entropy

Article

Evolutionary Optimization of Ensemble Learning to Determine Sentiment Polarity in an Unbalanced Multiclass Corpus

Consuelo V. García-Mendoza [1], Omar J. Gambino [1], Miguel G. Villarreal-Cervantes [2] and Hiram Calvo [3,*]

1 Escuela Superior de Cómputo, Instituto Politécnico Nacional, J.D. Bátiz e/M.O. de Mendizábal, Mexico City 07738, Mexico; cvgarcia@ipn.mx (C.V.G.-M.); jjuarezg@ipn.mx (O.J.G.)
2 Centro de Innovación y Desarrollo Tecnológico en Cómputo, Instituto Politécnico Nacional, J.D. Bátiz e/M.O. de Mendizábal, Mexico City 07700, Mexico; mvillarrealc@ipn.mx
3 Centro de Investigación en Computación, Instituto Politécnico Nacional, J.D. Bátiz e/M.O. de Mendizábal, Mexico City 07738, Mexico
* Correspondence: hcalvo@cic.ipn.mx; Tel.: +52-55-57296000 (ext. 56516)

Received: 7 August 2020; Accepted: 10 September 2020; Published: 12 September 2020

Abstract: Sentiment polarity classification in social media is a very important task, as it enables gathering trends on particular subjects given a set of opinions. Currently, a great advance has been made by using deep learning techniques, such as word embeddings, recurrent neural networks, and encoders, such as BERT. Unfortunately, these techniques require large amounts of data, which, in some cases, is not available. In order to model this situation, challenges, such as the Spanish TASS organized by the Spanish Society for Natural Language Processing (SEPLN), have been proposed, which pose particular difficulties: First, an unwieldy balance in the training and the test set, being this latter more than eight times the size of the training set. Another difficulty is the marked unbalance in the distribution of classes, which is also different between both sets. Finally, there are four different labels, which create the need to adapt current classifications methods for multiclass handling. Traditional machine learning methods, such as Naïve Bayes, Logistic Regression, and Support Vector Machines, achieve modest performance in these conditions, but used as an ensemble it is possible to attain competitive execution. Several strategies to build classifier ensembles have been proposed; this paper proposes estimating an optimal weighting scheme using a Differential Evolution algorithm focused on dealing with particular issues that multiclass classification and unbalanced corpora pose. The ensemble with the proposed optimized weighting scheme is able to improve the classification results on the full test set of the TASS challenge (General corpus), achieving state of the art performance when compared with other works on this task, which make no use of NLP techniques.

Keywords: sentiment polarity; ensemble learning; unbalanced classes; evolutionary optimization; Twitter sentiment analysis

1. Introduction

Sentiment polarity refers to the opinion people have about an entity (e.g., film, service, news, etc.). Several machine learning methods have been used to automatically determine polarity of text published on Internet [1–4]. In general, polarity is automatically determined in various domains using different approaches, for example, in health prediction [5–8] or transportation [9].

The task of sentiment polarity task can be tackled as a supervised classification problem, where classes correspond to the polarity expressed in opinions (v. gr. positive or negative); classifiers are trained on tagged examples and they generate a model that relates features to the corresponding tag. Some classifiers are able to learn better particular features, while other classifiers that fail on particular cases perform well on others. Ensemble learning uses a set of classifiers to combine their predictions in different ways. [10] showed that an ensemble of classifiers is more accurate than its individual members if each of these members has an error rate less than 0.5, and they generate different errors when classifying new instances—i.e., members are accurate and diverse. A way to combine their predictions is to apply a voting strategy than can give the same weight to each classifier (hard voting) or different weights (soft voting). There are two determining factors concerning a voting ensemble that have been studied, the set of classifiers to be combined [11] and the weight assigned to each classifier [12]. This work focuses on the second problem.

Our main goal is to find the best weights assigned to each classifier in ensemble learning while using a soft voting scheme to improve sentiment polarity classification in a multiclass, unbalanced corpus. Particularly, we focus on the Spanish TASS task [13] organized by the Spanish Society for Natural Language Processing (SEPLN) (www.sepln.org). There are several challenges to tackle in this task:

- The fact that polarities are specified in four labels: positive, negative, neutral, and none. Thus, this task has to be modeled as a multiclass problem.
- The corpus we experiment with, possesses several difficulties: it is designed to have a small training subset (approximately 10%), while the test set is around 90%.
- Classes are not uniformly represented, and their distribution varies in the two subsets.

Hence, in this work, the weighting scheme for Twitter sentiment polarity in an unbalanced corpus with four possible polarity values (positive, negative, neutral, and none) is addressed through an optimization approach. This approach involves the formulation of an optimization problem, where its solution is based on the use of the differential evolution algorithm.

Despite that several works [14,15] have proposed different strategies for calculating the weighting scheme of an ensemble, the corpora used for their experiments are balanced, so that, to our knowledge, the effects of applying a weighting scheme on an unbalanced corpus have not been explored. In addition, classifiers of previous works are designed to only learn two possible outputs (positive or negative). Adjusting the weighting scheme for multiclass classification, unlike binary classification, could be more challenging when considering the number of possible combinations in the solutions. Therefore, this paper proposes a solution to optimize the weighting scheme of an ensemble to tackle a multiclassification problem with unbalanced classes.

The rest of this paper is structured, as follows: Section 2 gives details of current methods dealing with this problem. Section 3 describes some preliminaries, such as details on the selected task, classifiers, and a formal definition of ensemble learning; Section 4 presents the main proposal of this work—the evolutionary optimization of the weighted ensemble classification. In Section 5, the experiments and results are presented, and finally in Section 6 our conclusions are drawn.

2. Related Work

The problem of learning from unbalanced datasets has been addressed in early works, such as [16]. With the aim of improving the performance of SVMs in the imbalanced dataset context, the authors integrate over-sampling and under-sampling to balance the data and propose the ensemble of SVM (EnSVM) model in order to integrate the classification results of weak classifiers constructed individually on the processed data, and develop a genetic algorithm-based model called EnSVM+ to improve the performance of classification through classifier selection. Inspired by this work, we aimed to propose an ensemble, but focused both on linguistic features and multiclass problems.

Regarding linguistic features, ref. [17] describes a linguistic analysis framework, in which a number of similarity or dissimilarity features are extracted for each entailment pair in a data set and various classifier methods are evaluated based on the instance data that were derived from the extracted features. They compare and contrast the performance of single and ensemble based learning algorithms of datasets from the RTE1 to RTE5 challenges. They show that only one heterogeneous ensemble approach demonstrated a slight improvement over the technique of Naïve Bayes and none of the homogeneous methods were more accurate than Naïve Bayes. Nevertheless, finding an optimal combination of classifiers is still an important issue.

Over the past few years, the use of evolutionary computing techniques in classification tasks has increased because these techniques help finding an approximate solution closer to the global solution, while retaining, at the same time, independence to particular characteristics of the optimization problem, such as discontinuities, nonlinearities, the need of discrete design variables, etc. Additionally, evolutionary computing techniques are flexible in the sense that they allow merging diverse strategies in order to improve the exploration and exploitation capabilities of the algorithm in the evolutionary process. In [18], the multi-objective version of Binary Bat Algorithm with local search strategies employing social learning concepts in designing random walks is used on three widely-used micro array cancer datasets to explore significant bio-markers. A bio-inspired hierarchical model for analyzing musical timbre is presented in [19]; the model extracts three profiles for timbre: time-averaged spectrum, global temporal envelope, and instantaneous roughness. Different weight assignment for each features in ensemble learning-based classification has been applied in [20].

Related to text classification, the Arabic Text Classification system (ATC-FA) is proposed in [21]; this system combines the algorithm of Support Vector Machines (SVM) with an intelligent Feature Selection method (FS) based on the Firefly Algorithm (FA). Genetic programming has been used in [22] to generate alternative term-weighting schemes (TWSs) in text classification, allowing to improve the performance of current schemes in text classification by combining TWSs, terms (TRs), and term-document (TDRs) with a predefined set of operators.

In [23], a hybrid ensemble pruning scheme that is based on clustering and randomized search for text sentiment classification is proposed. A consensus clustering scheme is presented to deal with the instability of clustering results that consists of self-organizing map algorithm (SOM), expectation maximization (EM), and K-means++ (KM++). The classifiers of the ensemble are initially clustered into groups according to their predictive characteristics. Subsequently, two classifiers from each cluster are selected as candidate classifiers based on their pairwise diversity. The search space of candidate classifiers is explored by the elitist Pareto-based multi-objective evolutionary algorithm for diversity reinforcement (ENORA).

In [24], a model is introduced in order to predict whether a tweet contains a location or not and show that location prediction is a useful pre-processing step for location extraction. To evaluate the model, the Ritter dataset and MSM2013 dataset were used. To train the model, they tried different machine learning algorithms: the Naive Baiyes (NB), Support Vector Machine (SMO), and Random Forest (RF) using 10-folds cross validation. To optimize accuracy and true positives, the thresholds were varied (0.05, 0.20, 0.50, 0.75) for NB and RF, and for SMO was varied epsilon (0.05, 0.20, 0.50, 0.75). The conclusion was that RF and NB are the best machine learning solutions for this problem they perform better than SMO.

Usually, sets of classifiers are more accurate than the individual classifiers that integrate them when any of their individual members has an error rate of less than 0.5, and, in general, individual members have different errors when classifying new examples—that is, they are precise and diverse [10]. In recent years, deep learning methods have achieved high performance for several tasks; however, there are several problems for which a traditional machine learning approach is able to obtain state of the art results, given that an appropriate ensemble is constructed [15,25–28].

In this sense, different schemes have been tried to combine the predictions of the base classifiers that form the ensemble classification. Particularly, for the soft weighting scheme, there has been two main approaches: the use of meta-heuristic algorithms proposed by [14] and the estimation of a weighting scheme based on the probabilities of classifiers and their accuracy, as described by [15].

In [14], the use of meta-heuristic algorithms in the weighting of ensemble learning improves classification's performance. Onan et al. proposed including a weighted ensemble learning for the analysis of the polarity opinion (positive and negative) based on differential evolution. Ensemble learning incorporates the following classifiers: Bayesian Logistic Regression (BLR), NB, Linear Discriminant Analysis (LDA), LR, and SVM. The allocation of the appropriate weighting values to classifier outputs is established as an optimization problem where precision and recall are the objective functions. Their proposal improves the accuracy of the base classifiers and other classic methods of ensemble learning.

In [15], the polarity of opinion is determined in two classes (positive and negative) of tweets of the *Stanford Sentiment140* English corpus, proposing a combination scheme of the ensemble learning of the weights for the base classifiers NB, RandomForest (RF), SVM, and LR. The proposal considers the weighting of the accuracies of each base classifier along with their probabilities of predict a negative or positive class to calculate prediction scores. According to these scores, the authors determine the polarity of the training data. If negative and positive scores are equal, the cosine similarity is calculated with other tweets in test data and the most similar tweet prediction is chosen. With ensemble learning, the accuracy of the base classifier with better precision (SVM) is improved by 0.2%.

A multiobjective optimization-based weighted voting scheme was presented in [29]. Zhang et al. [30] propose adjusting the weight values of each base classifier by using the DE algorithm. Onan et al. [14] present a static classifier selection involving majority voting error and forward search; and, Ankit and Seleena [15] consider the weighting of the accuracies of each base classifier along with their probabilities of predict a negative or positive class to calculate prediction scores. It is important to recall that the corpora used for all these experiments are balanced (Except for *First GOP debate twitter sentiment* dataset used in [15]). Additionally, classifiers of previous works are designed to only learn two possible outputs (positive or negative). This is why this paper proposes estimating an optimal weighting scheme using a Differential Evolution algorithm focused in dealing with particular issues that multiclass classification and unbalanced corpora pose.

3. Preliminaries

This section outlines the selected task (Section 3.1); briefly describes it as a multiclassification task (Section 3.2); gives details on selected classifiers (Section 3.3); and, formally defines the problem of ensemble classification that will be used throughout the rest of this work (Section 3.4).

3.1. The Twitter Sentiment Polarity Task

In this research, the Spanish TASS corpus that was organized by the Spanish Society for Natural Language Processing (SEPLN) was used. This corpus contains 68,017 tweets divided into two sets: an E training set with 7219 tweets and a Z test set with 60,798, with polarity frequencies indicated in Table 1. As can be seen, this is a strongly unbalanced corpus, additional to the particularity that the test Z set is more than eight times greater than the training set E.

Table 1. Opinion polarity distribution of tweets in TASS corpus.

Polarity	E		Z	
	Frecuency	**Tweets**	**Frecuency**	**Tweets**
Positive	39.94%	2,884	36.57%	22,233
Negative	30.22%	2,182	26.06%	15,844
None	20.54%	1,483	35.22%	21,416
Neutral	9.28%	670	2.15%	1,305

This work aims to train with the E set to automatically classify the Z set of TASS in the polarities of opinion: Positive (P), Negative (N), None (NONE), or Neutral (NEU), hence the need of multiclass classification. This is detailed in next section.

3.2. Multiclass Classification

Classification problems can be categorized according to the number of different values that classes can have. In binary classification, there are only two mutually exclusive classes; for instance, the spam detection task has two possible outputs: spam or valid email [31]. When the classification problem has more than two possible class values, it is considered to be a multiclass problem. An example of multiclass classification is to determine whether an opinion is positive, negative, or neutral [32]. Well-known classifiers, like Decision Trees and Neural Networks, can handle multiclass problems natively, but binary classifiers can be adapted to support multiclass classification. One of the most used strategies to transform a multiclass problem to a binary problem is One vs One (OVO) [33]. This strategy divides the original data set into two-class subsets, learning a different model for each new subset. Consider the following dataset to better explain this strategy:

$$Classes = \{P, N, NEU, NONE\}$$

$$Instances = \{I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10}\}$$

$$\begin{aligned} Dataset = \{&\{I_1, P\}, \{I_2, N\}, \{I_3, P\}, \{I_4, NEU\}, \{I_5, NONE\}, \\ &\{I_6, N\}, \{I_7, NEU\}, \{I_8, P\}, \{I_9, NEU\}, \{I_{10}, N\}\} \end{aligned}$$

The OVO strategy creates a data subset for each possible combination of pair of classes. For an m class problem, OVO creates $\frac{m(m-1)}{2}$ data sets and each data set is used to train a binary classifier that can distinguish between different pairs of classes [34]. For the dataset above, OVO creates the following data sets:

$$Dataset_{P-N} = \{\{I_1, P\}, \{I_2, N\}, \{I_3, P\}, \{I_6, N\}, \{I_8, P\}, \{I_{10}, N\}\}$$

$$Dataset_{P-NEU} = \{\{I_1, P\}, \{I_3, P\}, \{I_4, NEU\}, \{I_7, NEU\}, \{I_8, P\}, \{I_9, NEU\}\}$$

$$Dataset_{P-NONE} = \{\{I_1, P\}, \{I_3, P\}, \{I_5, NONE\}, \{I_8, P\}\}$$

$$Dataset_{N-NEU} = \{\{I_2, N\}, \{I_4, NEU\}, \{I_6, N\}, \{I_7, NEU\}, \{I_9, NEU\}, \{I_{10}, N\}\}$$

$$Dataset_{N-NONE} = \{\{I_2, N\}, \{I_5, NONE\}, \{I_6, N\}, \{I_{10}, N\}\}$$

$$Dataset_{NEU-NONE} = \{\{I_4, NEU\}, \{I_5, NONE\}, \{I_7, NEU\}, \{I_9, NEU\}\}$$

New instances can be predicted based on majority voting.

3.3. Classifiers

Three classifiers were selected relying on previous results that have shown good performance [32] to create the ensemble. Because Logistic Regression and Support Vector Machines are binary classifiers, it was necessary to use the multiclass transformation strategy explained in Section 3.2. The classifying methods (referred as classifiers from now on) and their parameters are described next:

- Multinomial Naïve Bayes (NB). This is a native multiclass classification algorithm. It is based on Bayes's theorem. It is called *naïve* because of the assumption of class conditional independence, but, in spite of this, good results are obtained with this algorithm, comparable to other more complex techniques like neural networks [35]. This classifier has an additive smoothing parameter, called alpha, which value was set to 0.5.
- Logistic Regression (LR). Models the probability of events' occurrence as a linear function of a set of predictor variables and can be used to predict the value of dependent variables. Because this algorithm builds a prediction model instead of a estimated point of dependent variables, it is used as a effective classifier [36]. Parameter C corresponding to the inverse of regularization strength was set to 1.0.

- Support Vector Machine (SVM). This algorithm uses a nonlinear matching method to transform the original dataset into a higher dimension—namely, a hyper-plane that acts as the decision boundary for partitioning data into classes [37]. Support vector machines are used to determine an optimal decision boundary to partition data into different classes. It is important to mention that SVM does not generate class probabilities, but they were calculated while using the algorithm proposed by [38]. Radial base was used as kernel, with kernel coefficient (gamma) set to 0.00001 and the penalty parameter of the error term (C) set to 3500.

3.4. Ensemble Classification

Ensemble classification considers the output of several classifiers, whose individual decisions are combined in some way—typically by weighted or un-weighted voting—in order to classify new examples [12]. In this work, ensemble classification is used to classify new tweets, consisting in $n = 3$ classifiers denoted by $C_1, C_2, \ldots, C_n$. For each tweet $\hat{t}_q, q = 1, 2, \ldots, t$ classifier $C_i, i = 1, 2, \ldots, n$ generates m probabilities $P_{i,j}, j = 1, 2, \ldots, m$. $P_{i,1}$ indicates the probability generated by the classifier C_i that the q-th tweet $\hat{t}_q$ belongs to class L_1, $P_{i,2}$ the probability that it belongs to the class L_2 and so on for the m classes, as shown in Table 2.

Table 2. Probabilities generated by the classifier C_i, of which the q-th tweet $\hat{t}_q$ belongs to each of the $L_j, j = 1, 2, \ldots, m$ classes.

	L_1	L_2	…	L_m
C_i	$P_{i,1}$	$P_{i,2}$	…	$P_{i,m}$

The proposed weighting is a soft weighting scheme, in such a way that it weights the m probabilities generated by classifier $C_i, i = 1, 2, \ldots, n$ with weights $w_i, i = 1, 2, \ldots, n$. See Table 3.

Table 3. Weighting scheme of ensemble classification.

	L_1	L_2	…	L_m
C_1	$P_{1,1}w_1$	$P_{1,2}w_1$	…	$P_{1,m}w_1$
C_2	$P_{2,1}w_2$	$P_{2,2}w_2$	…	$P_{2,m}w_2$
…	…	…	…	…
C_n	$P_{n,1}w_n$	$P_{n,2}w_n$	…	$P_{n,m}w_n$

3.5. Accuracy of the Ensemble Classifier

There are different metrics to evaluate an automatic learning model: accuracy, model error, completeness, precision, recall, and F1 measure are some of them. In this work, accuracy will be used to evaluate the ensemble classifier.

Once the probabilities generated by the n classifiers are weighted, the probability weighted by class can be obtained for the q-th tweet that is described in (1).

$$\widehat{P}_j = \sum_{i=1}^{n} P_{i,j} w_i, \; j = 1, \ldots, m \tag{1}$$

The prediction of the ensemble learning for the q-th tweet is the maximum probability weighted by class described in (2).

$$D_q = max\{\widehat{P}_j\}, \; j = 1, \ldots, m \tag{2}$$

Given the set of predictions $D = \{D_1, D_2, \ldots, D_t\}$ of the ensemble learning and the set of real classes $R = \{R_1, R_2, \ldots, R_t\}$ of tweets $\hat{t}_q, q = 1, 2, \ldots, t$, it is possible to know the number of intersections between these two sets, as described in (3).

$$e = |D \cap R| \tag{3}$$

Finally, the accuracy of the ensemble learning with weights $\vec{w}$ is described in (4).

$$J(\vec{w}) = \frac{e}{t} \tag{4}$$

3.6. Maximum Theoretical Accuracy

The intention of this work is to maximize the accuracy of the ensemble classifier. However, it is good to have a reference to know how far it is possible to maximize this accuracy. For this, the maximum accuracy of n classifiers was calculated. The prediction of classifier i for the q-th tweet is described in (5).

$$a_{i,q} = max\{P_{i,j}\} \tag{5}$$

The predictions of n classifiers for the q-th tweet can be calculated, as shown in (6).

$$A_q = \{a_{1,q}, a_{2,q}, \ldots, a_{n,q}\} \tag{6}$$

If the cardinality of the intersection of the set A_q with the real class R_q of the q-th tweet is greater than zero, it is considered that there is a coincidence between any of the predictions of the n classifiers with the real class of the $q-$th tweet, as described in (7).

$$d_q = \begin{cases} 0, & \text{if } |A_q \cap R_q| = 0 \\ 1, & \text{if } |A_q \cap R_q| > 0 \end{cases} \tag{7}$$

The maximum theoretical accuracy of m classifiers is described in (8).

$$TA = \frac{\sum_{q=1}^{t} d_q}{t} \tag{8}$$

As an example, we calculate the maximum accuracy of three classifiers for five tweets $\hat{t}_q$, with real classes R_1, R_2, R_3, R_4, R_5, and predictions from three classifiers $a_{i,k}$ with cardinalities d_q, as shown in Table 4. For this example, the maximum accuracy described in (9) is obtained.

$$\frac{3}{5} = 0.6 \tag{9}$$

Table 4. Predictions of three classifiers for five tweets.

$\hat{t}_1$	$\hat{t}_2$	$\hat{t}_3$	$\hat{t}_4$	$\hat{t}_5$
$R_1 = NONE$	$R_2 = NEU$	$R_3 = P$	$R_4 = P$	$R_5 = N$
$a_{1,1} = P$	$a_{1,2} = NONE$	$a_{1,3} = P$	$a_{1,4} = N$	$a_{1,5} = N$
$a_{2,1} = N$	$a_{2,2} = P$	$a_{2,3} = P$	$a_{2,4} = NONE$	$a_{2,5} = N$
$a_{3,1} = NONE$	$a_{3,2} = P$	$a_{3,3} = N$	$a_{3,4} = NEU$	$a_{3,5} = N$
$d_1 = 1$	$d_2 = 0$	d_3=1	$d_4 = 0$	$d_5 = 1$

4. Evolutionary Optimization of the Weighted Ensemble Classification

A mono-objective optimization problem is considered in order to maximize the accuracy of the ensemble classifier. This can be described in a general way as maximizing $J(\vec{w})$ subject to (10) and (11).

The design goal $J(\vec{w})$ considers the e matches between the set of predictions $D = \{D_1, D_2, \ldots, D_t\}$ of the ensemble learning, and the set of real classes $R = \{R_1, R_2, \ldots, R_t\}$ of the test tweets $t_q,\ q = 1, 2, \ldots, t$, where $\vec{w}$ are the weights that must be adjusted to maximize J, as defined in (4).

The design variables are the weights that are assigned to classifiers $C_i,\ i = 1, \ldots, n$. The set of design variables are grouped in vector $\vec{w} = [w_1, w_2, \ldots w_n]$.

It is necessary to narrow the search space, establishing boundaries for the design variables, in order to find optimal solutions to real-world problems. In the case of this problem, these limits are established as the inequality constraints (10).

$$g_i : 0 < w_i < 1,\ \forall i = 1, \ldots, n \tag{10}$$

Another restriction that must be met is that the sum of the weights $\vec{w}$ assigned to classifiers $C_i, i = 1, \ldots, n$ must be equal to 1. This constraint is described in (11).

$$h_1 : \sum_{i=1}^{n} w_i = 1 \tag{11}$$

4.1. Differential Evolution Algorithm

Differential Evolution (DE) is an evolutionary algorithm proposed by Rainer Storn and Kenneth Price to solve global optimization problems in continuous search spaces [39]. DE has characteristics of robustness, precision, and speed of convergence that have made it attractive, not only to solve problems with continuous search spaces [40,41], but also discrete spaces [42,43]. DE begins with a set of solutions, called parents population, to which processes of crossing, mutation and selection are applied to create child populations that approach optimum solutions in an iterative process. The parameters of DE are: population size NP, maximum number of generations G_{Max}, number of crossings C_r, and a factor of scale F.

There are different variants of DE, being the most popular $rand/1/bin$—the one used in this work. The word *rand* indicates that the three individuals selected to calculate the mutation value are selected randomly, 1 the number of pairs of solutions chosen, and *bin* that a binomial recombination is used [44].

In this work, DE creates an initial matrix population $W_G = [\vec{w}_G^1, \ldots, \vec{w}_G^{NP}] \in \mathbb{R}^{NP \times \mathrm{m}}$ with NP individuals, called population of parents. Each individual of W_G contains the design variables $\vec{w}$ generated randomly, as described in (12) and (13), respecting the inequality constraints (10) and the equality constraint (11). In the mutation process a mutant individual $\vec{v}_G^i$ is created with three parent individuals ($\vec{w}_G^a$, $\vec{w}_G^b$ and $\vec{w}_G^c$) different to the current father $\vec{w}_G^i$ and the scale factor F. In the crossing process, the crossing factor C_r is considered to determine whether the gene inherited from the individual child $\vec{u}_G^{i,j}$ is taken from the mutant individual $\vec{v}_G^{i,j}$ or from the parent individual $\vec{w}_G^{i,j}$. Subsequently, it is verified if the child individual $\vec{u}_G^i$ complies with constraints (10) and (11). If not, $\vec{u}_G^i$ is randomly generated with (12) and (13). Finally, in the selection process, the individual parent of the next generation $\vec{w}_{G+1}^i$ will be the individual with greater accuracy comparing the child individual $\vec{u}_G^i$ and the parent individual $\vec{w}_G^i$. These processes continue iteratively while $G <= G_{\text{Max}}$. The population of the maximum generation $W_{G_{\text{Max}}}$ has the individuals with better accuracy for *Max* generations. Algorithm 1 shows the complete pseudo-code of the implementation of the $DE/rand/1/bin$ algorithm in order to optimize the weights of the ensemble classifier.

$$y_i = rand(0, 10)\ \forall i = 1, \ldots, n \tag{12}$$

$$w_i = \frac{y_i}{\sum_{i=1}^{n} y_i}\ \forall i = 1, \ldots, n \tag{13}$$

Algorithm 1: Pseudocode of the DE algorithm for the evolutionary optimization of the ensemble classifier.

```
$G = 0$
Create the initial population $\vec{w}^i_G, i = 1, \ldots, NP$
Evaluate $J(\vec{w}^i_G), i = 1, \ldots, NP$
while $G <= G_{Max}$ do
    for $i = 1$ to $NP$ do
        Select three individuals $\{\vec{w}^a_G \neq \vec{w}^b_G \neq \vec{w}^c_G \neq \vec{w}^i_G\} \in W_G$
        $g_{rand}$ is initialized as a random number with uniform discrete distribution in the
          interval $[1, n]$
        $j_{rand}$ is initialized as a random number with uniform continuous distribution in the
          interval $(0, 1)$
        for $j = 1$ to $n$ do
            if $j_{rand} < C_r$ or $g_{rand} = j$ then
                $\vec{v}^{i,j}_G = \vec{w}^{a,j}_G + F(\vec{w}^{b,j}_G - \vec{w}^{c,j}_G)$
                $\vec{u}^{i,j}_G = \vec{v}^{i,j}_G$
            else
                $\vec{u}^{i,j}_G = \vec{w}^{i,j}_G$
            end
        end
        If $\vec{u}^i_G$ does not comply with constraints (10) and (11), then it is generated randomly with
          (12) and (13)
        Evaluate $J(\vec{u}^i_G)$
        if $J(\vec{u}^i_G) < J(\vec{w}^i_G)$ then
            $\vec{w}^i_{G+1} = \vec{w}^i_G$
        else
            $\vec{w}^i_{G+1} = \vec{u}^i_G$
        end
    end
    $G = G + 1$
end
```

4.2. K-Fold Cross-Validation and Stratified K-Fold Cross-Validation

It is important to estimate the performance of classifiers in order to select the most appropriate scheme. A common strategy for this purpose is to use k-fold cross-validation, in which a dataset S is split in k mutually exclusive subsets, called folds, S_1, S_2, . . . , S_k of approximately the same size [45]. Subsequently, classifiers are trained and tested k times; each time $g \in \{1, 2, \ldots, k\}$, it is trained on the training subset $S - S_g$, and tested on S_g (testing subset).

A variation of this strategy, called stratified k-fold cross-validation, considers the distribution of classes to create the folds [45]. The folds in this strategy are evenly distributed, so that they contain approximately the same proportions of labels as the original dataset. In our proposal, for both strategies k is equal to 10, which is, the training set is divided in 10 folds.

Both of the strategies show the robustness of classifiers and the average accuracy of folds is a good estimator of expected performance on the test set. Therefore, we apply the evolutionary optimization method described in Algorithm 1 on each fold to calculate the best weighting scheme. Selection of the best weighting scheme is described in the following subsection.

4.3. Best Weights Selection Strategy

Evolutionary optimization algorithms provide a set of good solutions. From these solutions, the one that maximizes (or minimizes, depending on the problem) the objective function must be selected. A simple solution could just select the weighting scheme that maximizes accuracy, but this weighting scheme would have been calculated on a single fold of a test set, and there is no certainty that it could obtain the same good results in the test subsets from other folds. To avoid this bias in selecting the best solution, the next next steps are followed:

1. Train the classifiers described in Section 3.3 with each of the 10 training sets.
2. Use Algorithm 1 to determine the weighting schemes that maximizes accuracy on each of the 10 testing sets. In this step 10 candidates for best weighting scheme are obtained, one for each testing fold.
3. Use the obtained weighting schemes of each test set on the ensemble to classify the tweets of remaining nine test sets.
4. Calculate the average accuracy obtained by each weighting scheme of the ensemble on the test sets.
5. Select the weighting scheme with the best average accuracy.

As well as cross-validation ensures the robustness of the classifiers, we consider that the selection strategy described above takes advantage of the diversity of samples on the folds to provide a global solution (The apparently straightforward selection strategy of averaging weights from the best weighting vectors in each fold was also tested, with no satisfactory results.).

The complexity of Algorithm 1 is calculated as $\mathcal{O}(G_{Max} \cdot NP \cdot n)$, where G_{Max} is the number of generations for crossover and mutation of individuals of the NP population, and n is the number of design variables that corresponds to the number of weights to be assigned to the classifiers (in this case, 3).

5. Experiments And Results

Our goal is to be able to correctly classify the polarity of tweets of the test set Z of the TASS corpus, as described in Section 3.1. In order to do so, first the classifiers are trained and adjusted on the training corpus E. Experiments and results on this set are described in Section 5.1; afterwards, the experiments on the test set Z are described in Section 5.2.

5.1. Experiments on the E Set

Several strategies can be explored for training and adjusting the ensemble learning scheme with differential evolution weight selection. The number of individuals and generations can be changed (See Section 5.1.1), as well as the way of creating folds (see Section 5.1.2). With these experiments, the optimal weighting scheme $\vec{w}$ is sought. Subsequently, it will be applied to classify the TASS test set Z, as described in Section 5.2.

5.1.1. Random Folds

Table 5 shows the results of the first experiment with the training set of the TASS corpus E and 10 folds. From rows one to three, the accuracy obtained by the NB, LR, and SVM classifiers is observed independently. The fourth row (*Hard w*) shows the accuracy obtained by ensemble classification with the same classifiers using a hard weighting (same weights for all classifiers). Row 5 (*Soft w:20–300*) shows the accuracy obtained by the ensemble classification using the weighting scheme of the best individual after the process of evolutionary optimization. For this selection a random initial population of 20 individuals over 300 generations was used and the experiment was run 30 times in order to ensure robustness in the results. The total execution time was about 16 h in a 20 core dual-processor Intel Xeon E5-2690V2 Server (TEN CORE @ 3.0 GHz). The average of these 30 runs is reported. For results shown in Row 6 (*Soft w:200–1000*) the population was increased from 20 individuals to 200, and generations were increased from 300 to 1000 with the intention of achieving greater diversity in the population. As well as in Row 5 (*Soft w:20–300*), the experiment was run 30 times. The execution time was similar to the previous experiment. Because these changes did not have a significant effect in results, no more increases in generations or individuals were tested. Row 7 (*TA*) shows the maximum theoretical accuracy that was obtained by ideally selecting the correct result, if provided by any classifier (see Section 3.6). The last column shows the average accuracy of the classifiers and the ensemble learning. The highest accuracy for each independent classifier on each fold is highlighted in italics, while the best overall accuracy (without considering TA) is shown in bold.

Table 5. Results on random folds of the TASS training set *E*.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Average
NB	0.5927	0.6066	*0.6745*	0.6149	0.6191	0.5775	0.6288	0.6288	0.5886	0.6282	0.6159
LR	*0.6301*	*0.6440*	0.6634	*0.6495*	0.6329	0.6204	*0.6481*	*0.6551*	*0.6246*	0.6324	0.6400
SVM	*0.6301*	0.6329	0.6606	0.6260	*0.6412*	*0.6218*	0.6426	0.6398	0.6218	*0.6421*	0.6358
Hard w	0.6163	0.6274	0.6897	0.6371	0.6315	0.6066	0.6481	0.6675	0.6218	0.6421	0.6388
Soft w:20–300	**0.6371**	**0.6468**	0.6939	**0.6537**	**0.6509**	**0.6343**	**0.6634**	**0.6828**	**0.6398**	**0.6560**	**0.6558**
Soft w:200–1000	**0.6371**	**0.6468**	**0.6966**	**0.6537**	**0.6509**	**0.6343**	**0.6634**	**0.6828**	**0.6398**	**0.6560**	**0.6561**
TA	0.7160	0.7299	0.7603	0.7229	0.7119	0.6966	0.7409	0.7368	0.7105	0.7392	0.7265

5.1.2. Stratified Folds

The folding strategy in previous experiments consisted of randomly selecting tweets from the *E* set of the TASS corpus. As can be seen in Figure 1, for some folds all classifiers in general achieve better accuracy. This might be due to the class bias of tweet polarities (unbalanced number of classes). Experiments with stratified k-folding were performed in order to lessen the impact of this bias on classification accuracy. In Table 6, results of using stratified k-folding are shown. Figure 2 shows performance for each fold using stratification. Both of the configurations of 20 individuals and 300 generations, and 200 individuals and 1000 generations were used. In general, soft weighting improves the classification accuracy on stratified folds as well. Nevertheless, there is still an heterogeneous performance for different folds.

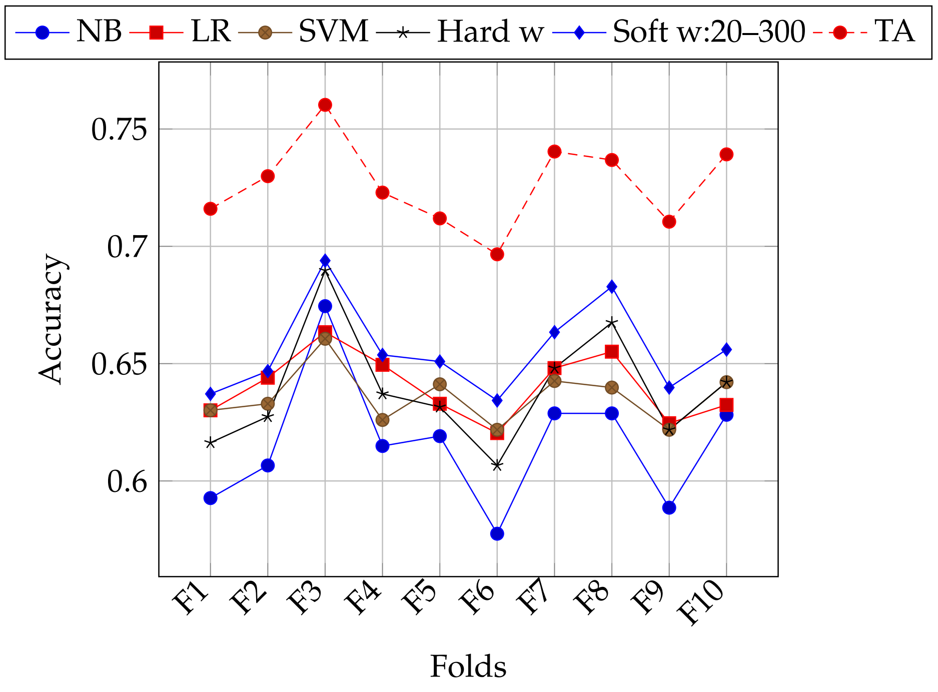

Figure 1. Accuracy of each classifier, different weighting schemes, and maximum theoretical accuracy on each fold of the *E* set of the TASS corpus.

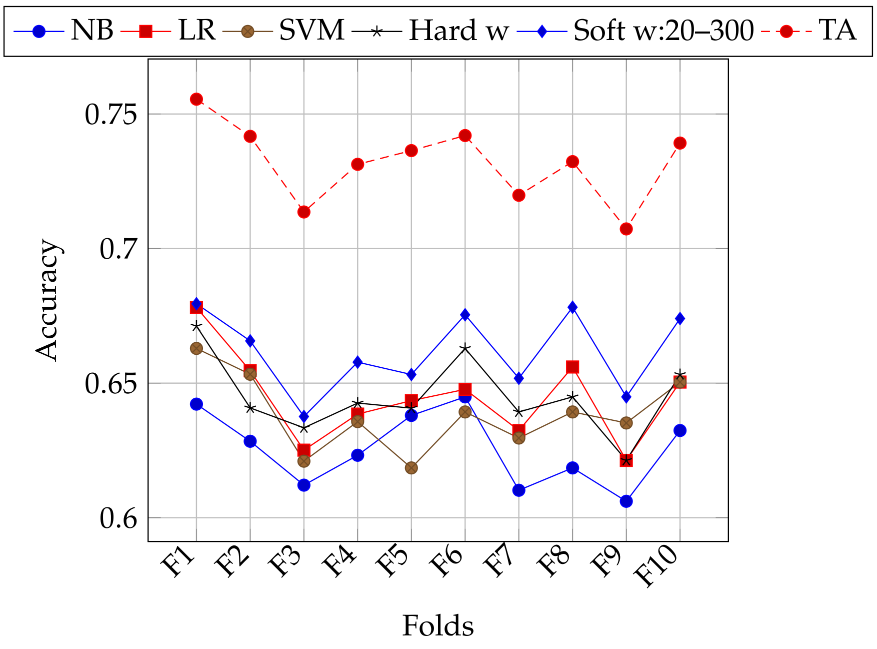

Figure 2. Accuracy of each classifier, different weighting schemes, and maximum theoretical accuracy on each stratified fold of the *E* set of the TASS corpus.

Figure 3 shows a comparison of accuracy obtained by *Soft w:20–300* on both folding strategies, random and stratified. Stratified folding improved the performance on most of the folds, but decreased on others. On average, the accuracy on random folds was 0.6558, while average accuracy on stratified folds was 0.6618.

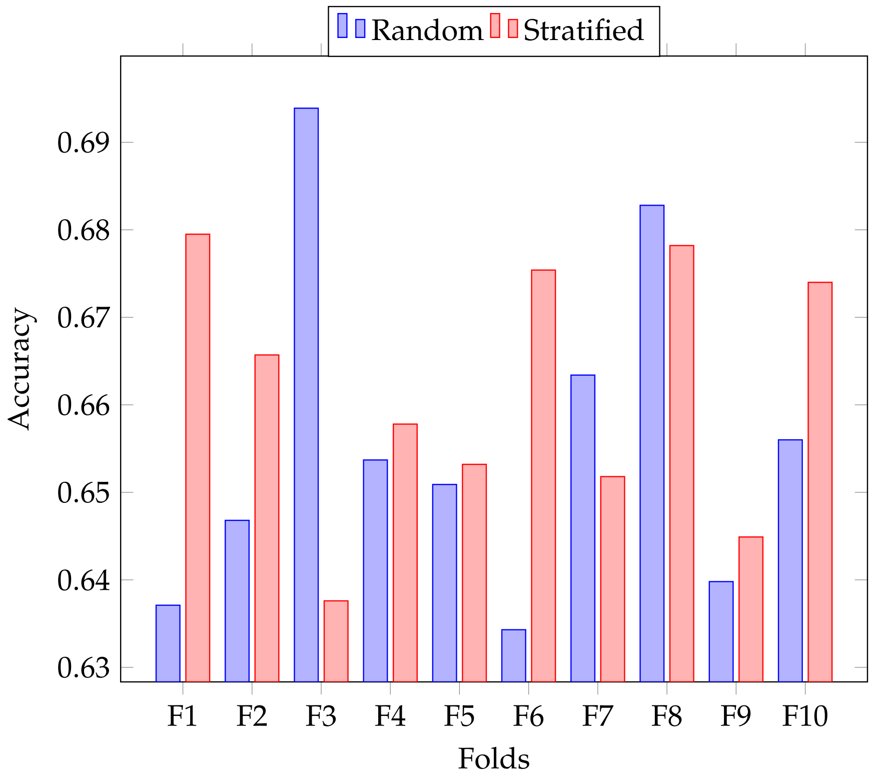

Figure 3. Accuracy of *Soft w:20–300* on both random and stratified folding strategy for each created fold of the *E* set of the TASS corpus.

For each fold, different soft weights were found. In the next section it is explained how the best weight on each folding strategy is selected in order to classify the tweets of the final test on the Z set.

Table 6. Results on stratified folds of the training set of the TASS corpus E.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Average
NB	0.6422	0.6284	0.6127	0.6232	0.6380	0.6449	0.6102	0.6185	0.6061	0.6324	0.6256
LR	*0.6781*	*0.6546*	*0.6251*	*0.6385*	*0.6435*	*0.6477*	*0.6324*	*0.6560*	0.6213	*0.6504*	0.6447
SVM	0.6629	0.6533	0.6210	0.6357	0.6185	0.6393	0.6296	0.6393	*0.6352*	*0.6504*	0.6385
Hard w	0.6712	0.6408	0.6334	0.6426	0.6407	0.6629	0.6393	0.6449	0.6213	0.6532	0.6450
Soft w:20–300	**0.6795**	**0.6657**	**0.6376**	**0.6578**	**0.6532**	**0.6754**	**0.6518**	**0.6782**	**0.6449**	0.6740	**0.6618**
Soft w:200–1000	**0.6795**	**0.6657**	**0.6376**	**0.6578**	**0.6532**	**0.6754**	**0.6518**	**0.6782**	**0.6449**	**0.6754**	**0.6619**
TA	0.7555	0.7417	0.7136	0.7313	0.7364	0.7420	0.7198	0.7323	0.7073	0.7392	0.7319

5.2. Experiments on the Z Set

For each experiment described in the previous sections, a vector of optimal weights $\vec{w}$ was obtained for each fold. The strategy detailed in Section 4.3 was applied for each experiment in order to select the best set of weights. The selected weighting vector on the random folds (from the *soft w:20–300* experiment) was $\vec{w}_r$ = [0.1713, 0.0380, 0.7905], while the vector corresponding to the stratified folds (using 20 individuals and 300 generations) was $\vec{w}_s$ = [0.1345, 0.0340, 0.8313]. Each value in this vector corresponds to the weight of each classifier, namely NB, LR, and SVM.

It can be seen that, in both $\vec{w}_r$ and $\vec{w}_s$, SVM is given a predominant weight (0.7905 and 0.8313 respectively); this is interesting, because this classifier obtained better average accuracy than NB, but lower than LR.

Once the weights were determined, tweets in the test set were classified with each classifier, and they were then assembled in a voting scheme with $\vec{w}_r$ and $\vec{w}_s$ weights, respectively. The results are shown in Table 7. As can be seen $\vec{w}_s$ based on stratified folds (which obtained better results in the training set E) also yielded the best result in the test set Z.

Table 7. Results of the experiment with Z set of the TASS corpus. **Soft** $\vec{w}_r$ shows results with soft weights calculated from random folds while those of **Soft** $\vec{w}_s$ were calculated from stratified folds.

Method	Accuracy
NB	0.6384
LR	0.6712
SVM	0.6721
Hard $\vec{w}$	0.6657
Soft $\vec{w}_r$	0.6768
Soft $\vec{w}_s$	**0.6771**

5.3. Comparison with Other Works

Table 8 presents the best results reported by other systems on the same task. To our knowledge, the best accuracy reported so far is 0.726 by the LIF system. However, in order to fairly compare these systems, it is necessary to consider the external resources they are using to improve classification. For example, the LIF system uses external affective lexicons, such as ElhPolar [46], SSL [47], LYSA [48], MPQA [49], and HGI [50]. A similar situation occurs with the first four systems with the highest accuracies. Isolation from the effect of other resources is desirable, as, in principle, we aim to improve classification accuracy by adjusting weights of a classification ensemble. In that sense, we are comparable to the LYS, SINAI-DW2Vec, and INGEOTEC systems. Our proposed classification method with soft weights on stratified k-folds overcomes the accuracy of these systems.

Table 8. External resources used and Accuracy for TASS Task 1, 4 classes, *Z* corpus. Best run reported for each system.

System	Accuracy	ElhPolar [46]	SOCAL [1]	iSOL [51]	SSL [47]	Own	DAL [52]	LYSA [48]	ML Senticon [53]	Semeval 2013 [54]	MPQA [49]	HGI [50]
LIF	0.726	✓			✓			✓			✓	✓
ELiRF	0.725				✓				✓	✓		
GTI-GRAD	0.695		✓	✓		✓	✓					
GSI (aspect)	0.691	✓	✓	✓	✓				✓			
DE:Soft $\vec{w}_s$ (us)	0.677											
LYS	0.664											
DLSI	0.655					✓						
SINAI-DW2Vec	0.619											
INGEOTEC	0.613											

Additionally, Table 9 gives a brief description of the tools used by the best methods for classifying polarity tweets on the TASS task. The first column after accuracy shows the maximum number of n-gram features being used. In our work, we used only bag of words, which is equivalent to using unigrams. We are not using a Named-Entity-Recognizer module or NLP techniques (such as lemmatization, using parts-of-speech tags, etc.). We do not handle negation with any particular method. Other works use feature augmenting methods that are based on deep learning (Word2Vec [55], Doc2Vec [56], GloVe [57]), distributional methods (LDA [58], LSI [59]), or other feature weighing methods (TF·IDF [60]). In this work, none of these was used.

The last column of Table 9 shows a very compact survey of the classifiers used by each system. Most works use Support Vector Machines (SVM). The first system (LIF) uses a ensemble of SVM, and Convolutional networks with skipgrams, bag of words, and vectors obtained from GloVe [57]. These results are fed to an SVM classifier. ELiRF, the second best system combines the output of several SVM classifiers with different parameters, and then this information is classified in cascade with another SVM classifier.

Table 9. Classifiers used for systems in Table 8. (LR = Logistic Regression, SVM = Support Vector Machines, ME = MaxEnt, SG = SkipGrams).

System	Accuracy	max n-gram	NER	NLP	Negation	Word2Vec	Doc2Vec	GloVe	TF·IDF	LDA	LSI	Classifier
LIF	0.726	1		✓		✓		✓			✓	(SVM SG Cbow)→SVM
ELiRF	0.725	1		✓					✓			SVM (+ SVM)
GTI-GRAD	0.695	2		✓								LR
GSI (aspect)	0.691	1	✓	✓	✓							SVM
DE:Soft $\vec{w}_s$ (us)	0.677	1										DE: (NB, LR, SVM)
LYS	0.664	1		✓								Logistic regression L2-LG
DLSI	0.655	2		✓								SVM
SINAI-DW2Vec	0.619	1				✓	✓					SVM
INGEOTEC	0.613	5							✓	✓	✓	SVM

5.4. Discussion

The Differential Evolution strategy for optimizing the weights in a soft-voting ensemble was able to overcome performance of the individual classifiers. As expected, in the *E* set performance was better for the soft voting scheme, compared with hard weighting. Specifically, this latter achieves 66.57% accuracy, while the best weights obtained by Differential Evolution reach 67.71%.

Additionally, two different ways of partitioning information for finding the best weights were explored. One was based on random k-folds, and other on stratified k-folding. Stratified k-folds tend to improve the final classification. The latter strategy had better performance on the *E* set (66.19% vs. 65.61%), and the weight vector *Soft* $\vec{w}_s$ calculated on these folds slightly contributed to obtain a better classification on the Z corpus (67.71% vs. 67.68%). In both folding strategies, the soft weighting always outperformed the hard weighting scheme.

We experimented with the InterTASS corpus of 2018 (Spanish) in order to test our solution with a different corpus [61]. We applied the DE:Soft $\vec{w}_s$ method without recalculating weights. The results are shown in Table 10. From this table, it can be seen that despite the full process of adjusting weights was not carried out, our method outperformed some of the neural-network-based methods (retuyt-cnn).

We have calculated the statistical significance of our experiments while using the STAC platform [62] considering the different results we obtained separately with each classifier, hard weights, and soft weights. With the Shapiro–Wilk test [63], we obtained that the null hypothesis is rejected with a level of significance of 0.093, while for the Kolmogorov–Smirnov test [64], it is rejected with $\rho < 0.001$.

Table 10. Results of our proposed method with the InterTASS ES corpus, as compared with top results [61].

System	M. F1	Acc.	System	M. F1	Acc.
elirf-es-run-1	0.503	0.612	atalaya-lr-50-2-roc	0.455	0.595
retuyt-lstm-es-1	0.499	0.549	ingeotec-run1	0.445	0.530
retuyt-combined-es	0.491	0.602	atalaya-svm-50-2	0.431	0.583
atalaya-ubav3-100-3-syn	0.476	0.544	itainnova-cl-base	0.383	0.433
DE:Soft $\vec{w}_s$	0.461	0.585	itainnova-cl-proc1	0.320	0.395
retuyt-cnn-es-1	0.458	0.592			

6. Conclusions

In this paper, we presented a method to optimize weights for a classification ensemble. When compared with other methods, DE:Soft $\vec{w}_s$ is able to obtain state of the art accuracy, given that no external resources are being used. As a future work, it would be interesting to assess the effect of using our proposed method along with external resources in order to further improve scores for this task.

In general, this proposal could be used for problems where training data are relatively small when compared with the amounts required for other state of the art methods, such as deep learning. Automatic optimization of weights for different classifiers allows for easily adapting this method for other problems, including those with multiclass labels.

In both $\vec{w}_r$ and $\vec{w}_s$, SVM is notoriously given a predominant weight, although it is interesting to see that this is not the best overall classifier if used alone. In this case, LR would be a better choice (see Tables 5 and 6). Additionally, one of the best reported systems (GTI-GRAD) uses LR as its main classifier. This suggests a deeper by-case analysis that may enable classifiers to specialize in particular cases, along with a meta-classifier that dynamically adjusts weights for each case. Another option is to create separate classifiers per class; this is left as future work. Other improvements to the Differential Evolution algorithm, such as different ways of partitioning data, are also considered for further exploration.

Author Contributions: Conceptualization, O.J.G. and M.G.V.-C.; formal analysis, C.V.G.-M.; funding acquisition, H.G.; investigation, O.J.G.; methodology, C.V.G.-M., O.J.G., M.G.V.-C. and H.C.; software, C.V.G.-M.; supervision, O.J.G., M.G.V.-C. and H.C.; validation, O.J.G., M.G.V.-C. and H.C.; writing—original draft, C.V.G.-M., O.J.G. and H.C.; writing—review & editing, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: Authors wish to thank support of the Government of Mexico via CONACYT, SNI, and Instituto Politécnico Nacional (IPN) grants SIP 2083, SIP 20200811, SIP 20201252, and SIP 20201362, IPN-COFAA and IPN-EDI; AMPLN and SMIA.

Acknowledgments: Authors are grateful with Sotiris Kotsiantis, as well as the anonymous reviewers for their useful comments and discussion.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Taboada, M.; Brooke, J.; Tofiloski, M.; Voll, K.D.; Stede, M. Lexicon-Based Methods for Sentiment Analysis. *Comput. Linguist.* **2011**, *37*, 267–307. [CrossRef]
2. Pang, B.; Lee, L.; Vaithyanathan, S. Thumbs up? Sentiment Classification Using Machine Learning Techniques. In Proceedings of the EMNLP2002, Philadelphia, PA, USA, 6–7 July 2002; pp. 79–86.
3. Vilares, D.; Alonso, M.Á.; Gómez-Rodríguez, C. Supervised polarity classification of Spanish tweets based on linguistic knowledge. In Proceedings of the 2013 ACM Symposium on Document Engineering, Florence, Italy, 10–13 September 2013; pp. 169–172.
4. Sidorov, G.; Miranda-Jiménez, S.; Viveros-Jiménez, F.; Gelbukh, A.; Castro-Sánchez, N.; Velásquez, F.; Díaz-Rangel, I.; Suárez-Guerra, S.; Treviño, A.; Gordon, J. Advances in Artificial Intelligence. In Proceedings of the 11th Mexican International Conference on Artificial Intelligence, MICAI 2012, San Luis Potosí, Mexico, 27 October–4 November 2012; Chapter Empirical Study of Machine Learning Based Approach for Opinion Mining in Tweets. Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–14.
5. Ali, F.; El-Sappagh, S.; Islam, S.R.; Ali, A.; Attique, M.; Imran, M.; Kwak, K.S. An intelligent healthcare monitoring framework using wearable sensors and social networking data. *Future Gener. Comput. Syst.* **2020**. [CrossRef]
6. Godínez, I.R.; López-Yáñez, I.; Yáñez-Márquez, C. Classifying patterns in bioinformatics databases by using Alpha-Beta associative memories. In *Biomedical Data and Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 187–210.
7. Uriarte-Arcia, A.V.; López-Yáñez, I.; Yáñez-Márquez, C. One-hot vector hybrid associative classifier for medical data classification. *PLoS ONE* **2014**, *9*, e95715. [CrossRef] [PubMed]
8. García-Floriano, A.; Ferreira-Santiago, Á.; Camacho-Nieto, O.; Yáñez-Márquez, C. A machine learning approach to medical image classification: Detecting age-related macular degeneration in fundus images. *Comput. Electr. Eng.* **2019**, *75*, 218–229. [CrossRef]
9. Ali, F.; Kwak, D.; Khan, P.; El-Sappagh, S.; Ali, A.; Ullah, S.; Kim, K.H.; Kwak, K.S. Transportation sentiment analysis using word embedding and ontology-based topic modeling. *Knowl.-Based Syst.* **2019**, *174*, 27–42. [CrossRef]
10. Hansen, L.K.; Salamon, P. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.* **1990**, *12*, 993–1001. [CrossRef]
11. Dos Santos, E.M. Emotion classification of online news articles from the reader's perspective. In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, NSW, Australia, 9–12 December 2008; pp. 419–430.
12. Dietterich, T.G. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 1–15.
13. Villena-Román, J.; García-Morera, J.; Cumbreras, M.Á.G.; Martínez-Cámara, E.; Martín-Valdivia, M.T.; López, L.A.U. Overview of TASS 2015. In Proceedings of the TASS 2015: Workshop on Semantic Analysis at SEPLN (TASS 2015), Alicante, Spain, 15 September 2015; pp. 13–21.
14. Onan, A.; Korukoğlu, S.; Bulut, H. A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. *Expert Syst. Appl.* **2016**, *62*, 1–16. [CrossRef]
15. Saleena, N. An Ensemble Classification System for Twitter Sentiment Analysis. *Procedia Comput. Sci.* **2018**, *132*, 937–946. [CrossRef]
16. Liu, Y.; Yu, X.; Huang, J.X.; An, A. Combining integrated sampling with SVM ensembles for learning from imbalanced datasets. *Inf. Process. Manag.* **2011**, *47*, 617–631. [CrossRef]
17. Rooney, N.; Wang, H.; Taylor, P.S. An investigation into the application of ensemble learning for entailment classification. *Inf. Process. Manag.* **2014**, *50*, 87–103. [CrossRef]
18. Dashtban, M.; Balafar, M.; Suravajhala, P. Gene selection for tumor classification using a novel bio-inspired multi-objective approach. *Genomics* **2018**, *110*, 10–17. [CrossRef] [PubMed]
19. Adeli, M.; Rouat, J.; Wood, S.; Molotchnikoff, S.; Plourde, E. A Flexible Bio-Inspired Hierarchical Model for Analyzing Musical Timbre. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 875–889. [CrossRef]
20. Ali, F.; El-Sappagh, S.; Islam, S.R.; Kwak, D.; Ali, A.; Imran, M.; Kwak, K.S. A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion. *Inf. Fusion* **2020**, *63*, 208–222. [CrossRef]

21. Marie-Sainte, S.L.; Alalyani, N. Firefly Algorithm based Feature Selection for Arabic Text Classification. *J. King Saud Univ. Comput. Inf. Sci.* **2018**. [CrossRef]
22. Escalante, H.J.; García-Limón, M.A.; Morales-Reyes, A.; Graff, M.; y Gómez, M.M.; Morales, E.F.; Martínez-Carranza, J. Term-weighting learning via genetic programming for text classification. *Knowl.-Based Syst.* **2015**, *83*, 176–189. [CrossRef]
23. Onan, A.; Korukoğlu, S.; Bulut, H. A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. *Inf. Process. Manag.* **2017**, *53*, 814–833. [CrossRef]
24. Hoang, T.B.N.; Mothe, J. Location extraction from tweets. *Inf. Process. Manag.* **2018**, *54*, 129–144. [CrossRef]
25. Jan, M.Z.; Verma, B. A novel diversity measure and classifier selection approach for generating ensemble classifiers. *IEEE Access* **2019**, *7*, 156360–156373. [CrossRef]
26. López, M.; Valdivia, A.; Martínez-Cámara, E.; Luzón, M.V.; Herrera, F. E2SAM: Evolutionary ensemble of sentiment analysis methods for domain adaptation. *Inf. Sci.* **2019**, *480*, 273–286. [CrossRef]
27. Tama, B.A.; Nkenyereye, L.; Islam, S.R.; Kwak, K.S. An Enhanced Anomaly Detection in Web Traffic Using a Stack of Classifier Ensemble. *IEEE Access* **2020**, *8*, 24120–24134. [CrossRef]
28. Rim, K.; Tu, J.; Lynch, K.; Pustejovsky, J. Reproducing Neural Ensemble Classifier for Semantic Relation Extraction in Scientific Papers. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 5569–5578.
29. Ekbal, A.; Saha, S. A multiobjective simulated annealing approach for classifier ensemble: Named entity recognition in Indian languages as case studies. *Expert Syst. Appl.* **2011**, *38*, 14760–14772. [CrossRef]
30. Zhang, Y.; Zhang, H.; Cai, J.; Yang, B. A weighted voting classifier based on differential evolution. In *Abstract and Applied Analysis*; Hindawi: London, UK, 2014; Volume 2014.
31. Sahami, M.; Dumais, S.; Heckerman, D.; Horvitz, E. A Bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*; AAAI Press: Palo Alto, CA, USA, 1998; pp. 55–62.
32. Gambino, O.J.; Calvo, H. A Comparison Between Two Spanish Sentiment Lexicons in the Twitter Sentiment Analysis Task. In *Ibero-American Conference on Artificial Intelligence*; Springer: Cham, Switzerland, 2016; pp. 127–138.
33. Knerr, S.; Personnaz, L.; Dreyfus, G. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In *Neurocomputing*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 41–50.
34. Galar, M.; Fernández, A.; Barrenechea, E.; Bustince, H.; Herrera, F. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognit.* **2011**, *44*, 1761–1776. [CrossRef]
35. John, G.H.; Langley, P. Estimating continuous distributions in Bayesian classifiers. In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Montreal, QC, Canada, 18–20 August 1995; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1995; pp. 338–345.
36. Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Springer Series in Statistics: New York, NY, USA, 2001; Volume 10.
37. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of the European Conference on Machine Learning, Chemnitz, Germany, 21–23 April 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 137–142.
38. Wu, T.F.; Lin, C.J.; Weng, R.C. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.* **2004**, *5*, 975–1005.
39. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
40. Li, H.; Li, N.; Liu, K. Two-Way Differential Evolution Algorithm: A Global Optimization Algorithm in Continuous Space. In Proceedings of the 2010 Second WRI Global Congress on Intelligent Systems, Wuhan, China, 16–17 December 2010; Volume 1, pp. 55–58. [CrossRef]
41. Iwai, R.; Kato, S. Optimization in multi-modal continuous space with little globally convex using differential evolution on scattered parents. In Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, Korea, 14–17 October 2012; pp. 2002–2007. [CrossRef]
42. Li, J. Resource Planning and Scheduling of Payload for Satellite with a Discrete Binary Version of Differential Evolution. In Proceedings of the 2009 IITA International Conference on Control, Automation and Systems Engineering (Case 2009), Zhangjiajie, China, 11–12 July 2009; pp. 62–65. [CrossRef]

43. Sauer, J.G.; dos Santos Coelho, L. Discrete Differential Evolution with local search to solve the Traveling Salesman Problem: Fundamentals and case studies. In Proceedings of the 2008 7th IEEE International Conference on Cybernetic Intelligent Systems, London, UK, 9–10 September 2008; pp. 1–6. [CrossRef]
44. Mezura-Montes, E.; Velázquez-Reyes, J.; Coello Coello, C.A. A Comparative Study of Differential Evolution Variants for Global Optimization. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; ACM: New York, NY, USA, 2006; pp. 485–492. [CrossRef]
45. Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the IJCAI, Montreal, QC, Canada, 20–25 August 1995; Volume 14, pp. 1137–1145.
46. Saralegi, X.; San Vicente, I. Elhuyar at TASS 2013. In Proceedings of the XXIX Congreso de la Sociedad Española de Procesamiento de Lenguaje Natural, Workshop on Sentiment Analysis at SEPLN (TASS 2013), Madrid, Spain, 20 September 2013; pp. 143–150.
47. Perez-Rosas, V.; Banea, C.; Mihalcea, R. Learning Sentiment Lexicons in Spanish. In Proceedings of the LREC, Istanbul, Turkey, 23–25 May 2012; Volume 12, p. 73.
48. Vilares, D.; Doval, Y.; Alonso, M.A.; Gómez-Rodrıguez, C. LyS at TASS 2014: A prototype for extracting and analysing aspects from Spanish tweets. In Proceedings of the TASS workshop at SEPLN, Girona, Spain, 16–19 September 2014.
49. Deng, L.; Wiebe, J. MPQA 3.0: An entity/event-level sentiment corpus. In Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015.
50. Stone, P.J.; Dunphy, D.C.; Smith, M.S. *The General Inquirer: A Computer Approach to Content Analysis*; American Psychological Association: Washington, DC, USA, 1966.
51. Molina-González, M.D.; Martínez-Cámara, E.; Martín-Valdivia, M.T.; Perea-Ortega, J.M. Semantic orientation for polarity classification in Spanish reviews. *Expert Syst. Appl.* **2013**, *40*, 7250–7257. [CrossRef]
52. Ríos, M.G.D.; Gravano, A. Spanish DAL: A Spanish Dictionary of Affect in Language. In Proceedings of the WASSA 2013, Zhangjiajie, China, 7–10 August 2013; p. 21.
53. Cruz, F.L.; Troyano, J.A.; Pontes, B.; Ortega, F.J. ML-SentiCon: Un lexicón multilingüe de polaridades semánticas a nivel de lemas. *Proces. Leng. Nat.* **2014**, *53*, 113–120.
54. Manandhar, S.; Yuret, D. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop On Semantic Evaluation (Semeval 2013)*; Omnipress, Inc.: Madison, WI, USA, 2013; Volume 2.
55. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1188–1196.
56. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*; NeurIPS: Lake Tahoe, NV, USA, 2013; pp. 3111–3119.
57. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global Vectors for Word Representation. In Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014; Volume 14, pp. 1532–1543.
58. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
59. Landauer, T.K.; Foltz, P.W.; Laham, D. An introduction to latent semantic analysis. *Discourse Process.* **1998**, *25*, 259–284. [CrossRef]
60. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* **1972**, *28*, 11–21. [CrossRef]
61. Martínez Cámara, E.; Almeida Cruz, Y.; Díaz Galiano, M.C.; Estévez-Velarde, S.; García Cumbreras, M.Á.; García Vega, M.; Gutiérrez, Y.; Montejo Ráez, A.; Montoyo, A.; Munoz, R.; et al. *Overview of TASS 2018: Opinions, Health and Emotions*; Sun SITE Central Europe: Sevilla, Spain, 2018.
62. Rodríguez-Fdez, I.; Canosa, A.; Mucientes, M.; Bugarín, A. STAC: A web platform for the comparison of algorithms using statistical tests. In Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Istanbul, Turkey, 2–5 August 2015.

63. Shapiro, S.S.; Wilk, M.B. An analysis of variance test for normality (complete samples). *Biometrika* **1965**, *52*, 591–611. [CrossRef]
64. Stephens, M. Introduction to Kolmogorov (1933) on the empirical determination of a distribution. In *Breakthroughs in Statistics*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 93–105.