true} \\ \textbf{false}, & \text{otherwise.} \end{cases} \quad (2)$$

where size(b) returns the number of instances in pattern b, instance(i, b) is a function that returns the ith instance to bag b and cond_I is a condition that is applied to an instance contained in a given bag. Considering the properties of the disjunction operator, Eq. (2) can be rewritten as

$$\mathrm{cond}_B(b) = \bigvee_{i=1,\mathrm{size}(b)} \mathrm{cond}_I(\mathrm{instance}(i,b)) \quad (3)$$

where $\vee$ is the disjunction operator. As can be seen, cond_I is the only variable part in Eqs. (1)–(3) that can experiment an evolutionary process. So, in G3P-MI, an individual's genotype is a syntax tree that contains the code of function cond_I, while the individual's phenotype is the whole rule that is applied to the bags (Eq. 1). Fig. 1 shows the context free grammar that represents these individuals in a general form. As can be seen, the code of the condition can contain one or several valid clauses, that check conditions related to instances contained in patterns. In the case of there being more than one clause, they can be combined by the conjunctions or disjunctions found in any order. The format of the clauses depends on the type of data contained in the instances analyzed (in Section 5 we will describe the format of the clauses used in the web index recommendation problem).

3.2. Initialization

To initialize the population in the algorithm, the procedure used is inspired by that defined by Geyer-Schulz (1995). This procedure builds a new syntax tree at random given the maximum number of derivations. To guarantee that the syntax tree generated is valid and uses a maximum number of derivations, the system calculates a selection probability for each symbol in the grammar for a specified number of available derivations. This table of probabilities, although implying some computational input, only has to be calculated once since it is saved with the rest of the structural information about individuals.

To guarantee greater diversity in the number of individuals generated, the initialization procedure generates individuals of different sizes, with two parameters as the minimum number of derivations and maximum number of individuals in the population.

3.3. Genetic operators

G3P-MI uses two genetic operators, called respectively selective crossover and selective mutation, to generate new individuals in a given generation of the evolutionary algorithm. Both operators were proposed by Whigham (1996) in the definition of grammar-based genetic programming. In this section, we will briefly describe their basic principles and functioning.

3.3.1. Selective crossover

This operator creates new programs by mixing the contents of two parent programs. To do so, a non-terminal symbol is chosen at random and two sub-trees (one from each parent) are selected whose roots coincide with the symbol adopted. Fig. 2 shows how this operation is performed.

$\langle S \rangle \rightarrow \langle \mathrm{cond}_I \rangle$
$\langle \mathrm{cond}_I \rangle \rightarrow \langle \mathrm{cmp} \rangle \mid \textbf{OR}\ \langle \mathrm{cmp} \rangle\ \langle \mathrm{cond}_I \rangle \mid \textbf{AND}\ \langle \mathrm{cmp} \rangle\ \langle \mathrm{cond}_I \rangle$
$\langle \mathrm{cmp} \rangle \rightarrow$ Any valid clause

Fig. 1. Grammar used for representing individuals' genotypes in G3P-MI.

Selective crossover presents several configuration parameters. On one hand, a list of eligible symbols can be defined in order to increase the probability of crossover for certain symbols and lessen that probability for certain others. On the other hand, in order to reduce bloating (Banzhaf, Francone, Keller, & Nordin, 1998), there is a parameter that defines the largest size possible for offspring generated in a crossover. Surpassing this size, the operations will reproduce the original parents which would also occur if one of the parents does not possess the symbol in question.

3.3.2. Selective mutation

Mutation is associated with a small change in the structure of the representation it is applied to. As can be seen in Fig. 3, this is achieved by randomly selecting a sub-tree within the individual to be mutated, and replacing this sub-tree with a new randomly-generated one. The procedure used to generate this sub-tree is the same as that used to create new individuals.

As in the case of selective crossover, this operator presents two configuration parameters: on one hand, the list of eligible non-terminal symbols as the root of the sub-tree to be mutated, and on the other, a maximum size for the offspring generated.

3.4. Evolutionary algorithm

G3P-MI follows the structure of a classical generational and elitist evolutionary algorithm. First of all, there is the creation of an initial population, following the procedure described in Section 3.2. Once the individuals are evaluated, we encounter the main loop of the algorithm composed of the following operations:

(1) *Parents selection*. The procedure followed to choose the individuals to be reproduced by crossover and/or mutation is roulette selection.
(2) *Parents reproduction*. Once the parents are obtained, the crossover operator is applied with a certain probability, and later on the mutation operator as well, also with a determined probability. The offspring obtained through this procedure are then evaluated.
(3) *Population update*. The population is updated by direct replacement, that is, the resulting offspring replace the present population. To guarantee that the best individual in the population is not lost during the updating process, the algorithm employs elitism.

Finally, there are two conditions for exiting from the main loop. On one hand, the algorithm ends if the maximum number of generations defined by the user is surpassed and, on the other, it also ends if the best individual in the population achieves the quality objectives indicated by the user.

4. Web index recommendation: a multiple instance problem

Web Index Pages are pages that provide titles or brief summaries of other pages. These pages contain a lot of information through references, leaving detailed presentations to their linked pages. An example of a web index page is http://health.yahoo.com as shown in Fig. 4.

The *web index recommendation* problem consists of building a model to establish exactly which web page index it is that interests a given user from among the contents of a myriad of web index pages that have already been labeled as being "of interest" or "not of interest" for this particular user. This problem is more difficult than other analogous ones related to the construction of user models for content-based recommendation systems (Mooney, Raymond, & Loriene, 2000; Pazzani & Billsus, 2007) since, in this case,

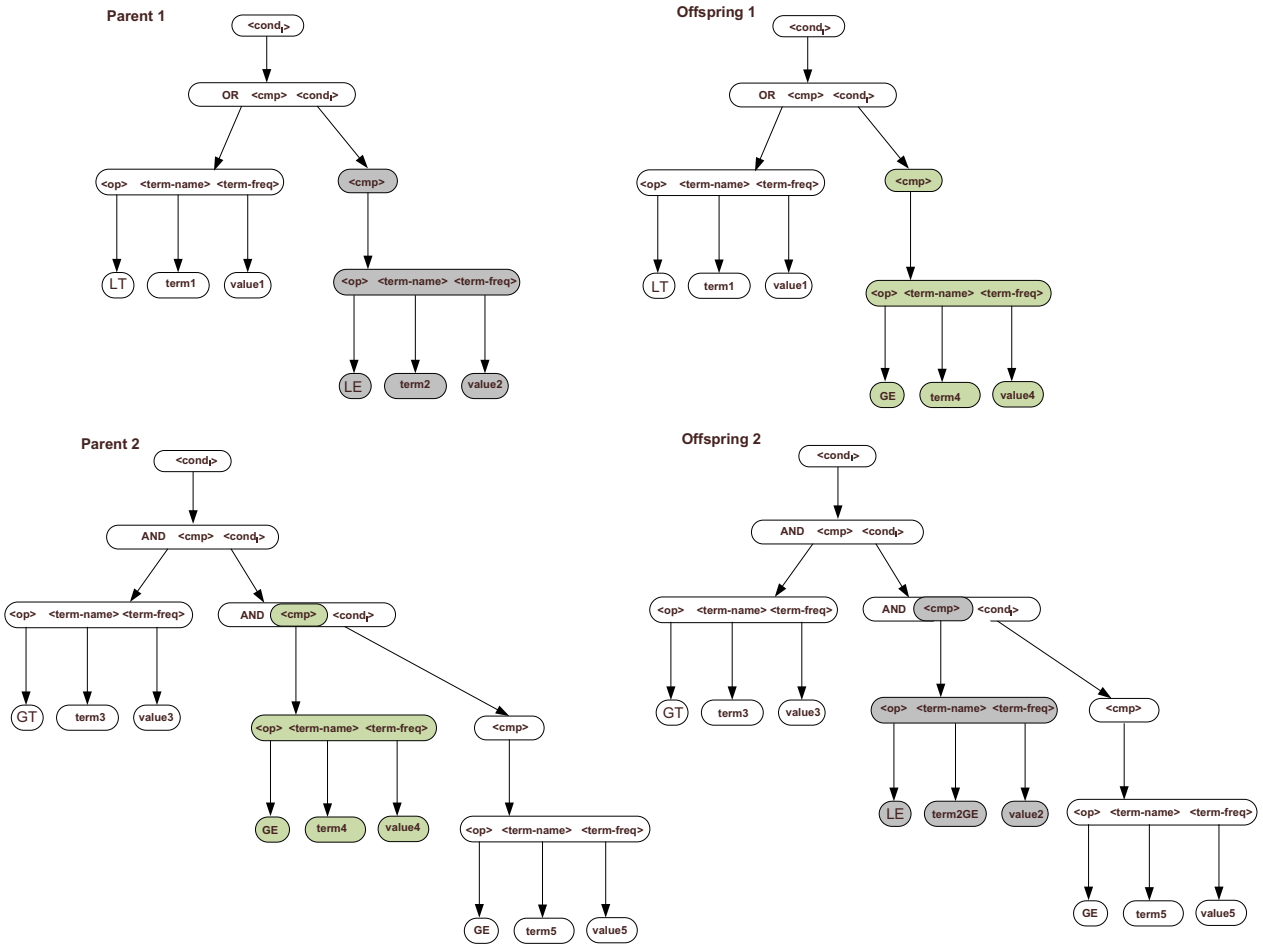

Fig. 2. Selective crossover.

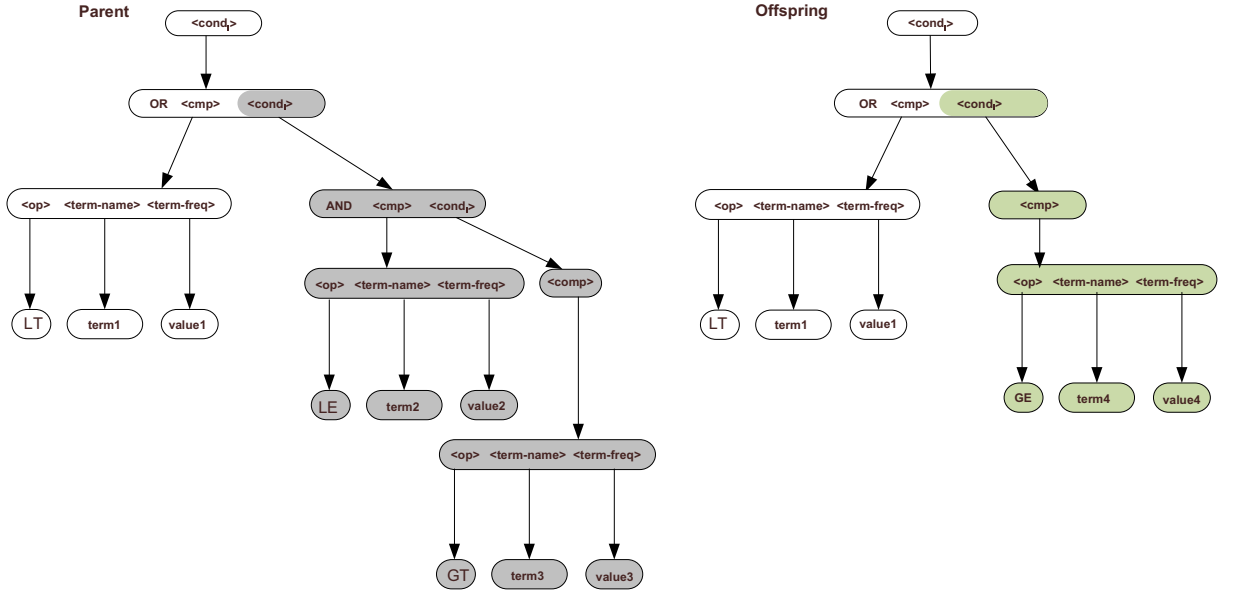

Fig. 3. Selective mutation.

the information on the user's preferences has to do with web index pages as a whole and not with the items contained on the page in question. For this reason, supervised learning algorithms could not produce results that met with expectations (Zhou et al., 2005).

On observing this problem in detail, it is seen to adjust perfectly to multi-instance representation. In fact, the web index page could be represented by a bag, and each of the connected pages is an example of a bag. Moreover, Dietterich's hypothesis also seems

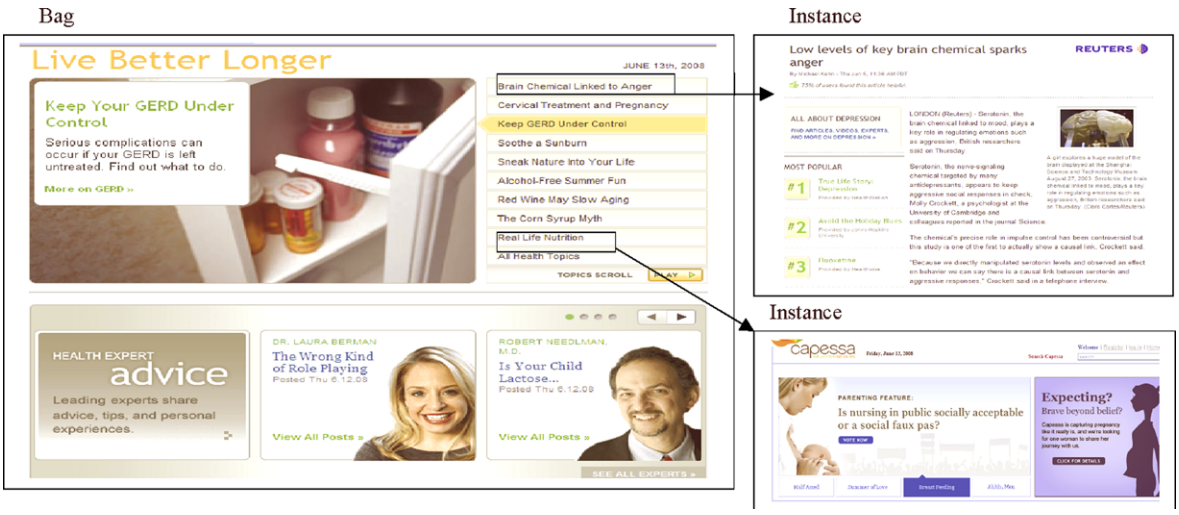

Fig. 4. Web index recommendation problem.

Table 1
Experimental data sets.

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

appropriate for resolving this problem. A web index page is considered to be of interest if at least one of the links on it leads to a page of interest for the user, while a page is not of interest if none of the pages indicated are useful for the user.

Based on previously mentioned considerations, we have considered the use of multi-instance representation for web index recommendation problems. Two different codification schemes normally used to categorize documents (Sebastiani, 2002) have been taken into consideration to build this type of instance. The first scheme is representing a page as a vector of Boolean values, each of which representing the presence (true) or absence (false) of a term in the document. The second scheme represents the page as a vector of integer values, each element represents the absolute frequency of the appearance of term on the page. As seen in Section 5, each of these coding schemes requires learning rules with different comparison operators and thus the result is two different versions of the G3P-MI algorithm.

5. Experimental setup

This section describes the data sets that have been used in the experimentation as well as several especially relevant methodological and configuration aspects.

5.1. Data sets

In order to evaluate our G3P-MI algorithm in the web index recommendation task, and to compare with other results previously published, we have used the data sets prepared by Prof. Z.H. Zhou's team. These datasets have been used in a previous paper about categorization of web index pages (Zhou et al., 2005). There are nine data sets and in each a different volunteer labeled 113 web index pages according to his/her interests. For each data set, 75 web index pages are randomly selected as training bags while the remaining 38 index pages are used as test bags. Table 1 summarizes the information about these data sets.[1]

To be able to use the original datasets, it has been necessary to preprocess information in order to adapt the original data to the format represented in our algorithms. In fact, although Zhou et al. represent web index pages as bags with one or various instances, they represent each instance by a list containing the N terms that most frequently appear on the page (where N takes values between 5 and 15). In our case, each instance is represented in vector format, where each component of the vector is related to a term in the corpus of available documents. The concrete value of each component can be binary (if we opt for Boolean representation) or whole if we opt for representation based on absolute frequencies.

Both these data sets are characterized by a high degree of dimensionality and great sparsity. In order to reduce both effects and probably also thus improve the yield of the algorithm, the original sets were preprocessed to eliminate the terms that appear only in a few documents and others that appear in most of them. In both cases, term discriminating capacity is so limited that their elimination does not reduce the information available. After a series of preliminary tests, it was decided to eliminate those terms that appeared in less than 3 documents or more than 40. In this way the total number of attributes is reduced from 5763 to 600.

In consequence, there are 4 different types of data sets:

Boolean all. These 9 data sets use Boolean representation (presence or absence of a term in a document), and each document vector presents a size of 5763 components. It is considered all terms belonging to the original corpus of documents.

[1] These datasets and a more detailed description of them can be downloaded from the following Internet address: http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/annex/milweb-datafile.htm.

Boolean filtered. These 9 data sets use Boolean representation (presence or absence of a term in the document), and each document vector is made up of 600 components. It is considered the terms from the corpus that appear in more than 3 and less than 40 documents.
Frequency all. These 9 data sets use numeric representation (frequency of appearance of a term in a document), and each document vector is composed of 5763 elements. It is considered all terms from the original corpus of documents.
Frequency filtered. These 9 data sets use numerical representation (frequency of appearance of a term in a document), and each document vector has 600 components. It is considered the terms that appear in more than 3 and less than 40 documents of the corpus.

As will be seen now, one of the objectives of this study is to evaluate which of the representation schemes (and associated versions of the G3P-Ml algorithm) produces the best results. Then, we will compare our best version with others algorithms which have solved the same problem.

5.2. Algorithm details

This section presents details about the adaptation of the G3P-MI algorithm for the task of recommendation of web index pages. First of all there are algorithm variants that have been defined to resolve this problem using, respectively, boolean and integer representations. The fitness function used in all the tests done is then explained. Finally there is a brief presentation of other configuration parameters of interest.

5.2.1. G3P-MI variants

As has already been mentioned, two variants of the G3P-MI algorithm have been specially developed for the two types of representations. Actually these variants only differ in the format of the prediction rules learned in the evolutionary process. Thus in the case of boolean representation, the rules generated refer to the presence or absence of a term on a given page, which respectively uses the functions "CONTAINS" and "DOES NOT CONTAIN", whose operator is the name of any of the terms that appear in the corpus of training documents (see Fig. 5).

When the representation of the pages is numerical (based on frequencies), the rules generated inform about whether a given term appears on a given page more or less frequently than a defined threshold. In this case the comparison operators "GT" (greater than), "GE" (greater than and equal to), "LT" (less than) and "LE" (less than and equal to) are used to compare the frequency of appearance of a term in the document with the threshold (an integer value). Fig. 6 shows the grammar that represents this type of rules.

5.2.2. Fitness function

The problem of developing good metrics to measure the effectiveness of recommendations is not a trivial task (Herlocker, Konstan, Borchers, & Riedl, 1999, 2004; Yang & Padmanabhan, 2005). To begin with, as it deals with a classification task, it would be logical to think that the use of such measures as *accuracy* (the percentage of web index pages that are correctly classified by the classifier) would be appropriate.

$\langle S\rangle \rightarrow \langle cond_I\rangle$
$\langle cond_I\rangle \rightarrow \langle cmp\rangle |$ **OR** $\langle cmp\rangle\ \langle cond_I\rangle |$ **AND** $\langle cmp\rangle\ \langle cond_I\rangle$
$\langle cmp\rangle \rightarrow \langle op\rangle\ \langle term\text{-}name\rangle$
$\langle op\rangle \rightarrow$ **CONTAINS** | **NOT CONTAINS**
$\langle term\text{-}name\rangle \rightarrow$ *Any valid term in dataset*

Fig. 5. Grammar used for learning from boolean datasets.

$\langle S\rangle \rightarrow \langle cond_I\rangle$
$\langle cond_I\rangle \rightarrow \langle cmp\rangle |$ **OR** $\langle cmp\rangle\ \langle cond_I\rangle |$ **AND** $\langle cmp\rangle\ \langle cond_I\rangle$
$\langle cmp\rangle \rightarrow \langle op\rangle\ \langle term\text{-}name\rangle\ \langle term\text{-}freq\rangle$
$\langle op\rangle \rightarrow$ **GT** | **GE** | **LT** | **LE**
$\langle term\text{-}name\rangle \rightarrow$ *Any valid term in dataset*
$\langle term\text{-}freq\rangle \rightarrow$ *Any integer value*

Fig. 6. Grammar used for learning from numerical (term frequency) datasets.

However, in recommender systems it is very common to work with unbalanced data sets, where the number of pages of interest is much lower than the number of pages that are not of interest. In this case, by solely using accuracy, the classifier obtains better results if no examples of the minority class (interesting examples) are classified than if only some are classified and mistakes are made with respect to the bigger class (uninteresting examples). For that reason, other measures are considered, such as *precision*, the percentage of accepted documents that are in fact relevant to a given user, that is

$$precision = \frac{\#interesting\ and\ recommended}{\#recommended} \quad (4)$$

In other cases, the problem is that there are many more interesting examples than uninteresting ones. In this case, precision is not the best measure to take into account, and it is necessary to use another measure such as *recall*

$$recall = \frac{\#interesting\ and\ recommended}{\#interesting} \quad (5)$$

For the above-mentioned reasons, we have tried to build a fitness function combining accuracy, precision and recall. Of all the combinations tested, the product has been that which produced the best results since it weighs the 3 measurements equally, as well as penalizing those individuals with a value of 0 in any of the above measurements. In consequence,

$$fitness = accuracy * recall * precision \quad (6)$$

has been the fitness function used in all the experiments.

5.2.3. Other algorithm parameters

Table 2 shows the configuration parameters used in the experiments carried out with all the versions of the G3P-MI algorithm. As can be seen, the only difference is the depth of the minimum depth used, which is less in the Boolean version than in the numerical one (perfectly logical taking into account that the number of substitutions needed to create a valid syntax tree is lower than in the Boolean version). All of the algorithms used have been developed in Java, using the Java Class Library for Evolutionary Computation (Ventura, Romero, Zafra, Delgado, & Hervás, 2008).

Table 2
Configuration parameters for all G3P-MI versions.

General	Population size = 1000 individuals Maximum of generations = 100
Initialization	Maximum tree depth = 15 Minimum tree depth = 5/6
Crossover	Crossover probability = 0.95 Eligible symbols = all non-terminals Maximum depth for sons = 15
Mutation	Mutation probability = 0.05 Eligible symbols = all non-terminals Maximum depth for sons = 15

6. Results and discussion

We carry out two types of experiments. The first experiment compares the performance of our proposals with respect to the problem of Web Index Recommendation. The second experiment compares the performance of our best algorithm to other classification techniques to solve this problem. This section describes these experiments and the results obtained. Also, at the end of the section we will comment on the type of knowledge discovered with G3P-MI algorithms.

6.1. Comparing different versions of G3P-MI

As mentioned previously, the objective of this first experiment was to analyze which variant of G3P-MI yields the best results in the web index recommendation task. To do so, we have executed every variant of the G3P-MI algorithm 10 times, with different random seeds, for the nine available data sets. Table 3 shows average values for accuracy, precision and recall.

To determine if there are significant differences between the results shown in Table 3, we have performed a Friedman test (Demsar, 2006). This is a non-parametric test that compares the average ranks of the algorithms, where the algorithm with the best results for a dataset is given a rank of 1 and the algorithm with the worst results is given a ranking of 4 (the ranking averages of all the available datasets are shown in Table 4). According to the Friedman test results, we accept the null-hypothesis, that is, with the available information, there do not exist significative differences between the different variants of G3P-MI analyzed.

In spite of the results yielded by the Friedman test, an inspection of Table 4 reveals that the versions that use the filtering operation (boolean filtered and frequency filtered) give slightly better results than their counterparts that do not have this preprocessing operation. This result is consistent with the fact that a reduction in search space facilitates the recommendation problem and achieves better results.

Another interesting question that can be deduced from the analysis of Table 4 is that, in general, boolean versions of G3P-MI seem to yield more precise results, while numerical versions of the algorithm are better in recall results. This seems to indicate that the algorithm has not been able to reach a complete trade-off between the metrics that are in conflict.

Table 4
Average rankings of the algorithms.

Algorithm	Ranking		
	Accuracy	Recall	Precision
Boolean all	2.6111	2.7777	2.7777
Boolean filtered	2.3888	3.3333	1.8888
Frequency all	2.8333	2.2777	2.8888
Frequency filtered	2.1666	1.6111	2.4444

6.2. Comparing G3P-MI with other proposals

In this section, we compare our algorithm with other proposals which have resolved this problem. These proposals are Frecit-kNN, Citation-kNN and Txt-kNN (Zhou et al., 2005). Txt-kNN is obtained through adapting the standard kNN algorithm to textual objects. Citation-kNN is similar to the latter algorithm but considers both the references and the citers of an unseen object in prediction. Frecit-kNN is similar to Citation-kNN, but is a multi-instance learning algorithm while the former is a single-instance learning algorithm. After analyzing our algorithm in the previous section, we choose the configuration which uses *filtered* data sets to compare these techniques. Table 5 shows a summary with the average values obtained by different algorithms using different data sets.

In order to make an empirical comparison between the different methods, we use Friedman test (Demsar, 2006) on the results reported in Table 5. The results of applying the Friedman test are reported in Table 6, for precision, recall and accuracy measurements. The test accepts the null-hypothesis for accuracy values and so we can determine that there are no significant differences for this measurement. For precision and recall measurements, the test rejects the null-hypothesis, indicating the existence of significant differences between the results of different algorithms. Due to these results, a posteriori statistical analysis is needed. In Fig. 7a and b, we show the application of the Bonferroni–Dunn test. We mark the interval of one critical difference (CD) to the left and right in Fig. 7. Any algorithm ranking outside this area is significantly different from the control algorithm (the control algorithm is that with the lowest ranking). We can see that with recall value, our proposal obtains the best value and we can assert that five algorithms are worse than our proposal because they rank outside the (CD) value. In the case of precision value, Frecit-kNN obtains the best value, and

Table 3
Results obtained with four different versions of G3P-MI (see text for details).

Algorithm	Accuracy								
	V1	V2	V3	V4	V5	V6	V7	V8	V9
Boolean all	0.8158	0.8684	0.8421	0.8684	0.7632	0.7895	0.7632	0.6842	0.6316
Boolean filtered	0.8421	0.7368	0.8421	0.8421	0.8421	0.7895	0.8421	0.6053	0.6842
Frequency all	0.9211	0.7368	0.8684	0.8947	0.7895	0.7632	0.4211	0.5263	0.4737
Frequency filtered	0.8684	0.9211	0.8684	0.8947	0.7632	0.8158	0.4737	0.5526	0.5526
	Precision								
	V1	V2	V3	V4	V5	V6	V7	V8	V9
Boolean all	0.2857	0.2500	0.5714	0.8684	0.7500	0.7838	0.6522	0.6429	0.5667
Boolean filtered	0.3333	0.2308	0.5714	0.9355	0.8889	0.8621	0.9167	0.5758	0.6875
Frequency all	1.0000	0.2308	0.7500	0.8919	0.7714	0.7632	0.4211	0.5263	0.4737
Frequency filtered	0.4286	0.5000	0.6250	0.8919	0.7500	0.8056	0.4444	0.5405	0.5143
	Recall								
	V1	V2	V3	V4	V5	V6	V7	V8	V9
Boolean all	0.5000	0.3333	0.5714	1.0000	1.0000	1.0000	0.9375	0.9000	0.9444
Boolean filtered	0.5000	1.0000	0.5714	0.8788	0.8889	0.8621	0.6875	0.9500	0.6111
Frequency all	0.2500	1.0000	0.4286	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Frequency filtered	0.7500	1.0000	0.7143	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 5
Results summary.

Algorithm	Accuracy								
	V1	V2	V3	V4	V5	V6	V7	V8	V9
Txt-kNN	0.8632	0.7948	0.7262	0.8894	0.7318	0.7946	0.5524	0.5896	0.5688
Citation-kNN	0.8788	0.7682	0.7632	0.8156	0.7578	0.8156	0.6842	0.6842	0.6528
Fretcit-kNN	0.9106	0.8682	0.8576	0.8474	0.8526	0.8632	0.7578	0.6898	0.6476
Boolean filtered	0.8421	0.7368	0.8421	0.8421	0.8421	0.7895	0.8421	0.6053	0.6842
Frequency Txt-kNN	0.8738	0.7422	0.7682	0.8948	0.7370	0.8054	0.6474	0.6002	0.5526
Frequency Citation-kNN	0.8526	0.8104	0.8368	0.8630	0.6686	0.8158	0.6792	0.7054	0.6370
Frequency Fretcit-kNN	0.9002	0.8366	0.8734	0.8264	0.8052	0.8000	0.7738	0.7160	0.7052
Frequency filtered	0.8684	0.9211	0.8684	0.8947	0.7632	0.8158	0.4737	0.5526	0.5526
	Precision								
	V1	V2	V3	V4	V5	V6	V7	V8	V9
Txt-kNN	0.3700	0.2710	0.3648	0.9046	0.7346	0.7886	0.4778	0.5974	0.5392
Citation-kNN	0.3998	0.1998	0.4088	0.9460	0.8302	0.8486	0.6132	0.7322	0.7582
Fretcit-kNN	0.5734	0.3476	0.6202	0.9538	0.8548	0.8792	0.7280	0.7388	0.6874
Boolean Filtered	0.3333	0.2308	0.5714	0.9355	0.8889	0.8621	0.9167	0.5758	0.6875
Frequency Txt-kNN	0.4238	0.1996	0.3542	0.9054	0.7612	0.7974	0.5564	0.6010	0.5290
Frequency Citation-kNN	0.3434	0.2762	0.6700	0.9214	0.8210	0.8316	0.6158	0.7316	0.7370
Frequency Fretcit-kNN	0.5000	0.6670	0.7140	0.9346	0.8096	0.8244	0.7224	0.7878	0.8946
Frequency filtered	0.4286	0.5000	0.6250	0.8919	0.7500	0.8056	0.4444	0.5405	0.5143
	Recall								
	V1	V2	V3	V4	V5	V6	V7	V8	V9
Txt-kNN	0.4500	0.9334	0.5714	0.9758	0.9778	1.0000	0.7128	0.7300	0.7222
Citation-kNN	0.3000	0.6002	0.4004	0.8362	0.8298	0.9244	0.6626	0.6300	0.3888
Fretcit-kNN	0.6000	0.6670	0.5996	0.8664	0.9556	0.9522	0.7002	0.6410	0.4666
Boolean Filtered	0.5000	1.0000	0.5714	0.8788	0.8889	0.8621	0.6875	0.9500	0.6111
Frequency Txt-kNN	0.5500	0.7332	0.3146	0.9820	0.9186	1.0000	0.8126	0.7100	0.7002
Frequency Citation-kNN	0.3500	0.7336	0.2860	0.9210	0.6818	0.9520	0.6504	0.7000	0.4332
Frequency Fretcit-kNN	0.5334	0.2826	0.6428	0.8604	0.9482	0.9382	0.7504	0.6310	0.4332
Frequency filtered	0.7500	1.0000	0.7143	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 6
Test Friedman ($p < 0.01$).

χ^2	Ranking		
	Accuracy	Recall	Precision
18.475	15.7314	28.5185	37.111

in this case, the Bonferroni–Dunn test does not consider there to be differences with our proposal, with $p = 0.05$.

Summing up, according to statistical tests, Fretcit-kNN is the control algorithm for recall measures, but our proposal is not significantly different according to the Bonferroni–Dunn test. On the other hand, our proposal for the precision measurement is considered the control algorithm and, in this case, the Bonferroni–Dunn test results show that both Fretcit-kNN versions are significantly different with respect to it. Therefore, from a statistical point of view, we can indicate that G3P-MI results are slightly better than Fretcit-kNN results. However, the most significant contribution of

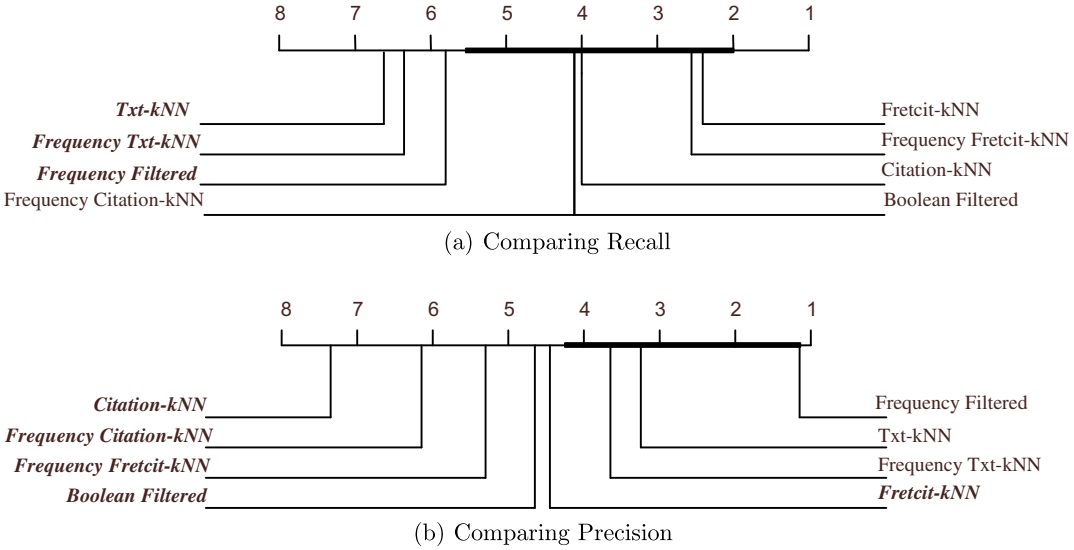

Fig. 7. Comparison of one classifier against the others with the Bonferroni–Dunn test, ($p < 0.05$).

our proposal is that it adds comprehensibility and clarity to the knowledge acquired. This point is dealt in more depth in the following section.

6.3. Knowledge acquired

As we noted in the introduction, the greatest advantage of the G3P-MI algorithm over other black-box proposals is the comprehensibility of the knowledge acquired, that is presented to the user in the form of prediction rules. In this section we introduce examples of these rules, and we discuss the advantages of using Boolean representation instead of frequency representation for comprehensibility criterion.

Firstly, we show a rule obtained for the first user/dataset using Boolean representation.

IF ((*contains* ***planet*** *AND contains* ***forecast*** *) OR*
(contains ***atmospheric***) *OR* (*not_contains* ***football***))
THEN *Recommend page to V1 user.*
ELSE *No recommend page to V1 user.*

By means of this rule we can learn what topics can be recommended to the user. Thus, user 1 is interested in such topics as ecology and environment (with words like planet, forecast and atmospheric) and is not interested in football.

Secondly, we show a rule obtained for the first user/dataset using numerical representation.

IF ((***frech*** > ***16***) ∨ (***house*** > *11)* ∨
(***science*** > *2)* ∨ (***aol*** ⩽ *20* ∧
on-line > *4)*))
THEN *Recommend page to V1 user.*
ELSE *No recommend page to V1 user.*

We can see that this rule is more complex because the words are limited by their frequency and it is more difficult to identify user preferences. For this, although both representations obtain similar results, after this study we can conclude that numerical representation is less effective because it obtains less comprehensive rules.

7. Conclusions and future work

This study describes the use of the G3P-MI algorithm for recommending Web Index Pages. This algorithm applies grammar-guided genetic programming to learn rules about whether or not a page referred to on a Web Index Page is of interest to a given user. To represent the Web Index Page, this algorithm applies the concept of multi-instances, representing the web pages as a set of instances where each instance represent the different referenced pages and stores information related to reference page. Two versions of the algorithms have been developed, one which is applied when pages are represented in Boolean form and the other which is applied when the representation format is based on the frequency of the appearance of certain terms on a page. There has also been analyzed the possibility of carrying out a previous filtering of information to eliminate less discriminating terminology. The experiments carried out show that although there is no significant differences in the application of a Friedman test, the techniques using a previous information filtering produce better results than those that do not while those that use numerical representation produce a less understandable knowledge.

Moreover, our proposal have been compared to the results of various versions of the kNN algorithm published by the Zhou team. The statistical tests carried out do not show significant differences in accuracy, although they do for precision and recall. In fact, our proposal is the one that produces the best results with respect to precision, while the Fretcit-kNN algorithm is that with the best results for recall. Finally, some examples of the rules discovered with the G3P-MI algorithm are presented. These rules show the high degree of comprehensibility of the knowledge acquired with the G3P-MI algorithm as compared to in black box methods like those based on kNN.

Although the results obtained are of great interest, we feel that the yield of the G3P-MI algorithm in the task of Web Index Page recommendation could be improved in some ways. On one hand the algorithm has not always been able to establish an equilibrium among conflicting objectives. In this sense the problem could be considered from a multi-objective perspective to see if the balance of different objectives could be achieved more appropriately. On the other hand it has been confirmed that a reduction in the space dedicated to characteristics can improve the yield of the algorithm. For this reason we would consider it to be of special interest to study the application of selection techniques for characteristics and compare the effect these would produce on the yield of our system.

Acknowledgments

This work has been subsidised in part by the research project SAINFOWEB (P05-TIC-00602) and the TIN2005-08386-C05-02, TIN2007-61079 and TIN2008-06681-C06-03 projects of the Spanish Inter-Ministerial Commission of Science and Technology (CICYT) and FEDER funds.

References

Andrews, S., Tsochantaridis, I., & Hofmann, T. (2002). Support vector machines for multiple-instance learning. In *NIPS'02: Proceedings of neural information processing system* (pp. 561–568).

Auer, P. (1997). On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *ICML'97: Proceedings of the 14th international conference on machine learning* (pp. 21–29). San Francisco, CA, USA: Morgan Kaufmann Publishers.

Banzhaf, W., Francone, F. D., Keller, R. E., & Nordin, P. (1998). *Genetic programming: An introduction. On the automatic evolution of computer programs and its applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Chai, Y.-M., & Yang, Z.-W. (2007). *A multi-instance learning algorithm based on normalized radial basis function network. LNCS* (Vol. 4491).

Chen, Y., Bi, J., & Wang, J. (2006). Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(12), 1931–1947.

Chen, Y., & Wang, J. Z. (2004). Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research, 5*, 913939.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence, 89*(1-2), 31–71.

Felfernig, A., Friedrich, G., & Schmidt-Thieme, L. (2007). Guest editors' introduction: Recommender systems. *IEEE Intelligent Systems, 22*(3), 18–21.

Gärtner, T., Flach, P. A., Kowalczyk, A., & Smola, A. J. (2002). Multi-instance kernels. In *ICML'02: Proceedings of the 19th international conference on machine learning* (pp. 179–186). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Geyer-Schulz, A. (1995). *Fuzzy rule-based expert systems and genetic machine learning* (1st ed.). Heidelberg, Germany: Physica Verlag.

Gu, Z., Mei, T., Tang, J., Wu, X., & Hua, X. (2008). *MILC2: A multi-layer multi-instance learning approach to video concept detection* (Vol. 4903).

Herlocker, J., Konstan, J., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *SIGIR'99: Proceedings of the 22nd annual international ACM conference on research and development in information retrieval, Berkeley, California, United States* (pp. 230–237).

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transaction Information Systems, 22*(1), 5–53.

Kalai, A., & Blum, A. (1998). A note on learning from multiple-instance examples. *Machine Learning, 30*(1), 23–30.

Long, P. M., & Tan, L. (1998). PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples. *Machine Learning, 30*(1), 7–21.

Mangasarian, O. L., & Wild, E. W. (2008). Multiple instance classification via successive linear programming. *Journal of Optimization Theory and Applications, 137*(3), 555–568.

Maron, O., & Lozano-Pérez, T. (1997). A framework for multiple-instance learning. *NIPS'97: Proceedings of neural information processing system* (Vol. 10, pp. 570–576). Cambridge, MA, USA: MIT Press.

Mooney, Raymond, J., & Loriene, R. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the ACM international conference on digital libraries* (pp. 195–204).

Pao, H. T., Chuang, S. C., Xu, Y. Y., & Fu, H. . (2008). An em based multiple instance learning method for image classification. *Expert Systems with Applications, 35*(3), 1468–1472.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: Methods and strategies of web Personalization. Lecture notes in computer science* (Vol. 4321, pp. 325–341). Springer.

Ramon, J., & De Raedt, L. (2000). Multi-instance neural networks. In *Proceedings of the ICML-workshop on attribute-value and relational learning.*

Ruffo, G. (2000). *Learning single and multiple instance decision tree for computer security applications.* Ph.D. thesis, Department of Computer Science. University of Turin, Torino, Italy.

Schafer, J. B., Herlocker, D. F. J., & Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: Methods and strategies of web personalization. Lecture notes in computer science* (Vol. 4321, pp. 291–324). Springer.

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

Ventura, S., Romero, C., Zafra, A., Delgado, J. A., & Hervás, C. (2008). JCLEC: A java framework for evolutionary computation soft computing. *Soft Computing, 12*(4), 381–392.

Wang, J., & Zucker, J.-D. (2000). Solving the multiple-instance problem: A lazy learning approach. In *ICML'00: Proceedings of the 17th international conference on machine learning* (pp. 1119–1126). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Whigham, P. A. (1996). *Grammatical bias for evolutionary learning.* Ph.D. thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia.

Yang, Y., & Padmanabhan, B. (2005). Evaluation of online personalization systems: A survey of evaluation schemes and a knowledge-based approach. *Journal of Electronic Commerce Research, 6*(2), 112–122.

Zhang, L., Jack, L. B., & Nandi, A. K. (2005). Fault detection using genetic programming. *Mechanical Systems and Signal Processing, 19*(2), 271–289.

Zhang, M.-L., & Zhou, Z.-H. (2004). Improve multi-instance neural networks through feature selection. *Neural Processing Letters, 19*(1), 1–10.

Zhang, M.-L., & Zhou, Z.-H. (2006). Adapting rbf neural networks to multi-instance learning. *Neural Processing Letters, 23*(1), 1–26.

Zhang, Q., & Goldman, S. (2001). EM-DD: An improved multiple-instance learning technique. In *NIPS'01: Proceedings of neural information processing system* (Vol. 14).

Zhou, Z.-H., Jiang, K., & Li, M. (2005). Multi-instance learning based web mining. *Applied Intelligence, 22*(2), 135–147.

Zhou, Z.-H., & Zhang, M.-L. (2007). Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowledge and Information Systems, 11*(2), 155–170.

Zucker, J.-D., & Chevaleyre, Y. (2000). Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem. In *Proceedings of the 14th Canadian conference on artificial intelligence. Lecture notes in artificial intelligence, Ottawa, Canada* (pp. 204–214).