Knowledge-Based Systems 83 (2015) 176–189

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Term-weighting learning via genetic programming for text classification

Hugo Jair Escalante [a,*], Mauricio A. García-Limón [a], Alicia Morales-Reyes [a], Mario Graff [b], Manuel Montes-y-Gómez [a], Eduardo F. Morales [a], José Martínez-Carranza [a]

[a] Computer Science Department, Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro 1, Puebla 72840, Mexico
[b] INFOTEC – Centro de Investigación e Innovación, en Tecnologías de la Información y Comunicación, Cátedras CONACyT, Aguascalientes, Mexico

A R T I C L E I N F O

Article history:
Received 16 December 2014
Received in revised form 16 March 2015
Accepted 19 March 2015
Available online 28 March 2015

Keywords:
Term-weighting learning
Genetic programming
Text mining
Representation learning
Bag of words

A B S T R A C T

This paper describes a novel approach to learning term-weighting schemes (TWSs) in the context of text classification. In text mining a TWS determines the way in which documents will be represented in a vector space model, before applying a classifier. Whereas acceptable performance has been obtained with standard TWSs (e.g., Boolean and term-frequency schemes), the definition of TWSs has been traditionally an art. Further, it is still a difficult task to determine what is the best TWS for a particular problem and it is not clear yet, whether better schemes, than those currently available, can be generated by combining known TWS. We propose in this article a genetic program that aims at learning effective TWSs that can improve the performance of current schemes in text classification. The genetic program learns how to combine a set of basic units to give rise to discriminative TWSs. We report an extensive experimental study comprising data sets from thematic and non-thematic text classification as well as from image classification. Our study shows the validity of the proposed method; in fact, we show that TWSs learned with the genetic program outperform traditional schemes and other TWSs proposed in recent works. Further, we show that TWSs learned from a specific domain can be effectively used for other tasks.

1. Introduction

Text classification (TC) is the task of associating documents with predefined categories that are related to their content. TC is an important and active research field because of the large number of digital documents available and the consequent need to organize them. The TC problem has been approached with pattern classification methods, where documents are represented as numerical vectors and standard classifiers (e.g., naïve Bayes and support vector machines) are applied [35]. This type of representation is known as the vector space model (VSM) [34]. Under the VSM one assumes a document is a point in a N-dimensional space and documents that are closer in that space are similar to each other [41]. Among the different instances of the VSM, perhaps the most used model is the bag-of-words (BOW) representation. In the BOW it is assumed that the content of a document can be determined by the (orderless) set of terms[1] it contains. Documents are represented as points in the vocabulary space, that is, a document is represented by a numerical vector of length equal to the number of different terms in the vocabulary (the set of all different terms in the document collection). The elements of the vector specify how important the corresponding terms are for describing the semantics or the content of the document. BOW is the most used document representation in both TC and information retrieval. In fact, the BOW representation has been successfully adopted for processing other media besides text, including, images [7], videos [37], speech signals [38], and time series [43] among others.

A crucial component of TC systems using the BOW representation is the so called term-weighting scheme (TWS), which is in charge of determining how relevant a term is for describing the content of a document [20,4,26,11]. Traditional TWSs are term-frequency (*TF*), where the importance of a term in a document is given by its frequency of occurrence in the document; Boolean (*B*), where the importance of a term in a document is either 1, when the term appears in the document or 0, when the term does not appear; and term-frequency inverse-document-frequency (*TF-IDF*), where the importance of a term for a document is determined by its occurrence frequency times the inverse frequency of the term across the corpus (i.e., frequent terms in the corpus, as prepositions and articles, receive a low weight). Although, TC is a widely studied topic with very important developments in the last two

* Corresponding author.
E-mail addresses: hugojair@inaoep.mx (H.J. Escalante), mauricio.garcia.cs@gmail.com (M.A. García-Limón), a.morales@inaoep.mx (A. Morales-Reyes), mgraffg@gmail.com (M. Graff), mmontesg@inaoep.mx (M. Montes-y-Gómez), emorales@inaoep.mx (E.F. Morales), carranza@inaoep.mx (J. Martínez-Carranza).

[1] A term is any basic unit by which documents are formed, for instance, terms could be words, phrases, and sequences (n-grams) of words or characters.

http://dx.doi.org/10.1016/j.knosys.2015.03.025
0950-7051/© 2015 Elsevier B.V. All rights reserved.

decades [35,20], it is somewhat surprising that little attention has been paid to the development of new TWSs to better represent the content of documents for TC. In fact, it is quite common in TC systems that researchers use one or two common TWSs (e.g., *B*, *TF* or *TF-IDF*) and put more effort in other processes, like feature selection [21,44], or the learning process itself [1,2,14]. Although all of the phases in TC are equally important, we think that by putting more emphasis on defining or learning effective TWSs we can achieve substantial improvements in TC performance.

This paper introduces a novel approach to learning TWS for TC tasks. A genetic program is proposed in which a set of primitives and basic TWSs are combined through arithmetic operators in order to generate alternative schemes that can improve the performance of a classifier. Genetic programming is an evolutionary algorithm in which a population of programs is evolved [27], where programs encode solutions to complex problems, in this work programs encode TWSs. The underlying hypothesis of our proposed method is that an evolutionary algorithm can learn TWSs of comparable or even better performance than those proposed so far in the literature.

Traditional TWSs combine term-importance and term-document-importance factors to generate TWSs. For instance in *TF-IDF*, *TF* and *IDF* are term-document-importance and term-importance factors, respectively. Term-document weights are referred as local factors, because they account for the occurrence of a term in a document (locally). On the other hand, term-relevance weights are considered global factors, as they account for the importance of a term across the corpus (globally). It is noteworthy that the actual factors that define a TWS and the combination strategy itself have been determined manually. Herein we explore the suitability of learning these TWSs automatically, by providing a genetic program with a pool of TWSs' building blocks with the goal of evolving a TWS that maximizes the classification performance for a TC classifier. We report experimental results in many TC collections that comprise both: thematic and non-thematic TC problems. Throughout extensive experimentation we show that the proposed approach is very competitive, learning very effective TWSs that outperform most of the schemes proposed so far. We evaluate the performance of the proposed approach under different settings and analyze the characteristics of the learned TWSs. Additionally, we evaluate the generalization capabilities of the learned TWSs and even show that a TWS learned from text can be used to effectively represent images under the BOW formulation.

The rest of this document is organized as follows. Next section formally introduces the TC task and describes common TWSs. Section 3 reviews related work on TWSs. Section 4 introduces the proposed method. Section 5 describes the experimental settings adopted in this work and reports results of experiments that aim at evaluating different aspects of the proposed approach. Section 6 presents the conclusions derived from this paper and outlines future research directions.

2. Text classification with the Bag of words

The most studied TC problem is the so called thematic TC (or simply text categorization) [35], which means that classes are associated to different themes or topics (e.g., classifying news into "*Sports*" vs. "*Politics*" categories). In this problem, the sole occurrence of certain terms may be enough to determine the topic of a document; for example, the occurrence of words/terms *"Basketball"*, *"Goal"*, *"Ball"*, and *"Football"* in a document is strong evidence that the document is about *"Sports"*. Of course, there are more complex scenarios for thematic TC, for example, distinguishing documents about sports news into the categories: *"Soccer"* vs. *"NFL"*. Non-thematic TC, on the other hand, deals with the problem of associating documents with labels that are not related to their topics. Non-thematic TC includes the problems of authorship attribution [39], opinion mining and sentiment analysis [32], authorship verification [25], author profiling [24], among several others [33,23]. In all of these problems, the thematic content is of no interest, nevertheless, it is common to adopt standard TWSs for representing documents in non-thematic TC as well (e.g., BOW using character n-grams or part-of-speech tags [39]).

It is noteworthy that the BOW representation has even trespassed the boundaries of the text media. Nowadays, images [7], videos [37], audio [38], and other types of data [43] are represented throughout analogies to the BOW. In non-textual data, a codebook is first defined/learned and then the straight BOW formulation is adopted. In image classification, for example, visual descriptors extracted from images are clustered and the centers of the clusters are considered as visual words [7,45]. Images are then represented by numerical vectors (i.e., a VSM) that indicate the relevance of visual words for representing the images. Interestingly, in other media than text (e.g., video, images) it is standard to use only the *TF* TWS, hence motivating the study on the effectiveness of alternative TWSs in non-textual tasks. Accordingly, in this work we also perform preliminary experiments on learning TWSs for a standard computer vision problem [19].

TC is a problem that has been approached mostly as a supervised learning task, where the goal is to learn a model capable of associating documents to categories [35,20,1]. Consider a data set of labeled documents $\mathcal{D} = (\mathbf{x}_i, y_i)_{\{1,\dots,N\}}$ with N pairs of documents ($\mathbf{x}_i$) and their classes (y_i) associated to a TC problem; where we assume $\mathbf{x}_i \in \mathbb{R}^p$ (i.e., a VSM) and $y_i \in C = \{1, \dots K\}$, for a problem with K-classes. The goal of TC is to learn a function $f : \mathbb{R}^p \rightarrow C$ from $\mathcal{D}$ that can be used to make predictions for documents with unknown labels, the so called test set: $\mathcal{T} = \{\mathbf{x}_1^T, \dots, \mathbf{x}_M^T\}$. Under the BOW formulation, the dimensionality of documents' representation, p, is defined as $p = |V|$, where V is the vocabulary (i.e., the set all the different terms/words that appear in a corpus). Hence, each document d_i is represented by a numerical vector $\mathbf{x}_i = \langle x_{i,1} \dots, x_{i,|V|} \rangle$, where each element $x_{i,j}, j = 1, \dots, |V|$, of $\mathbf{x}_i$ indicates how relevant word t_j is for describing the content of d_i, and where the value of $x_{i,j}$ is determined by the TWS.

Many TWSs have been proposed so far, including unsupervised [35,34,20] and supervised schemes [11,26], see Section 3. Unsupervised TWSs are the most used ones, they were firstly proposed for information retrieval tasks and latter adopted for TC [35,34]. Unsupervised schemes rely on term frequency statistics and measurements that do not take into account any label information. For instance, under the Boolean (B) scheme $x_{i,j} = 1$ *iff* term t_j appears in document i and 0 otherwise; while in the term-frequency (*TF*) scheme, $x_{i,j} = \#(d_i, t_j)$, where $\#(d_i, t_j)$ accounts for the times term t_j appears in document d_i. On the other hand, supervised TWSs aim at incorporating discriminative information into the representation of documents [11]. For example in the *TF-IG* scheme, $x_{i,j} = \#(d_i, t_j) \times IG(t_j)$, is the product of the TF TWS for term t_j and document d_i (a local factor) with the information gain of term t_j ($IG(t_j)$, global factor). In this way, the discrimination power of each term is taken into account for the document representation; in this case through the information gain value [44]. It is important to emphasize that most TWSs combine information from both term-importance (global) and term-document-importance (local) factors (see Section 3), for instance, in the *TF-IG* scheme, IG is a term-importance factor, whereas TF is a term-document-importance factor.

Although acceptable performance has been reported with existing TWS, it is still an art determining the adequate TWS for a particular data set; as a result, mostly unsupervised TWSs (e.g., *B*, *TF* and *TF-IDF*) have been adopted for TC systems [20,2]. A first hypothesis of this work is that different TWSs can achieve better performance on different TC tasks (e.g., thematic TC vs. non-thematic TC); in fact, we claim that within a same domain (e.g., news classification) different TWSs are required to obtain better classification performance on different data sets. On the other hand, we notice that TWSs have been defined as combinations of term-document weighting factors (which can be seen as other TWSs, e.g., *TF*) and term-relevance measurements (e.g., *IDF* or *IG*), where the definition of TWSs has been done by relying on the expertise of users. Our second hypothesis is that the definition of new TWSs can be automated. With the aim of verifying both hypotheses, this paper introduces a genetic program that learns how to combine term-document-importance and term-relevance factors to generate effective TWSs for diverse TC tasks.

3. Related work

As previously mentioned, in TC it is rather common to use unsupervised TWSs to represent documents, specifically *B, TF* and *TF-IDF* schemes are very popular (see Table 1). Their popularity derives from the fact that these schemes have proved to be very effective in information retrieval [34,5,41] and in many TC problems as well as [35,20,1–3]. Unsupervised TWSs mainly capture term-document occurrence (e.g., term occurrence frequency, *TF*) and term-relevance (e.g., inverse document frequency, *IDF*) information. While acceptable performance has been obtained with such TWSs in many applications, in TC one has available labeled documents, and hence, document-label information can also be exploited to obtain more discriminative TWSs. This observation was noticed by Debole & Sebastiani and other authors that have introduced supervised TWSs [11,26].

Supervised TWSs take advantage of labeled data by incorporating a discriminative term-weighting factor into the TWSs. In [11] TWSs were defined by combining the unsupervised TF scheme with the following term-relevance criteria: information gain (*TF-IG*), which measures the reduction of entropy when using a term as classifier [44]; χ^2 (*TF-CHI*), that makes an independence test regarding a term and the classes [35]; and gain-ratio (*TF-GR*) measuring the gain-ratio when using the term as classifier [11]. The conclusions from [11] were that small improvements can be obtained with supervised TWSs over unsupervised ones. Although somewhat disappointing, it is interesting that for some scenarios supervised TWSs were beneficial. More recently, Lan et al. proposed an alternative supervised TWS [26], the so called *TF-RF* scheme. *TF-RF* combines *TF* with a criterion that takes into account the true positives and true negative rates when using the occurrence of the term as classifier. In [26] the proposed *TF-RF* scheme obtained better performance than unsupervised TWSs and even outperformed the schemes proposed in [11]. In [4] the *RF* term-relevance factor was compared with alternative weights, including mutual information, odds ratio and χ^2; in that work *RF* outperformed the other term-importance criteria.

Table 1 shows the most common TWSs proposed so far for TC. It can be observed that TWSs are formed by combining term-document (*TDR*) and term (*TR*) relevance weights. The selection of what TDR and TR weights to use rely on researchers choices (and hence on their biases). It is quite common to use *TF* as *TDR*, because undoubtedly the term-occurrence frequency carries on very important information: we need a way to know what terms a document is associated with. However, it is not that clear what *TR* weight to use, as there is a wide variety of TR factors that have been proposed. The goal of *TRs* is to determine the importance of a given term, with respect to the documents in a corpus (in the unsupervised case) or to the classes of the problem (in the supervised case). Unsupervised *TRs* include: global term-frequency, and inverse document frequency (IDF) TRs. These weights can capture word importance depending on its global usage across a corpus, however, for TC seems more appealing to use discriminative TRs as one can take advantage of training labeled data. In this aspect, there is a wide variety of supervised TRs that have been proposed, including: mutual information, information gain, odds ratio, etcetera [3].

The goal of a supervised TR weight is to determine the importance of a given term with respect to the classes. The simplest, TR would be to estimate the correlation of term frequencies and the classes, although any other criterion that accounts for the association of terms and classes can be helpful as well. It is interesting that although many TRs are available out there, they have been mostly used for feature selection rather than for building TWSs for TC. Comprehensive and extensive comparative studies using supervised TRs for feature selection have been reported [4,21,44,30]. Although not being conclusive, these studies serve to identify the most effective TRs weights, such weights are considered in this study.

To the best of our knowledge, the way we approach the problem of learning TWSs for TC is novel. Similar approaches based on genetic programming to learn TWSs have been proposed in [9,10,8,40,31,17], however, these researchers have focused on the information retrieval problem, which differs significantly from TC. Early approaches using genetic programming to improve the *TF-IDF* scheme for information retrieval include those from [40,31,17,18]. More recently, Cummins et al. proposed improved genetic programs to learn TWSs also for information retrieval [9,10,8].

Although the work by Cummins et al. is very related to ours, there are major differences (besides the problem being approached): Cummins et al. approached the information retrieval task and defined a TWS as a combination of three factors: local, global weighting schemes and a normalization factor.[2] The authors designed a genetic program that aimed at learning a TWS by evolving the local and global schemes separately. Only 11 terminals, including constants, were considered. Since information retrieval is an unsupervised task, the authors had to use a whole corpus with relevance judgements (i.e., a collection of documents with queries and the set of relevant documents to each query) to learn the TWS, which, once learned, could be used for other information retrieval tasks. Hence they require a whole collection of documents to learn a TWS. On the other hand, the authors learned a TWS separately, first a global TWS was evolved fixing a binary local scheme, then a local scheme was learned by fixing the learned global weight. Hence, they restrict the search space for the genetic program, which may limit the TWSs that can be obtained. Also, it is worth noticing that the focus of the authors of [9,10,8] was on learning a single, and generic TWS to be used for other information retrieval problems, hence the authors performed many experiments and reported the single best solution they found after extensive experimentation. Herein, we provide an extensive evaluation of the proposed approach, reporting average performance over many runs and many data sets. Finally, one should note that the approach from [9,10,8] required of large populations and numbers of generations (1000 individuals and 500 generations were used), whereas in this work competitive performance is obtained with only 50 individuals and 50 generations.

[2] Recall a local factor incorporates term information (locally) available in a document, whereas a global term factor takes into account term statistics estimated across the whole corpus. In information retrieval it is also common to normalize the vectors representing a document to reduce the impact of the length of a document.

Table 1
Common term weighting schemes for TC. In every TWS, $x_{i,j}$ indicates how relevant term t_j is for describing the content of document d_i, under the corresponding TWS. N is the number of documents in training data set, $\#(d_i, t_j)$ indicates the frequency of term t_j in document d_i, $df(t_j)$ is the number of documents in which term t_j occurs, $IG(t_j)$ is the information gain of term t_j, $CHI(t_j)$ is the χ^2 statistic for term t_j, and TP, TN are the true positive and true negative rates for term t_j (i.e., the number of positive, resp. negative, documents that contain term t_j).

Acronym	Name	Formula	Description	Ref.
B	Boolean	$x_{i,j} = \mathbf{1}_{\{\#(t_i,d_j)>0\}}$	Indicates the prescense/abscense of terms	[34]
TF	Term-Frequency	$x_{i,j} = \#(t_i, d_j)$	Accounts for the frequency of occurrence of terms	[34]
TF-IDF	TF – Inverse Document Frequency	$x_{i,j} = \#(t_i, d_j) \times \log(\frac{N}{df(t_j)})$	An TF scheme that penalizes the frequency of terms across the collection	[34]
TF-IG	TF – Information Gain	$x_{i,j} = \#(t_i, d_j) \times IG(t_j)$	TF scheme that weights term occurrence by its information gain across the corpus	[11]
TF-CHI	TF – Chi-square	$x_{i,j} = \#(t_i, d_j) \times CHI(t_j)$	TF scheme that weights term occurrence by its χ^2 statistic	[11]
TF-RF	TF – Relevance Frequency	$x_{i,j} = \#(t_i, d_j) \times \log(2 + \frac{TP}{\max(1,TN)})$	TF scheme that weights term occurrence by its *RF* relevance	[26]

4. Learning term-weighting schemes via GP

As previously mentioned, the traditional approach for defining TWSs has been somewhat successful so far. Nevertheless, it is still unknown whether we can automatize the TWS definition process and obtain TWSs of better classification performance in TC tasks. In this context, we propose a genetic programming solution that aims at learning effective TWSs automatically. We provide the genetic program with a pool of TDR and TR weights as well as other TWSs and let a program search for the TWS that maximizes an estimate of classification performance. Thus, instead of defining TWSs based on our own experiences on text mining, we let a computer itself to build an effective TWS. The advantages of this approach are that, on the one hand, it may allow to learn a specific TWS for each TC problem, or, on the other hand, to learn TWSs from one data set (e.g., a small one) and implement it in a different collection (e.g., a huge one). Furthermore, the method reduces the dependency on users/data-analysts and their degree of expertise and biases for defining TWSs. The rest of this section describes the proposed approach. We start by providing a brief overview of genetic programming, then we explain in detail the proposal, finally, we close this section with a discussion on the benefits and limitations of our approach.

4.1. Genetic programming

Genetic programming (GP) [27] is an evolutionary technique which follows the reproductive cycle of other evolutionary algorithms such as genetic algorithms (see Fig. 1): an initial population is created (randomly or by a pre-defined criterion), after that, individuals are selected, recombined, mutated and then placed back into the solutions pool. The distinctive feature of GP, when compared to other evolutionary algorithms, is in that complex data structures are used to represent solutions (individuals), for example, trees or graphs. As a result, GP can be used for solving complex learning/modeling problems.

In our case, GP automatically determines what primitives to combine and what operators to apply for combining them. This would not be possible to do with other bio-inspired algorithms or meta-heuristics (e.g., genetic algorithms), were we would have to assume a fixed way to combine primitives (e.g., by addition or product of primitives) in that case, the optimization strategy would select what primitives to use, but not what operators to use (e.g., power, square-root, product, division, logarithm, etc.). In the following we describe the GP approach to learn TWSs for TC.

4.2. TWS learning with genetic programming

We face the problem of learning TWSs as an optimization one, in which we want to find a TWSs that maximizes the classification

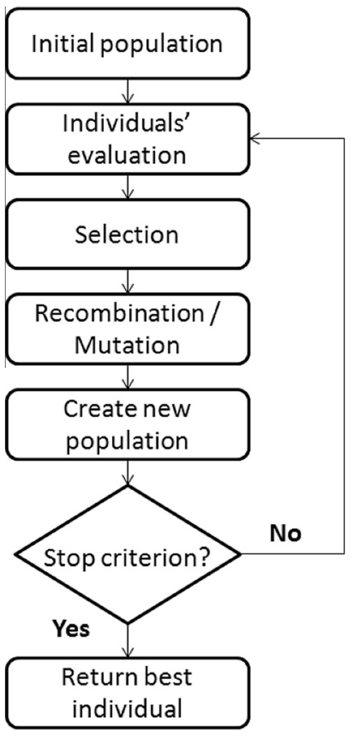

Fig. 1. A generic diagram of an evolutionary algorithm.

performance of a classifier trained with the TWS. We define a valid TWS as the combination of: (1) other TWSs, (2) TR and (3) TDR factors, and restrict the way in which such components can be combined by a set of arithmetic operators. We use GP as optimization strategy, where each individual corresponds to a tree-encoded TWS. The proposed genetic program explores the search space of TWSs that can be generated by combining TWSs, TRs and TDRs with a predefined set of operators. The remainder of this section details the components of the proposed genetic program: representation, terminals, function set, operators and fitness function; Table 2 summarizes its main parameters.

4.2.1. Representation

Solutions to our problem are encoded as trees, where we define terminal nodes to be the building blocks of TWSs. On the other hand, we let internal nodes of trees to be instantiated by arithmetic operators that combine the building blocks to generate new TWSs. The representation is graphically described in Fig. 2.

Table 2
GP parameters.

Parameter	Value
Population size	50
Number of generations	50
Terminals	see Table 1
Function set ($\mathcal{F}$)	$\{+, -, *, /, \log_2 x, \sqrt{x}, x^2\}$
Crossover rate	90%
Mutation rate	10%
Maximum init. depth	6
Maximum depth search	17
Tournament	Lexictour [28]

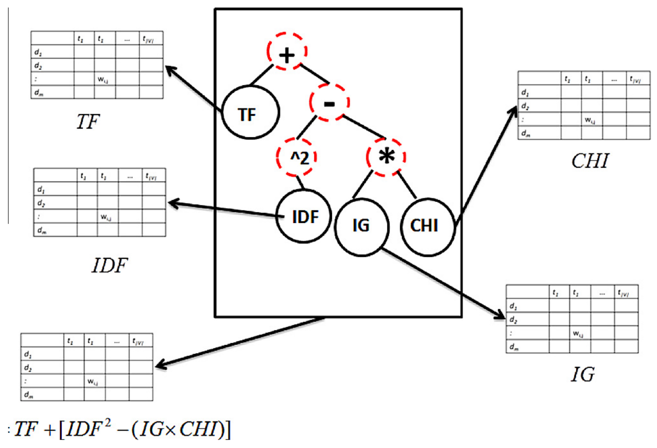

Fig. 2. Representation adopted for TWS learning.

Table 3
Terminal set.

Variable	Meaning
W_1	*N*, Constant matrix, the total number of training documents
W_2	$\|V\|$, Constant matrix, the number of terms
W_3	*CHI*, Matrix containing in each row the vector of χ^2 weights for the terms
W_4	*IG*, Matrix containing in each row the vector of information gain weights for the terms
W_5	*TF – IDF*, Matrix with the TF-IDF term weighting scheme
W_6	*TF*, Matrix containing the TF term-weighting scheme
W_7	*FGT*, Matrix containing in each row the global term-frequency for all terms
W_8	*TP*, Matrix containing in each row the vector of true positives for all terms
W_9	*FP*, Matrix containing in each row the vector of false positives
W_{10}	*TN*, Matrix containing in each row the vector of true negatives
W_{11}	*FN*, Matrix containing in each row the vector of false negatives
W_{12}	*Accuracy*, Matrix in which each row contains the accuracy obtained when using the term as classifier
W_{13}	*Accuracy_Balance*, Matrix containing the AC_Balance each (term, class)
W_{14}	*BNS*, An array that contains the value for each BNS per (term, class)
W_{15}	*DFreq*, Document frequency matrix containing the value for each (term, class)
W_{16}	*FMeasure*, F-Measure matrix containing the value for each (term, class)
W_{17}	*OddsRatio*, An array containing the OddsRatio term-weighting
W_{18}	*Power*, Matrix containing the Power value for each (term, class)
W_{19}	*ProbabilityRatio*, Matrix containing the ProbabilityRatio each (term, class)
W_{20}	*Max_Term*, Matrix containing the vector with the highest repetition for each term
W_{21}	*RF*, Matrix containing the RF vector
W_{22}	*TF × RF*, Matrix containing *TF × RF*

4.2.2. Terminals and function set

As previously mentioned, traditional TWSs are usually formed by two factors: a term-document relevance (*TDR*) weight and a term-relevance (*TR*) factor. The most used *TDR* is term frequency (*TF*), as allows one to relate documents with the vocabulary. We consider *TF* as *TDR* indicator, but also we consider standard TWSs (e.g., *Boolean, TD, RF*) as *TDR* weights. The decision to include other TWSs as building blocks is in order to determine whether standard TWSs can be further enhanced with GP. Regarding *TR*, there are many alternatives available. In this work we analyzed the most common and effective *TR* weights as reported in the literature [35,4,26,11,21] and considered them as building blocks for generating TWSs. Finally we also considered some constants as building blocks. The full set of building blocks (terminals in the tree representation) considered is shown in Table 1, whereas the set of operators considered in the proposed method (i.e., the function set, $\mathcal{F}$) is shown in Table 2.

In the proposed approach, a TWS is seen as a combination of building blocks by means of arithmetic operators. One should note, however, that three types of building blocks are considered: *TDR*, *TR* and constants. Hence we must define a way to combine matrices (*TDR* weights), vectors (*TR* scores) and scalars (the constants), in such a way that the combination leads to a TWS (i.e., a form of *TDR*). Accordingly, and for easiness of implementation, each building block shown in Table 1 is processed as a matrix of the same length as the TWS (i.e., $N \times |V|$) and operations are performed element-wise. In this way a tree can be directly evaluated, and the operators are applied between each element of the matrices, leading to a TWS.

TDRs are already matrices of the same size as the TWSs: $N \times |V|$. In the case of *TRs*, we have a vector of length $|V|$, thus for each *TR* we generate a matrix of size $N \times |V|$ where each of its rows is the *TR*; that is, we repeat *N* times the TR weight. In this way, for example, a TWS like *TF-IDF* can be obtained as *TF* × *IDF*, where the × operator means that each element $tf_{i,j}$ of *TF* is multiplied by each element of the *IDF* matrix $idf_{i,j}$ and where $idf_{i,j} = \log\left(\frac{N}{df(t_j)}\right)$ for $i = 1, \ldots, N$, all *TRs* were treated similarly. In the case of constants we use a scalar-matrix operator, which means that the constant is operated with each element of the matrix under analysis.

Estimating the matrices each time a tree is evaluated can be a time consuming process, therefore, at the beginning of the search process we compute the necessary matrices for every terminal from Table 1. Hence, when evaluating an individual we only have to use the values of the precomputed matrices and apply the operators specified by a tree.

4.2.3. Genetic operators

As explained above, in GP a population of individuals is initialized and evolved according to some operators that aim at improving the quality of the population. This section details such operators.

For initialization we used the standard ramped-half-and-half strategy [13], which generates half of the population with (balanced) trees of maximum depth (using the full method), and the other half with trees of variable depth (with the grow method). This is the most common initialization operator in GP. For this work we fixed the maximum initialization deep to 6, see Table 2. One should note that this restriction on the maximum depth is on the initialization stage only, during the search process deeper trees may be generated; the maximum deep allowed during the search process is of 17, longer trees are considered as invalid solutions and they are assigned a quite small fitness value (e.g., 0 or $-\infty$).

As genetic operators we also used standard mechanisms: we considered the subtree crossover and point mutation operators,

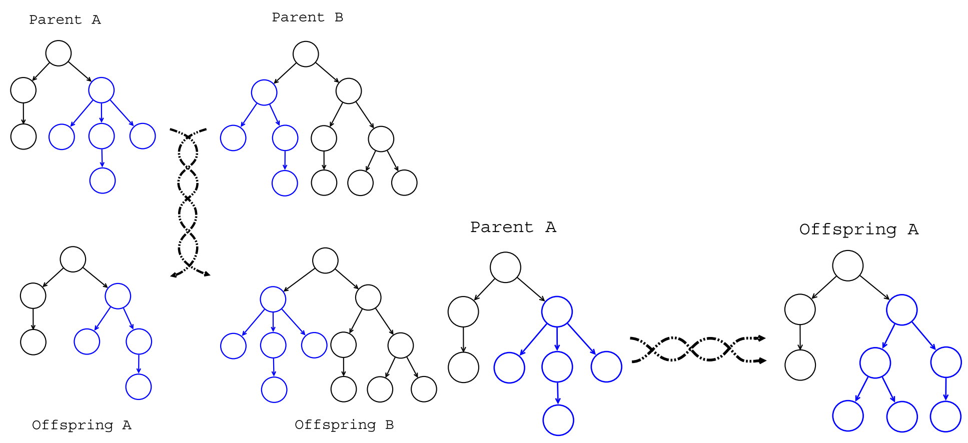

Fig. 3. Graphical description of crossover (left) and mutation (right) operators considered in this work.

Fig. 3 describes graphically these operators. The role of crossover is to take two promising solutions and combine their information to give rise to two offspring, with the goal that the offspring have better performance than the parents. The subtree crossover works by selecting two parent solutions/trees (in our case, via tournament[3]) and randomly select an internal node in each of the parent trees (see the top of the left plot in Fig. 3). Two offspring are created by interchanging the subtrees below the identified nodes in the parent solutions (see the bottom of the left plot in Fig. 3).

The function of the mutation operator is to produce random variations in the population, facilitating the exploration capabilities of GP. The considered mutation operator first selects an individual to be mutated. Next an internal node of the individual is identified, and next the selected node is replaced by a randomly generated subtree (see the right plot in Fig. 3). One should note that the considered crossover and mutation operators may produce trees whose deep exceeds the maximum allowed (17), in such case, the solution is considered as invalid.

4.2.4. Fitness function

As previously mentioned, the aim of the proposed GP approach is to generate a TWS that obtains competitive classification performance. In this direction, the goodness of an individual is assessed via the classification performance of a predictive model that uses the representation generated by the TWS. Specifically, given a solution to the problem, we first evaluate the tree to generate a TWS using the training set. Once training documents are represented by the corresponding TWS, we perform a k-fold cross-validation procedure to assess the effectiveness of the solution. In k-fold cross validation, the training set is split into k disjoint subsets, and k rounds of training and testing are performed; in each round $k-1$ subsets are used as training set and 1 subset is used for testing, the process is repeated k times using a different subset for testing each time ($k = 2$ was used in all of our experiments). The average classification performance is directly used as fitness function. Specifically, we evaluate the performance of classification models with the f_1 measure. Let TP, FP and FN to denote the true positives, false positives and false negative rates for a particular class, precision ($Prec$) is defined as $\frac{TP}{TP+FP}$ and recall (Rec) as $\frac{TP}{TP+FN}$. f_1-measure is simply the harmonic average between precision and recall: $f_1 = \frac{2\times Prec\times Rec}{Prec+Rec}$. The average across classes is reported (also called, macro-average f_1), this way of estimating the f_1-measure is known to be particularly useful when tackling unbalanced data sets [35].

Since under the fitness function k models have to be trained and tested for the evaluation of a single TWS, we need to look for an efficient classification model that, additionally, can deal naturally with the high-dimensionality of data. Support vector machines (SVM) comprise a type of models that have proved to be very effective for TC [35,22]. SVMs can deal naturally with the sparseness and high dimensionality of data, however, training and testing an SVM can be a time consuming process. Therefore, we opted for efficient implementations of SVMs that have been proposed recently [46,12]. That methods are trained online and under the scheme of learning with a budget. We use the predictions of an SVM as the fitness function for learning TWSs. Among the methods available in [12] we used the low-rank linearized SVM (LLSMV) [46]. LLSVM is a linearized version of non-linear SVMs, which can be trained efficiently with the so called block minimization framework [6]. We selected LLSVM instead of alterative methods, because this method has outperformed several other efficient implementations of SVMs, see e.g., [12,46].

4.3. Summary

We have described the proposed approach to learn TWSs via GP. When facing a TC problem we start by estimating all of the terminals described in Table 1 for the training set. The terminals are feed into the genetic program, together with the function set. We used the GPLAB toolbox for implementing the genetic program with default parameters [36]. The genetic program searches for the tree that maximizes the k-fold cross validation performance of an efficient SVM using training data only. After a fixed number of generations, the genetic program returns the best solution found so far, the best TWS. Training and test (which was not used during the search process) data sets are represented according to such TWS. One should note that all of the supervised term-weights in Table 1 are estimated from the training set only (e.g., the information gain for terms is estimated using only the labeled training data); for representing test data we use the pre-computed term-weights. Next, the LLSVM is trained in training data and the trained model makes predictions for test samples. We evaluate the performance of the proposed method by comparing the predictions of the model and the actual labels for test samples. The next section reports results of experiments that aim at evaluating the validity of the proposed approach.

[3] The so called Lexictour tournament was considered [36,28], which consists in selecting a random number of individuals from the population and using the best solution in the subset as parent. Whenever two individuals have equal fit, the shortest one is chosen as parent.

5. Experiments and results

This section presents an empirical evaluation of the proposed TWL approach. The goal of the experimental study is to assess the effectiveness of the learned TWSs and compare their performance to existing schemes. Additionally, we evaluate the generalization performance of learned schemes, and their effectiveness under different settings.

5.1. Experimental settings

For experimentation we considered a suite of benchmark data sets associated to three types of tasks: thematic TC, authorship attribution (AA, a non-thematic TC task) and image classification (IC). Table 4 shows the characteristics of the data sets. We considered three types of tasks because we wanted to assess the generality of the proposed approach.

Seven thematic TC data sets were considered, in these data sets the goal is to learn a model for thematic categories (e.g., sports news vs. religion news). The considered data sets are the most used ones for the evaluation of TC systems [35]. For TC data sets, indexing terms are the words (unigrams), stop words were removed and stemming was applied before indexing. Likewise, seven data sets for AA were used, the goal in these data sets is to learn a model capable of associating documents with authors. Opposed to thematic collections, the goal in AA is to model the writing style of authors, hence, it has been shown that different representations and attributes are necessary for facing this task [39]. Accordingly, indexing terms in AA data sets were 3-grams of characters, that is, sequences of 3-characters found in documents, these terms have proved to be the most effective ones in AA [39,16,29]; stop words were not removed for AA data sets. Finally, two data sets for image classification, taken from the CALTECH-101 collection, were used. We considered the collection under the standard experimental settings (15 images per class for training and 15 images for testing), two subsets of the CALTECH-101 data set were used: a small one with only 5 categories and the whole data set with 102 classes (101 object categories plus background) [19]. Images were represented under the Bag-of-Visual-Words formulation using dense sift descriptors (PHOW features): sift descriptors were extracted from training images, and then clustered using k-means, the centers of the clusters (i.e., prototypical visual descriptors) are considered as the visual words (i.e., the indexing terms); both, training and test, images are then represented by accounting the frequency of occurrence of visual words (visual descriptors in each image are compared to the visual words, each visual descriptor is assigned to their nearest visual word), the VLFEAT toolbox was used for processing images [42].

Table 4
Data sets considered for experimentation.

Data set	Classes	Terms	Train	Test
Text categorization				
Reuters-8	8	23583	5339	2333
Reuters-10	10	25,283	6287	2811
20-Newsgroup	20	61,188	11,269	7505
TDT-2	30	36,771	6576	2818
WebKB	4	7770	2458	1709
Classic-4	4	5896	4257	2838
CADE-12	12	193,731	26,360	14,618
Authorship attribution				
CCA-10	10	15,587	500	500
Poetas	5	8970	71	28
Football	3	8620	52	45
Business	6	10,550	85	90
Poetry	6	8016	145	55
Travel	4	11,581	112	60
Cricket	4	10,044	98	60
Image classification				
Caltech-101	101	12,000	1530	1530
Caltech-tiny	5	12,000	75	75

The considered data sets have been partitioned into training and test subsets (the number of documents for each partition and each data set are shown in Table 4). For some data sets there were predefined categories, while for others we randomly generated them using 70% of documents for training and the rest for testing. All of the preprocessed data sets in Matlab format are publicly available under request.

For each experiment, the training partition was used to learn the TWS, as explained in Section 4. The learned TWS is then evaluated in the corresponding test subset. We report two performance measures: accuracy, which is the percentage of correctly classified instances, and f_1 measure, which assesses the tradeoff between precision and recall across classes (macro-average f_1), recall that f_1 was used as fitness function (see Section 4).

The genetic program was run for 50 generations using populations of 50 individuals, we would like to point out that in each run of the proposed method we have used default parameters. It is expected that by optimizing parameters and running the genetic program for more generations and larger populations we could obtain even better results. The goal of our study, however, was to show the potential of our method even with default parameters.

5.2. Evaluation of TWS learning via genetic programming

This section reports experimental results on learning TWSs with the genetic program described in Section 4. The goal of this experiment is to assess how TWSs learned via GP compare with traditional TWSs. The GP method was run on each of the 16 data sets from Table 4, since the vocabulary size for some data sets is huge we decided to reduce the number of terms by using term-frequency as criterion. Thus, for each data set we considered the top 2000 more frequent terms during the search process. In this way, the search process is accelerated at no significant loss of accuracy. In Section 5.3 we analyze the robustness of our method when using the whole vocabulary size for some data sets.

For each data set we performed 5 runs with the GP-based approach, we evaluated the performance of each learned TWS and report the average and standard deviation of performance across the five runs. Tables 5–7 show the performance obtained by TWSs learned for thematic TC, AA and IC data sets, respectively. In the mentioned tables we also show the result obtained by the *best baseline* in each collection. *Best baseline* is the best TWS we found (from the set of TWSs reviewed in related work and the TWSs in Table 1) for each data set (using the test-set performance). Please note that under these circumstances *best baseline* is in fact, a quite strong baseline for our GP method. Also, we would like to emphasize that no parameter of the GP has been optimized, we used the same default parameters for every execution of the genetic program. For completion, we also show in Table 5 the performance obtained by the best TWS obtained with the method of [9]. For this, we evaluated all of the schemes reported in Cummins and O'Riordan [9,10,8] and report the result for the best TWS we found.[4] It is important to emphasize that this TWS is tailored for information retrieval tasks and we provide results for illustration purposes.

[4] The selected weight is defined as follows: $\mathbf{B} \times \left(\frac{\log N/df}{\sqrt{df}} \times \log \frac{cf}{df} \times \log df \right)$, where $\mathbf{B}$ is the Boolean TWS, N the number of documents in the collection, df the number of documents a term appears in, and cf the frequency of a term in the collection. The first term, $\mathbf{B}$, is a TDR weight whereas the second part is a *TR* factor. For details see Eq. (9) in [9].

Table 5
Classification performance on thematic TC obtained with learned TWSs, the best baseline, and the TWS of [9].

Data set	PG-avg.		Best baseline			Cummins et al.	
	f_1	Acc.	f_1	Acc.	Baseline	f_1	Acc.
Reuters-8	**90.56 ± 1.43**	**91.35 ± 1.99**	86.94	88.63	TF	74.44	88.30
Reuters-10	**88.21 ± 2.69**	91.84 ± 1.01	85.24	**93.25**	TFIDF	76.59	87.59
20-Newsgroup	**66.23 ± 3.84**	**67.97 ± 4.16**	59.21	61.99	TF	65.78	66.45
TDT-2	**96.95 ± 0.41**	**96.95 ± 0.57**	95.20	95.21	TFIDF	95.71	97.02
WebKB	**88.79 ± 1.26**	**89.12 ± 1.30**	87.49	88.62	B	71.97	75.42
Classic-4	**94.75 ± 1.08**	**95.42 ± 0.67**	94.68	94.86	TF	86.59	87.14
CADE-12	**41.03 ± 4.45**	**53.80 ± 4.0**	39.30	41.89	TF	38.91	48.44

Table 6
Classification performance on AA obtained with learned TWSs, the best baseline, and the TWS of [9].

Data set	PG-avg.		Best baseline			Cummins et al.	
	f_1	Acc.	f_1	Acc.	Baseline	f_1	Acc.
CCA-10	**70.32 ± 2.73**	**73.72 ± 2.14**	65.90	73.15	TF-IG	1.82	10
Poetas	**72.23 ± 1.49**	**72.63 ± 1.34**	70.61	71.84	TF-IG	7.36	22.54
Football	76.37 ± 9.99	83.76 ± 4.27	76.45	**83.78**	TF-CHI	**76.54**	77.78
Business	**78.08 ± 4.87**	**83.58 ± 1.57**	73.77	81.49	TF-CHI	75.48	75.56
Poetry	**70.03 ± 7.66**	74.05 ± 7.38	59.93	**76.71**	B	44.89	56.36
Travel	**73.92 ± 10.26**	**78.45 ± 6.72**	71.75	75.32	TF-CHI	67.58	68.33
Cricket	88.10 ± 7.12	**92.06 ± 3.29**	89.81	91.89	TF-CHI	**91.70**	91.67

Table 7
Classification performance on IC obtained with learned TWSs, the best baseline, and the TWS of [9].

Data set	PG- avg.		Best baseline			Cummins et al.	
	f_1	Acc.	f_1	Acc.	Baseline	f_1	Acc.
Caltech-101	**61.91 ± 1.41**	**64.02 ± 1.42**	58.43	60.28	B	52.54	54.18
Caltech-tiny	**89.70 ± 2.44**	**91.11 ± 2.36**	85.65	86.67	TF	82.43	82.67

From Table 5 it can be seen that, regarding the best baseline, different TWSs obtained better performance for different data sets. Hence evidencing the fact that different TWSs are required for different problems. On the other hand, it can be seen that the average performance of TWSs learned with our GP outperformed significantly the best baseline in all but one result (accuracy for Reuters-10 data set). The differences in performance are large, mainly for the f_1 measure, which is somewhat expected as this was the measure used as fitness function (recall f_1 measure is appropriate to account for the class imbalance across classes); hence showing the competitiveness of our proposed approach for learning effective TWSs for thematic TC tasks. Regarding the TWS from [9], it can be seen that this TWS performed, in general, the worst. This is not that surprising as this TWS has been optimized for information retrieval tasks, yet, for some tasks the performance is competitive with the best baseline (see, e.g., 20-Newsgroup, CADE-12 and TDT-2 data sets).

From Table 6 it can be seen that for AA data sets the best baseline performs similarly as the proposed approach. In terms of f_1 measure and accuracy, our method outperforms the best baseline in 5 out of 7 data sets. Therefore, our method still obtains comparable (slightly better) performance to the best baselines, which for AA tasks were much more competitive than in thematic TC problems. One should note that for PG we are reporting the average performance across 5 runs, among the 5 runs we found TWSs that consistently outperformed the best baseline. The performance of the TWS from [9] in AA data sets is erratic: it obtains the best performance in f_1 measure for two data sets, but it obtained performance close to random guessing in two other data sets, our method outperforms significantly such TWS.

It is quite interesting that, comparing the best baselines from Tables 5 and 6, for AA tasks supervised TWSs obtained the best results (in particular TF-CHI in 4 out of 7 data sets), whereas for thematic TC unsupervised TWSs performed better. Again, these results show that different TWSs are required for different data sets and different types of problems. In fact, our results confirm the fact that AA and thematic TC tasks are quite different, and, more importantly, our study provides evidence on the suitability of supervised TWSs for AA; to the best of our knowledge, supervised TWSs have not been used in AA problems.

Table 7 shows the results obtained for the image categorization data sets. Again, the proposed method obtained TWSs that outperformed the best baselines and the TWS from [9]. This result is quite interesting because we are showing that the TWS plays a key role in the classification of images under the BOVWs approach. In computer vision most of the efforts so far have been devoted to the development of novel/better low-level image-descriptors, using a BOW with predefined TWS. Therefore, our results pave the way for research on learning TWSs for image categorization and other tasks that rely in the BOW representation (e.g. speech recognition and video classification).

Fig. 4 and Table 8 complement the results presented so far. Fig. 4 indicates the difference in performance between the (average of) learned TWSs and the best baseline for each of the considered data sets. We can clearly appreciate from this figure the magnitude of improvement offered by the learned TWSs, which in some cases is too large.

Table 8, on the other hand, shows a more fair comparison between our method and the reference TWSs: it shows the average performance obtained by reference schemes and the average performance of our method for thematic TC, AA and IC data sets. It is clear from this table that in average our method performs consistently better than any of the reference methods in terms of both accuracy and f_1 measure for the three types of tasks. Thus, from the results of this table and those from Tables 5–7, it is evident that standard TWSs are competitive, but one can take advantage of

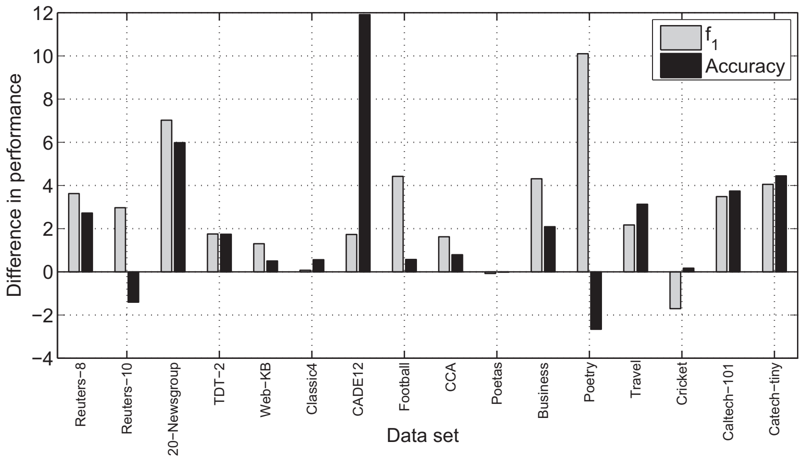

Fig. 4. Difference in performance between learned TWSs and best baseline per each data set, values above zero indicate better performance obtained by the TWSs.

Table 8
Average performance on thematic TC obtained with learned TWSs and the baselines.

TWS	Thematic TC		AA		IC	
	f_1	Acc.	f_1	Acc.	f_1	Acc.
TF	76.60	79.53	62.17	72.43	68.86	71.54
B	77.42	79.73	66.07	76.76	71.22	72.78
TFIDF	61.69	76.17	40.88	55.26	62.27	67.56
TF-CHI	71.56	75.63	68.75	73.69	65.38	67.45
TF-IG	64.22	69.00	68.96	74.91	66.02	67.93
PG-worst	77.81	81.19	66.47	74.84	74.30	75.67
PG-Avg.	**81.01**	**83.63**	**75.58**	**79.75**	**75.81**	**77.07**
PG-best	82.88	85.81	81.37	83.98	76.97	78.18

them only when the right TWS is selected for each data set. Also, the performance of TWSs learned with our approach are a better option than standard TWSs, as in average we were able to obtain much better representations.

Summarizing the results from this section, we can conclude that:

- The proposed GP obtained TWSs that outperformed the best baselines and the TWS from [9] in the three types of tasks: thematic TC, AA and IC. Evidencing the generality of our proposal across different data types and modalities. Larger improvements were observed for thematic TC and IC data sets. In average, learned TWSs outperformed standard ones in the three types of tasks.
- Our results confirm our hypothesis that different TWSs are required for facing different tasks, and within a same task (e.g., AA) a different TWS may be required for a different data set. Hence, motivating further research on how to select TWS for a particular TC problem.
- We show evidence that the proposed TWS learning approach is a promising solution for enhancing the classification performance in other tasks than TC, e.g., IC.
- Our results show that for AA supervised TWS seem to be more appropriate, whereas unsupervised TWS performed better on thematic TC and IC. This is a quite interesting result that may have an impact in non-thematic TC and supervised term-weighting learning.

5.3. Varying vocabulary size

For the experiments from Section 5.2 each TWS was learned by using only the top 2000 most frequent terms during the search process. This reduction in the vocabulary allowed us to speed up the search process significantly, however, it is worth asking ourselves what the performance of the TWSs would be when using an increasing number of terms. We aim to answer such question in this section.

For this experiment we considered three data sets, one from each type of task: thematic TC, AA, and IC. The considered data sets were the Reuters-8 (R8) for thematic TC, the CCA benchmark for AA, and Caltech-101 for IC. These data sets are the representative ones from each task: Reuters-8 is among the most used TC data sets, CCA has been widely used for AA as well, and Caltech-101 is *the benchmark* in image categorization For each of the considered data sets we use a specific TWS learned using the top-2000 most frequent terms (see Section 5.2), and evaluate the performance of such TWSs when increasing the vocabulary size: terms were sorted in ascending order of their frequency. Figs. 5–7 show the results of this experiment in terms of f_1 measure and accuracy (the selected TWS is shown in the caption of each image).

Different performance behavior can be observed in the different data sets. Regarding Fig. 5, which shows the performance for a thematic TC data set, it can be seen that the TWS learned by our method outperformed all other TWSs for any data set size. Hence confirming the suitability of the proposed method for thematic TC.

Fig. 6, on the other hand, behaves differently: the proposed method outperforms all the other TWSs only for a single data set size (when 20% of the terms were used). In general, our method consistently outperformed TF-CHI and TF-IG TWSs, and performs similarly as TF-IDF, but it was outperformed by the TF-RF TWS. This result can be due to the fact that for this AA data set, the genetic program learned a TWS that was suitable only for the vocabulary size that was used during the optimization. Although, interesting, this result is not that surprising: in fact, it is well known in AA that the number of terms considered in the vocabulary plays a key role on the performance of AA systems. AA studies suggest that using a small amount of the most-frequent terms when approaching an AA problem results in better performance [39,16,29]. Our results from Fig. 6 corroborate the latter and seem to indicate that when approaching an AA problem, one should first determine an appropriate vocabulary size and then apply our method. One should note, however, that our method outperforms the other TWSs for the data set size that was used during the optimization, and this is, in fact, the highest performance that can be obtained with any other TWS and data set size combination.

Finally, Fig. 7 reports the performance of TWSs for the Caltech-101 data set under different data set sizes. In this case, the learned TWS outperforms all other TWSs when using more than 20% and 30% in terms of f_1 measure and accuracy, respectively. The

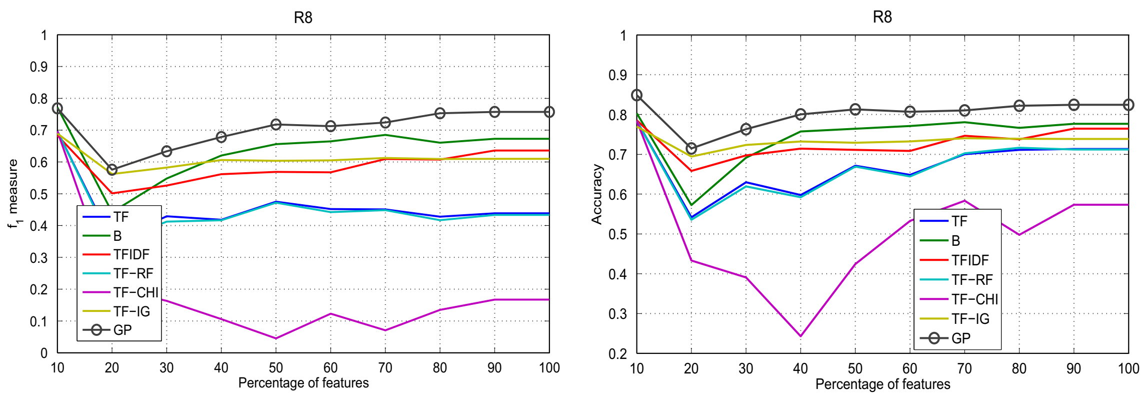

Fig. 5. Classification performance on the Reuters-8 data set for the TWS: $\sqrt{\mathbf{W}_5} - \frac{\log\sqrt{\mathbf{W}_{19}}}{\mathbf{W}_{21}}$ when increasing the number of considered terms. The left plot shows results in terms of f_1 measure while the right plot shows accuracy performance.

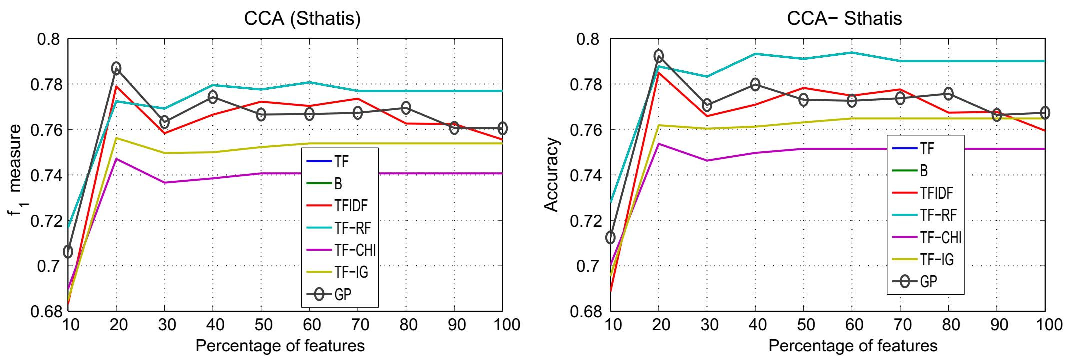

Fig. 6. Classification performance on the CCA data set of the TWS: $\mathbf{W}_4 - (\mathbf{W}_{22} + \mathbf{W}_5)$ when increasing the number of considered terms.

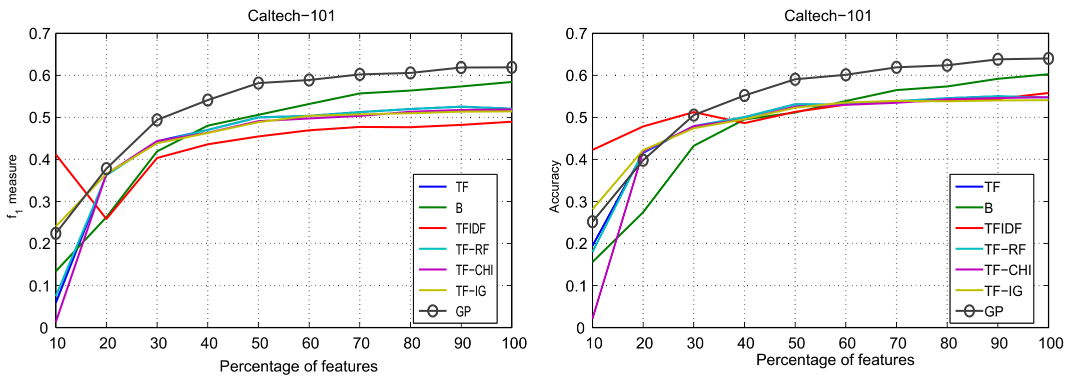

Fig. 7. Classification performance on the Caltech-101 data set of the TWS: $\sqrt{\sqrt{\mathbf{W}_{17} - \sqrt{\sqrt{\mathbf{W}_{22}}}}}$ when increasing the number of considered terms.

improvement is consistent and monotonically increases as more terms are considered. Hence showing the robustness of the learned TWS when increasing the vocabulary size for IC tasks. Among the other TWSs, TFIDF obtains competitive performance when using a small vocabulary, this could be due to the fact that when considering a small number of frequent terms the IDF component is important for weighting the contribution of each of the terms.

Summarizing the results from this section we can conclude the following:

- TWSs learned with our method are robust to variations in the vocabulary size for thematic TC and IC tasks. This result suggests, we can learn TWSs using a small number of terms (making more efficient the search process) and evaluating the learned TWSs with larger vocabularies.
- Learned TWSs outperform standard TWSs in thematic TC and IC tasks when varying the vocabulary size.
- For AA, TWSs learned with our proposed approach seem to be more dependent on the number of terms used during training. Hence, when facing this type of problems it is a better option to fix the number of terms beforehand and then running our method.

5.4. Generalization of the learned term-weights

In this section we evaluate the inter-data set generalization capabilities of the learned TWSs. Although results presented so far show the generality of our method across three types of tasks, we have reported results obtained with TWSs that were learned for each specific data set. It remains unclear whether the TWSs learned

Table 9
Considered TWSs for the inter-data set generalization experiment for each data set. In column 2 each TWS is shown as a prefix expression, the names of the variables are self-explanatory. Column 3 shows the mathematical expression of each TWS using the terminal set from Table 3.

ID	Data set	Learned TWS	Formula
Text categorization			
1	Reuters-8	-(sqrt(TFIDF),div(log2(sqrt(ProbR)),RF))	$\sqrt{\mathbf{W}_5} - \frac{\log\sqrt{\mathbf{W}_{19}}}{\mathbf{W}_{21}}$
2	Reuters-10	sqrt(div(pow2(sqrt(TFIDF)),div(pow2(TF-RF),pow2(TF-RF))))	$\sqrt{\frac{(\sqrt{\mathbf{W}_5})^2}{\frac{\mathbf{W}_{22}^2}{\mathbf{W}_{22}^2}}}$
3	20-Newsg.	sqrt(sqrt(div(TF,GLOBTF)))	$\sqrt{\sqrt{\frac{\mathbf{W}_6}{\mathbf{W}_7}}}$
4	TDT-2	sqrt(×(sqrt(sqrt(TFIDF)), sqrt(×(sqrt(TFIDF),IG))))	$\sqrt{\sqrt{\sqrt{\mathbf{W}_5}} \times \sqrt{\sqrt{\mathbf{W}_5} \times \mathbf{W}_4}}$
5	WebKB	div(TF-RF,+(+(+(RF,TF-RF),FMEAS),FMEAS))	$\frac{\mathbf{W}_{22}}{(((\mathbf{W}_{21}+\mathbf{W}_{22})+\mathbf{W}_{16})+\mathbf{W}_{16})}$
6	Classic-4	×(ProbR,TFIDF)	$\mathbf{W}_5 \times \mathbf{W}_{19}$
7	CADE-12	div(TF,sqrt(log2(ACCU)))	$\frac{\mathbf{W}_6}{\sqrt{\log \mathbf{W}_{12}}}$
Authorship attribution			
8	CCA-10	-(IG,plus(TF-RF,TFIDF))	$\mathbf{W}_4 - (\mathbf{W}_{22} + \mathbf{W}_5)$
9	Poetas	-(-(RF,TF-RF),TF-IDF)	$(\mathbf{W}_{21} - \mathbf{W}_{22}) - \mathbf{W}_5$
10	Football	div(TF-RF,pow2(ODDSR))	$\frac{\mathbf{W}_{22}}{\mathbf{W}_{17}^2}$
11	Business	minus(TF-RF,PROBR)	$\mathbf{W}_{22} - \mathbf{W}_{19}$
12	Poetry	div(TF,log2(div(TF,log(TF-RF))))	$\frac{\mathbf{W}_6}{\log\frac{\mathbf{W}_6}{\log \mathbf{W}_{22}}}$
13	Travel	+(-(TF-RF,-(-(-(TF-RF,-(TF-RF,POWER)),POWER)),TF)	$(\mathbf{W}_{22} - ((\mathbf{W}_{22} - (\mathbf{W}_{22} - \mathbf{W}_{18})) - \mathbf{W}_{18})) + \mathbf{W}_6$
14	Cricket	×(IG,TF-RF)	$\mathbf{W}_4 \times \mathbf{W}_{22}$
Image classification			
15	Caltech-101	sqrt(sqrt(-(ODDSR,sqrt(sqrt(TF-RF)))))	$\sqrt{\sqrt{\mathbf{W}_{17} - \sqrt{\sqrt{\mathbf{W}_{22}}}}}$
16	Caltech-tiny	sqrt(-(TF-RF,ACBAL))	$\sqrt{\mathbf{W}_{22} - \mathbf{W}_{13}}$

for a collection can perform similarly in other collections, we aim to answer to this question in this section.

To assess the inter-data set generalization of TWSs learned with our method we performed an experiment in which we considered for each data set a single TWS and evaluated its performance across all the 16 considered data sets. The considered TWSs are shown in Table 9, we named the variables with meaningful acronyms for clarity but also show the mathematical expression using variables as defined in Table 3.

Before presenting the results of the experiments it is worth analyzing the type of solutions (TWSs) learned with the proposed approach. First of all, it can be seen that the learned TWSs are not too complex: the depth of the trees is small and solutions have few terminals as components. This is a positive result because it allows us to better analyze the solutions and, more importantly, it is an indirect indicator of the absence of the over-fitting phenomenon. Secondly, as in other applications of genetic programming, it is unavoidable to have unnecessary terms in the solutions, for instance, the subtree: *div(pow2(TF-RF),pow2(TF-RF))*, (from TWS 2) is unnecessary because it reduces to a constant matrix; the same happens with the term *pow2(sqrt(TFIDF))*. Nevertheless, it is important to emphasize that this type of terms do not harm the performance of learned TWSs, and there are not too many of these type of subtrees. On the other hand, it is interesting that all of the learned TWSs incorporate supervised information. The most used TR weight is RF, likewise the most used TDR is TFIDF. Also it is interesting that simple operations over standard TWSs, TR and TDR weights results in significant performance improvements. For instance, compare the performance of TF-RF and the learned weight for Caltech-101 in Fig. 7. By simply subtracting an odds-ratio from the TF-RF TWS and applying scaling operations, the resultant TWS outperforms significantly TF-RF.

The 16 TWSs shown in Table 9 were evaluated in the 16 data sets in order to determine the inter-data set generality of the learned TWSs. Fig. 8 shows the results of this experiment. We show the results with boxplots, where each boxplot indicates the normalized[5] performance of each TWSs across the 16 data sets, for completion we also show the performance of the reference TWSs on the 16 data sets.

It can be seen from Fig. 8 that the generalization performance of learned TWSs is mixed. On the one hand, it is clear that TWSs learned for thematic TC (boxplots 7–13) achieve the highest generalization performance. Clearly, the generalization performance of these TWSs is higher than that of traditional TWSs (boxplots 1–6). It is interesting that TWSs learned for a particular data set/problem/modality perform well across different data sets/problems/modalities. In particular, TWSs learned for *Reuters-10* and *TDT-2* obtained the highest performance and the lowest variance among all of the TWSs. On the other hand, TWSs learned for AA and IC tasks obtained lower generalization performance: the worst in terms of variance is the TWS learned for the *Poetry* data set, while the worst average performance was obtained by the TWS learned for the *Football* data set. TWSs learned for IC are competitive (in generalization performance) with traditional TWSs. Because of the nature of the tasks, the generalization performance of TWSs learned from TC is better than that of TWSs learned for AA and IC. One should note that these results confirm our findings from previous sections: (i) the proposed approach is very effective mainly for thematic TC and IC tasks; and, (ii) AA data sets are difficult to model with TWSs.

Finally, we evaluate the generality of learned TWSs across different classifiers. The goal of this experiment is to assess the extend to which the learned TWSs are tailored for the classifier they were learn for. For this experiment, we selected two TWSs corresponding to Caltech-tiny and Caltech-101 (15 and 16 in

[5] Before generating the boxplots we normalized the performance on a per-data set basis: for each data set, the performance of the 16 TWSs was normalized to the range $[0, 1]$, in this way, the variation in f_1-values across data sets is eliminated, i.e, all f_1 values are in the same scale and are comparable to each other.

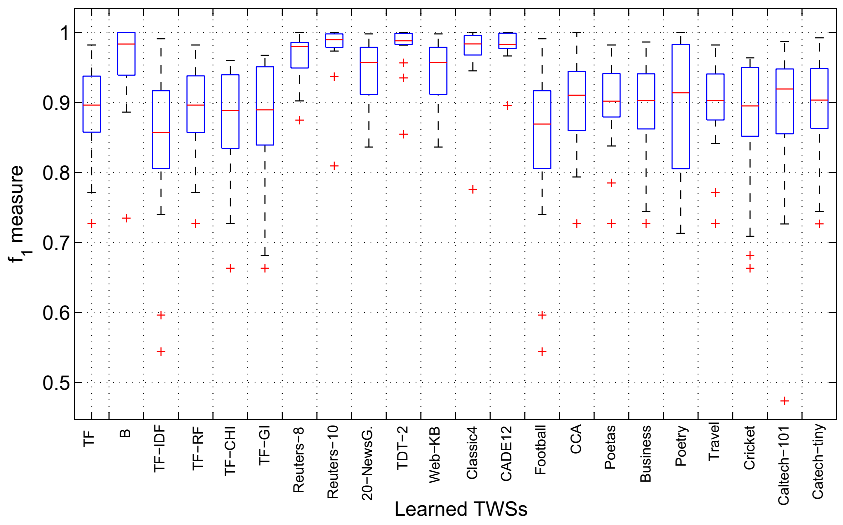

Fig. 8. Boxplots reporting the average performance of TWSs from Table 9 in the 16 data sets (x axis) considered in the study. For completion, we also show the performance of standard TWSs.

Table 9) and evaluated their performance of different classifiers across the 16 data sets. Fig. 9 shows the results of this experiment.

It can be seen from Fig. 9 that the considered TWSs behaved quite differently depending on the classifier. On the one hand, the classification performance when using naïve Bayes (*Naive*), kernel-logistic regression (KLogistic), and 1−nearest neighbors (*KNN*) classifiers degraded significantly. On the other hand, the performance of SVM and the neural network (NN) was very similar. These results show that TWSs are somewhat robust across classifiers of similar nature as SVM and NN are very similar classifiers: both are linear models in the parameters. The other classifiers

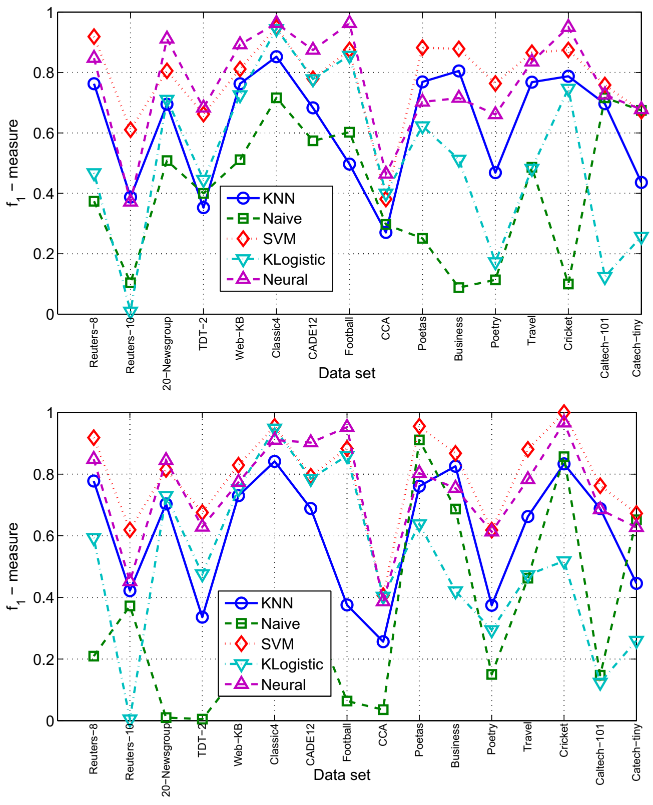

Fig. 9. Classification performance of selected TWSs across different classifiers, f_1 measure is reported. The plot at the top is a TWS learned for Caltech-tiny, while the bottom plot shows the performance for a TWS learned for Caltech-101.

are quite different to the reference SVM and, therefore, the performance is poor.[6] It is interesting that in some cases the NN classifier outperformed the SVM, although in average the SVM performed better. This is a somewhat expected result as the performance of the SVM was used as fitness function.

An experiment that we have postponed for future work is that of using the performance of alternative classifiers for the fitness function. In the current setting this is not feasible because, nowadays, learning on a budget is not a well studied topic for other classifiers than SVM; therefore, we would have to use standard implementations of classifiers, which would increase the computational cost of our TWS learning strategy.

According to the experimental results from this section we can draw the following conclusions:

- TWSs learned with the proposed approach are not too complex despite their effectiveness. Most of the learned TWSs included a supervised component, evidencing the importance of taking advantage of labeled documents.
- TWSs offer acceptable inter-data set generalization performance, in particular, TWSs learned for TC generalize pretty well across data sets.
- We showed evidence that TWSs learned for a modality (e.g., text/ images) can be very competitive when evaluated on other modality.
- Learned TWSs are somewhat robust to the classifier choice. It is preferable to use the classifier used to estimate the fitness function, although classifiers of similar nature perform similarly.

5.5. Computational cost and convergence

This section provides insights into the computational cost and convergence capabilities of the proposed approach. Table 10 reports the average runtime[7] of the TWS learning process for three selected data sets: the one which took more time (20-Newsgroup), one with average runtime (Reuters-8), and the one which took the less time (Caltech-tiny).

[6] One should note that among the three worse classifiers, KNN, Naive and KLogistic, the latter obtained better performance than the former two, this is due to the fact that KLogistic is closer, in nature, to an SVM.

[7] Experiments were performed in a Laptop (Qosmio X75A Toshiba) with processor Corei7-4700 at 2.4 GHz and 32 GB in RAM. All code was implemented in Matlab.

Table 10
Runtime (in min) of the TWS learning process for selected data sets.

Data set	Runtime (m)
20-Newsgroup	1094.57 ± 147.14
Reuters-8	276.94 ± 98.53
Caltech-tiny	24.39 ± 13.21

As previously mentioned, it can be seen that our method is somewhat computationally expensive, where computational cost is proportional with the size of the data set. For the 20-Newsgroup data set our method took more than 18 h to obtain a TWS, whereas for the Caltech-tiny data set it took less than half an hour; for the mid-size collection, Reuters-8, took almost 5 h. Although being computational demanding, one should note that the process of learning TWSs is an offline procedure that, in practice, has to be done a single time. Therefore, we can say that applying our method is affordable for data sets of size comparable to those used in our experiments.

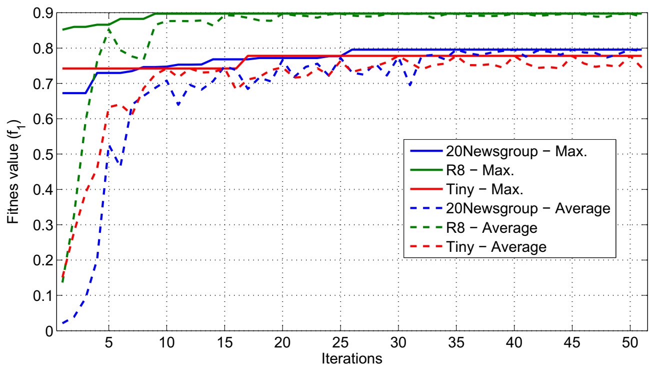

Fig. 10. Maximum and average fitness value obtained by the genetic program during the search process for 20-Newsgroup, Reuters-8 and Caltech-tiny data sets.

On the other hand, recall in all of the previously reported experiments the genetic programm was ran for 50 generations only. Thus, it is worth analyzing the convergence behavior of our method. Fig. 10 shows the maximum and average fitness function value obtained during the search process for the 20-Newsgroup, Reuters-8 and Caltech-tiny data sets.

It can be seen from this figure that after generation 25, approximately, the maximum value of the fitness value does not change. The average fitness value of the population shows more variation, but after generation 35 it seems to converge. In Fig. 11 we analyze the behavior of the fitness function when running the genetic program for up to 200 generations for the Reuters-8 data set. Similarly, the maximum value of the fitness function does not change after generation 20 or so. The standard deviation of the fitness values (top plot in Fig. 11) shows that the algorithm converges very fast. Therefore running the genetic program for more generations does not seem to further improve the fitness value, this result is in agreement with GP literature [27].

Furthermore, one should note that early stopping (i.e., optimizing a fitness function for not too long) is a widely known procedure to avoid overfitting, when optimizing classification models (see e.g. Escalante et al. [15]). Hence, running the genetic program for not too many iterations may help to reduce the risk of capturing particularities of the training set that are not very helpful to generalize.

6. Conclusions

We have described a novel approach to term-weighting scheme (TWS) learning in text classification (TC). TWSs specify the way in which documents are represented under a vector space model. We proposed a genetic programming solution in which standard TWSs, term-document, and term relevance weights are combined to give rise to effective TWSs. We reported experimental results in 16 well-known data sets comprising thematic TC, authorship attribution and image classification tasks. The performance of the

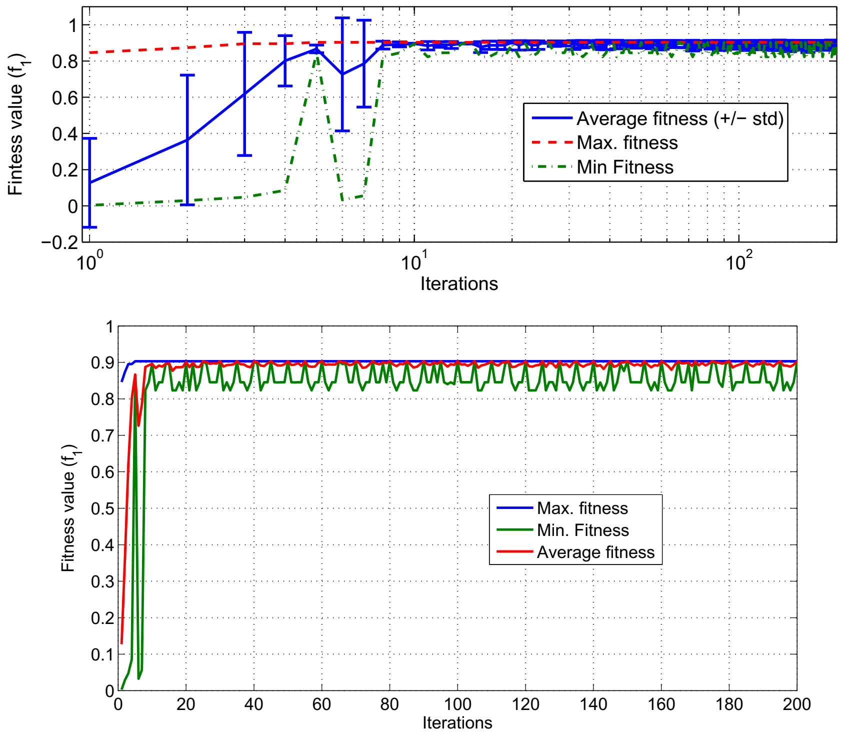

Fig. 11. Maximum, minimum and average fitness value obtained by the genetic program during the search process for the Reuters-8 data set. The top plot shows errorbars around the average fitness value in log scale.

proposed method is evaluated under different scenarios. Experimental results show that the proposed approach learns very effective TWSs that outperform standard TWSs. The main findings of this work can be summarized as follows:

- TWSs learned with the proposed approach outperformed significantly to standard TWSs and those proposed in related work.
- Defining the appropriate TWS is crucial for image classification tasks, an ignored issue in the field of computer vision.
- In authorship attribution, supervised TWSs are beneficial, in comparison with standard TWSs.
- The performance of learned TWSs do not degrades when varying the vocabulary size for thematic TC and IC. For authorship attribution a near-optimal vocabulary size should be selected before applying our method.
- TWSs learned for a particular data set or modality can be applied to other data sets or modalities without degrading the classification performance. This generalization capability is mainly present in TWSs learned for thematic TC and IC.
- Learned TWSs are easy to analyze/interpret and do not seem to overfit the training data.

Future work directions include studying the suitability of the proposed approach to learn weighting schemes for cross domain TC. Also we would like to perform an in deep study on the usefulness of the proposed GP for computer vision tasks relying in the Bag-of-Visual-Words formulation.

Acknowledgments

This work was supported by CONACYT under project grant CB-2014-241306 (*Clasificación y recuperación de imágenes mediante técnicas de minería de textos*). Also this work was partially supported by the LACCIR programme under project ID R1212LAC006. Hugo Jair Escalante was supported by the internships programme of CONACyT under Grant No. 234415.

References

[1] B. Agarwal, N. Mittal, Text classification using machine learning methods-a survey, in: Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), of Advances in Intelligent Systems and Computing, vol. 236, 2014, pp. 701–709.
[2] C.C. Aggarwal, Mining Text Data, Springer, 2012 (Ch. A Survey of Text Classification Algorithms).
[3] C.C. Aggarwal, C. Zhai (Eds.), Mining Text Data, Springer, 2012.
[4] H. Altyncay, Z. Erenel, Analytical evaluation of term weighting schemes for text categorization, Patt. Recog. Lett. 31 (2010) 1310–1323.
[5] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999.
[6] K.W. Chang, D. Roth, Selective block minimization for faster convergence of limited memory large-scale linear models, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2011.
[7] G. Csurka, C.R. Dance, L. Fan, J. Willamowski, C. Bra, Visual categorization with bags of keypoints, in: International Workshop on Statistical Learning in Computer Vision, 2004.
[8] R. Cummins, C. O'Riordan, Evolving general term-weighting schemes for information retrieval: tests on larger collections, Artif. Intell. Rev. 24 (2005) 277–299.
[9] R. Cummins, C. O'Riordan, Evolving local and global weighting schemes in information retrieval, Inf. Retrieval 9 (2006) 311–330.
[10] R. Cummins, C. O'Riordan, Evolved term-weighting schemes in information retrieval: an analysis of the solution space, Artif. Intell. (2007).
[11] F. Debole, F. Sebastiani, Supervised term weighting for automated text categorization, in: Proceedings of the 2003 ACM Symposium on Applied Computing, SAC '03, ACM, New York, NY, USA, 2003, pp. 784–788. http://doi.acm.org/10.1145/952532.952688.
[12] N. Djuric, L. Lan, S. Vucetic, Z. Wang, Budgetedsvm: a toolbox for scalable SVM approximations, J. Mach. Learn. Res. 14 (2013) 3813–3817.
[13] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Natural Computing, Springer, 2010.
[14] H.J. Escalante, M. Montes, L. Villasenor, Particle swarm model selection for authorship verification, in: Proceedings of the 14th Iberoamerican Congress on Pattern Recognition, LNCS, vol. 5856, Springer, 2009, pp. 563–570.
[15] H.J. Escalante, M. Montes, E. Sucar, Particle swarm model selection, J. Mach. Learn. Res. 10 (2009) 405–440.
[16] H.J. Escalante, T. Solorio, M.M. Gomez, Local histograms of character *n*-grams for authorship attribution, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011, pp. 288–298.
[17] W. Fan, E.A. Fox, P. Pathak, H. Wu, The effects of fitness functions on genetic programming based ranking discovery for web search, J. Am. Soc. Inf. Sci. Technol. 55 (7) (2004) 628–636.
[18] W. Fan, M.D. Gordon, P. Pathak, A generic ranking function discovery framework by genetic programming for information retrieval, Inf. Process. Manage. (2004).
[19] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories, in: IEEE, CVPR 2004, Workshop on Generative-Model Based Vision, 2004.
[20] R. Feldman, J. Sanger, The text mining handbook advanced approaches in analyzing unstructured data, ABS Vent. (2006).
[21] G. Forman, An extensive empirical study of feature selection metrics for text classification, J. Mach. Learn. Res. 3 (2003) 1289–1305.
[22] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of ECML-98, LNCS, vol. 1398, Springer, 2008, pp. 137–142.
[23] C. Kiddon, Y. Brun, That's what she said: double entendre identification, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011, pp. 89–94.
[24] M. Koppel, S. Argamon, A.R. Shimoni, Automatically categorizing written texts by author gender, Liter. Ling. Comput. 17 (4) (2002) 401–412.
[25] M. Koppel, J. Schler, Authorship verification as a one-class classification problem, in: Proceedings of the 21st International Conference on Machine Learning, 2004, p. 62.
[26] M. Lan, C.L. Tan, J. Su, Y. Lu, Supervised and traditional term weighting methods for automatic text categorization, Trans. PAMI 31 (4) (2009) 721–735.
[27] W.B. Langdon, R. Poli, Foundations of Genetic Programming, Springer, 2001.
[28] S. Luke, L. Panait, Lexicographic parsimony pressure, in: Proceedings of GECCO-2002, 2002, pp. 829–836.
[29] K. Luyckx, W. Daelemans, The effect of author set size and data size in authorship attribution, Liter. Ling. Comput. (2010) 1–21.
[30] D. Mladenic, M. Grobelnik, Feature selection for unbalanced class distribution and nave Bayes, in: Proceedings of the 16th Conference on Machine Learning, 1999, pp. 258–267.
[31] N. Oren, Re-examining tf.idf based information retrieval with genetic programming, in: SAICSIT, 2002.
[32] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up, sentiment classification using machine learning techniques, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002, pp. 79–86.
[33] A. Reyes, P. Rosso, On the difficulty of automatically detecting irony: beyond a simple case of negation, Knowl. Inf. Syst. 40 (3) (2014) 595–614.
[34] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manage. (1988) 513–523.
[35] F. Sebastiani, Machine learning in automated text categorization, ACM Comp. Surv. 34 (1) (2008) 1–47.
[36] S. Silva, J. Almeida, Gplab-a genetic programming toolbox for MATLAB, in: Proceedings of the Nordic MATLAB Conference, 2003, pp. 273–278.
[37] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos, in: International Conference on Computer Vision, vol. 2, 2003, pp. 1470–1477.
[38] S. Manchala, V.K. Prasad, V. Janaki, GMM based language identification system using robust features, Int. J. Speech Technol. 17 (2014) 99–105.
[39] E. Stamatatos, A survey of modern authorship attribution methods, J. Ame. Soc. Inform. Sci. Technol. 60 (3) (2009) 538–556.
[40] A. Trotman, Learning to rank, Inform. Retrieval 8 (2005) 359–381.
[41] P. Turney, P. Pantel, From frequency to meaning: vector space models of semantics, J. Artif. Intell. Res. 37 (2010) 141–188.
[42] A. Vedaldi, B. Fulkerson, VLFeat: An Open and Portable Library of Computer Vision Algorithms, 2008.
[43] J. Wanga, P. Liub, M.F. Shea, S. Nahavandia, A. Kouzanid, Bag-of-words representation for biomedical time series classification, Biomed. Signal Process. Control 8 (6) (2013) 634–644.
[44] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization. In: Proceedins of the 14th International Conference on Machine Learning, 1997, pp. 412–420.
[45] J. Zhang, M. Marszablek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: a comprehensive study, Int. J. Comput. Vis. 73 (2) (2007) 213–238.
[46] K. Zhang, L. Lan, Z. Wang, F. Moerchen, Scaling up kernel SVM on limited resources: a low-rank linearization approach, in: Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS), 2012.