DOI: 10.15439/2019F183
ISSN 2300-5963 ACSIS, Vol. 18

Deep Learning Hyper-parameter Tuning for Sentiment Analysis in Twitter based on Evolutionary Algorithms

Nuria Rodríguez-Barroso, Antonio R. Moya, José A. Fernández, Elena Romero,
Eugenio Martínez-Cámara, Francisco Herrera
Andalusian Research Institute in Data Science and Computational Intelligence,
University of Granada,
Granada (Spain)
Email: {rbnuria,anmomar85,fernandezja,elenaromeroc}@correo.ugr.es; {emcamara,herrera}@decsai.ugr.es

***Abstract*—The state of the art in Sentiment Analysis is defined by deep learning methods, and currently the research efforts are focused on improving the encoding of underlying contextual information in a sequence of text. However, those neural networks with a higher representation capacity are increasingly more complex, which means that they have more hyper-parameters that have to be defined by hand. We argue that the setting of hyper-parameters may be defined as an optimisation task, we thus claim that evolutionary algorithms may be used to the optimisation of the hyper-parameters of a deep learning method. We propose the use of the evolutionary algorithm SHADE for the optimisation of the configuration of a deep learning model for the task of sentiment analysis in Twitter. We evaluate our proposal in a corpus of Spanish tweets, and the results show that the hyper-parameters found by the evolutionary algorithm enhance the performance of the deep learning method.**

I. Introduction

OPINIONS, sentiments, experiences, private states, broadly speaking subjective information, are continuously posted on micro-blogging sites as Twitter. The processing of this kind of information is crucial for other users and for any kind of organisation, because it offers a valuable source of knowledge to understand the perspectives of users on topics of interest, which eases the process of making decisions [1]. Sentiment Analysis (SA) is the task centred on labelling the opinion meaning of a text, and it is defined as the computational treatment of opinions, sentiments and subjectivity in texts [2].

Since the use of language in Twitter has its own characteristics that make it different from the use of language in formal genre of writing, specific computational methods have to be developed [3]. The main contributions to the processing of the sentiment of tweets can be found in the respective tasks of the workshops SemEval[1] for the English language and TASS[2] for the Spanish language. The state of the art on those workshops has evolved from the use of linear classification systems grounded in the use of a big bunch of hand-crafted linguistic features [4], [5] to the use of deep learning methods without the need in most of the cases of hand-crafted features [6], [7].

Besides the strong results of deep learning methods in SA in Twitter, we stress out that those deep learning methods has reduced the need of feature engineering, because they are based on the use of unsupervised pre-train features, which the most used are vectors of word embeddings. Deep learning methods depend on the configuration of some parameters that are known as hyper-parameters, such as the number of output units of each neural layer or the dropout rate. Those hyper-parameters must be defined by hand, hence the positive reduction of the effort in the designing of features has been changed to the effort of setting the right hyper-parameters value. The current trend in the development of neural networks for SA is to attempt to encode as much contextual information as possible, which is the aim, for instance, of the self-attentive networks [8] and memory networks [9]. The high complexity of those deep learning architectures entails to define a higher number of hyper-parameters, which means that their configuration would not be an easy task.

We define the process of hyper-parameter setting as an optimisation task, because the optimisation of the value of the hyper-parameters allows to optimise the performance of the neural network. In this paper we thus claim that the use of an optimisation method, as an Evolutionary Algorithm [10], may find out the right hyper-parameters values and consequently optimise the performance of a neural network. We propose the use of the evolutionary algorithm SHADE [11] for optimising the hyper-parameters of a self-attentive neural network for the task of SA in Twitter.

We evaluate our proposal in the task of SA in Twitter in Spanish, and we used the Spanish set of the corpus InterTASS [12]. We define as a baseline model our self-attentive neural network with a set of hyper-parameters values defined by hand, and we compare its performance with the optimised version of the neural model. Likewise, we compare the performance of our proposal with the results reached in the TASS 2018

[1]https://aclweb.org/aclwiki/SemEval_Portal
[2]http://www.sepln.org/workshops/tass/

competition,[3] and we show how our proposal without any external knowledge reaches a similar performance than the highest ranked systems in the competition. Moreover, we show how our evolutionary proposal has the ability to improve the learning of minority classes in a imbalanced dataset, as the InterTASS corpus is, and reduces the complexity of the neural model. Although we evaluated our proposal in an imbalanced dataset, we did not conduct any standard data augmentation technique that are usually performed for enhancing the performance of deep learning methods [13], because our aim is to evaluate our claim without the influence of any data preprocessing method.

The reminder of this paper is organised as what follows: Section II exposes some related works to SA in Twitter and hyper-parameter learning. Subsequently, Section III presents our deep learning model for SA in Twitter, which is optimised by an evolutionary algorithm that is detailed in Section IV. Sections V and VI are focused on the description of the experimental set up and the analysis of the results. Finally, Section VII presents the conclusions of our work.

II. Related Works

We propose the automatically learning of the hyper-parameters of a deep learning method in order to tackle the task of SA in Twitter. Accordingly, Section II-A describe some works related to SA, and Section II-B is focused on the task of neural networks hyper-parameters learning.

A. Sentiment Analysis in Twitter

Since the first days of Twitter, this microblogging site has attracted the attention of the research community, although the first works were closer to social sciences [14] than computer science, as well as to the concept of the electronic word of mouth [15]. However, as the popularity of Twitter was increasing, it was becoming in a communication tool in which users exchange their private states, or in other words their experiences, sentiments and opinions.

The first works on SA in Twitter were similar to the first ones in regular texts [16], [17], they were focused on the study of how to represent the opinion meaning of texts and the comparison of linear machine learning classification algorithms. In [18], the first corpus of SA in Twitter is described, and the authors evaluated the performance of three linear classification methods with three different feature vector representations approaches. The following works centred the efforts on feature engineering, broadly speaking, on the use of linguistic features and external knowledge for the representation of the opinion meaning of tweets. For instance, in [19] the tweets were represented with a combination of weighted unigrams and features generated from a sentiment lexicon. Similarly, in [20] the authors used a list of subjetive hashtags besides the use of a sentiment lexicon and unigrams to classify the polarity of tweets from different topics. The use of sentiment external knowledge is essential in [21], in which the authors first represented the tweets as bag of unigrams and bigrams, and each unigram and bigram is represented as a vector of sentiment values aggregated from several sentiment lexicons.

The classification of the polarity of tweets was also used to the prediction of future events, such as the outcome of elections [22], [23]. Likewise, in [24], the authors use the classification of the opinion to predict the evolution of stock markets. As the previous works, the method are based on the representation of the tweets with a great bunch of features and the use of linear classifiers.

Besides the strong results of deep learning methods in Twitter SA, they allow to extremely reduce the efforts in feature engineering and in the use of external knowledge. However, this is caused by the representation of the input sequences of text, in this case, tweets, with unsupervised pre-trained feature vectors. Those feature vectors are known as word embedding that represent the meaning of each word, and they are based on the distributional semantics hypothesis. Accordingly, deep leaning methods allow to reach strong results with a low designing effort. For instance, in [25], the authors classify the polarity and the language of tweets with a convolutional neural network (CNN). Likewise, the straightforward neural network described in [26] also reached good results in SA in Twitter in Spanish. Other example of the use of deep learning methods for SA in texts different from English can be read in [27]. However, in some cases, the enhancing of the performance of polarity classification in Twitter forces to use deeper and more complex deep learning methods. In [28], the authors propose the combination of a Long-Short Term Memory (LSTM) Recurrent Neural network (RNN) layer and CNN layer for polarity classification of tweets written in English.

B. Hyper-parameters Learning

The trend in SA in Twitter is the addition of more encoding layers (CNN, LSTM), and other kind of mechanisms to increase the capacity of the network to represent the contextual information of the input sequence of text. Those layers depend on a set of configuration parameters or hyper-parameters, which their right definition is essential for the global performance of the neural network. Moreover, regularisation layers, as Dropout or penalty rates for the loss function, are key elements of the architecture of neural networks in order to avoid the over-fitting. Consequently, the design of a neural network required of an effort of selecting the right hyper-parameters for each of the layers of the architecture. Therefore, the feature engineering effort has evolved to hyper-parameter engineering.

The definition of the right hyper-parameters is not an easy task, and there is not any rule of thumb to do it. However, there exist some strategies to address it, as well as, some computational approaches, which we indicate as what follows:

1) Brute force. It consists in the exhaustive evaluation of all possible values of all the hyper-parameters, which is not feasible because of limitation of time and computational resources.

[3]http://www.sepln.org/workshops/tass/2018/

2) Grid search. It is a brute force approach constrained by a pre-defined set of hyper-parameters values. This is a feasible strategy because the number of evaluations is lower in comparison with the brute force, and it allows to reach good results as show in [29]. However, the hyper-parameters values must be defined by hand.
3) Random search. In [30] is shown that the random search of the values of the hyper-parameters allows the neural network to reach good results. However, the random search cannot assure to find out the values that optimise the performance of the network.
4) Bayesian approximations [31]. The positive side of this strategies is that they do not have to completely run the neural network to optimise it, because they are grounded in a approximation. However, the complexity of those methods make them close to be unfeasible and difficult to be parallelised.
5) Evolutionary algorithms [10]. As the bayesian approximations, these algorithms seek in the hyper-parater values search space those ones that may optimise the performance of the network. Nevertheless, the own definition of evolutionary algorithms has specific strategies for finding the right values in the search space. Moreover, these algorithms are parallelisable in contrast to bayesian approximations, indeed they are parallelisable in GPUs [32]. In [33] is described the use of the CMA-ES [34] for tuning the hyper-parameters of a neural network. In [35] is again used the CMA-ES algorithm for the otpimimisation of a neural network, but in this case for the generation of a language model. The use of evolutionary methods for hyper-parameter tuning has not ceased, and recently in [36] a new evolutionary method has been proposed with positive results.

Since evolutionary algorithms are showing a positive performance on the task of hyper-parameter optimisation, we select that strategy for our experimentation, and we propose the use of the algorigthm SHADE for the tuning of the hyper-parameters of a self-attentive neural network for the task of SA in Twitter.

III. Deep Learning model for SA

Since our aim is to show the suitability of evolutionary algorithms for tuning the value of hyper-parameters, we propose a deep neural network with several layers with the aim of encoding as much contextual information as possible, which also goes in the line of the proposals of the state of the art (see Section II-A). In the subsequent sections we describe the architecture of our neural network that is composed of three main layers: (1) encoding layer (see Section III-A), self-attention layer (see Section III-B) and classification layer (see Section III-C).

A. Encoding layer

Two kind of information may be encoded from a sequence of text: local and temporal. The local information is the underlying one from the inter-dependencies among words in a local context. On the other hand, the entire sequence of text has also their own meaning which depends on the relation of all the words. Because of these two kind of information, we define an encoding layer composed of a CNN, focused on the local information, and an RNN LSTM layer, centred on the temporal information.

a) CNN: We choose a CNN layer in order to focus on the local information motivated by its sparse interactions and the ability to combine features of a local context. CNNs get this ability by implementing the discrete convolution operator (see Equation 1).

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{1}$$

where x is the input and w the kernel. The output is sometimes referred to as the feature map of size CNN_{fm}.

The input of a CNN layer is always a grid-structured dataset. For example, the sequence of vectors $\mathbf{w} = (w_1, w_2, ...w_n)$. This layer performs the convolution function for a fixed kernel size $\mathbf{k}$. For an 1-dimensional CNN, the output is another grid-structured dataset of size $n \times \text{CNN}_{fm}$. Equation 2 summarise the definition for an 1-dimensional CNN layer:

$$\begin{aligned} \text{CNN}(\mathbf{w_{1:n}}, \mathbf{k}) &= \mathbf{y_{1:n}} \\ \mathbf{y_i} &= s(\mathbf{k}) \\ \mathbf{w_i} \in \mathbb{R}^d, \mathbf{k} &\in [1, 2, ..., n] \end{aligned} \tag{2}$$

b) Bidirectional Long-Short Term Memory: The election of RNN to capture temporal information is due to the fact that they maintain memory based on information history. These networks are defined by a non-lineal function σ applied recursively on a sequence of inputs $(w_1, w_2, ..., w_n)$. The input of σ is a state vector $\mathbf{s_{i-1}}$ and an element of the sequence input $\mathbf{w_i}$. The output of the non-lineal function σ is a new state vector $\mathbf{s_i}$, which is transformed to the output vector y_i by a deterministic function O. Equation 3 summarise the definition:

$$\begin{aligned} \text{RNN}(\mathbf{w_{1:n}}, \mathbf{s_0}) &= \mathbf{y_{1:n}} \\ \mathbf{y_i} &= O(\mathbf{s_i}) \\ \mathbf{s_i} &= R(\mathbf{w_i}, \mathbf{s_{i-1}}); \\ \mathbf{w_i} \in \mathbb{R}^d, \mathbf{s_i} &\in \mathbb{R}^{f(h_{lstm})}, \mathbf{y_i} \in \mathbb{R}^{h_{lstm}} \end{aligned} \tag{3}$$

LSTM is a gating-based architecture of RNN that uses several gates in order to solve the gradient vanishing (or exploding) problem of RNN. However, LSTM still has a limitation, the recurrence is only implemented in one direction (from left to right). Nevertheless, the meaning of each word depends on their surrounding context words, broadly speaking, the words in their left and right. Accordingly, we use a bidirectional LSTM (biLSTM). These networks consist in two consecutive LSTM layers, each one in one direction (forward (LSTM^f) and backward (LSTM^b)), encoding the full context information. We formally define biLSTM in Equation 4.

$$\text{biLSTM}(\mathbf{w_{1:n}}) = [\text{LSTM}^f(\mathbf{w_{1:n}}, s_0^f), \text{LSTM}^b(\mathbf{w_{1:n}}, s_0^b)] \tag{4}$$

B. Self-Attention mechanism

The aim of attention mechanisms is to give the neural network the capacity of selecting what to learn from the input data, as humans do. Attention mechanisms have become an essential part of sequence modelling in a wide range of tasks. They are commonly used in conjunction with a RNN.

The attention mechanism in NLP tasks allow to learn what words are the most salient for the global meaning of a sequence of text, but it does not take into account the dependencies that each word has with the others. Self-Attention mechanism [37] calculates the relation of each word with the others, hence it uses more information in order to identify the most salient words. Since Self-Attention allows to use more contextual information of the input data, we chose it in order to automatically learn the set of more prominent words for the polarity classification of the input tweets.

The input of the attention mechanism is a matrix of features, in our case the output of a dense layer inmediatly after the biLSTM layer, $H = (\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_n})$ where $h_i \in \mathbb{R}^d$. This mechanism aims at selecting the best linear combination of the n hidden vectors in H. The output of the attention layer is a vector of weights $\mathbf{a}$, which are calculated according to Equation 5.

$$\mathbf{a} = \text{sigmoid}(\mathbf{w_{s2}}\tanh(\mathbf{W_{s1}}H^T)) \tag{5}$$

where $\mathbf{W_{s1}}$ is a matrix of size $c \times d$ and $\mathbf{w_{s2}}$ a vector of size c, with c arbitrary fixed in $[1, n]$ (usually equal to n). The *sigmoid* function ensures that the output weights are in $[0, 1]$.

The output or the attention mechanism has to be combined with the processing pipeline in order to select the most salient words from the input. Accordingly, the output of the attention layer is added up to the output of the dense layer that is inmediately after the biLSTM layer.

An extension of the mechanism that performs multiple hops of attention can be used. It is enough to replace the vector $\mathbf{w_{s2}}$ with a matrix $\mathbf{W_{s2}}$ of dimension $r \times n$, where r is the number of weighted outputs we want to generate.

$$\mathbf{A} = \text{sigmoid}(\mathbf{W_{s2}}\tanh(\mathbf{W_{s1}}H^T)) \tag{6}$$

Finally, the mechanism encodes the weighted sums by multiplying the matrix $\mathbf{A}$ and the matrix of features $\mathbf{H}$, resulting the matrix $\mathbf{M} = \mathbf{AH}$. To ensure the matrix $\mathbf{M}$ does not suffer from redundancy problems, the mechanism uses a penalisation term in order to encourage the diversity of the weighted sums across different hops of attention.

C. Classification layer

The classification starts with the tokenization of the sequence of input text (n). The meaning of each word is represented with its corresponding word embedding vector, which is looked up in a set of pre-trained word embeddings vector. Accordingly, the output of the input layer is $I_{n \times d}$.

The output of the input layer is processed by the encoding layer. First, a CNN layer of kernel 2 with feature map of size CNN$_{fm}$. Subsequently, we use an one-dimensional maxpooling operation with pool size as two using padding in order to keep the sentence size. Likewise, we add a dropout layer with rate dr^1 after the maxpooling layer.

After the convolution, we use a biLSTM layer with h_{lstm} hidden units. We use the L2 kernel regulariser with rate L_2r^1 in each LSTM layer. After that, we reduce the dimension of the output of the biLSTM with a dense layer with h^1 hidden units.

We apply the self-attention mechanism at this point in order to capture the relevance of each word with the generated features. We merge the results of the attention mechanism with the previous output by an addition. We apply a fully-connected layer with output size $n \times h^1$.

$$\begin{aligned}
\text{sigmoid}(y^9_{\mathbf{n}\times\mathbf{h_2}}) &= \mathbf{pred_4} \\
\text{Dropout}(y^8_{\mathbf{n}\times\mathbf{h_1}}, d^3_r) &= y^9_{\mathbf{n}\times\mathbf{h_2}} \\
\text{Dense}(y^7_{\mathbf{n}\times\mathbf{h_1}}) &= y^8_{\mathbf{n}\times\mathbf{h_2}} \\
\text{Attention}(y^6_{n\times h_1}) &= y^7_{\mathbf{n}\times\mathbf{h_1}} \\
\text{Dropout}(y^5_{\mathbf{n}\times\mathbf{h_1}}, d^2_r) &= y^6_{\mathbf{n}\times\mathbf{h_1}} \\
\text{Dense}(y^4_{n\times 2h_{lstm}}) &= y^5_{\mathbf{n}\times\mathbf{h_1}} \\
\text{biLSTM}(y^3_{n\times \text{CNN}_{fm}}) &= y^4_{\mathbf{n}\times\mathbf{2h_{lstm}}} \\
\text{Dropout}(y^2_{n\times \text{CNN}_{fm}}, d^1_r) &= y^3_{n\times CNN_{fm}} \\
\text{MaxPooling}(y^1_{n\times \text{CNN}_{fm}}) &= y^2_{n\times CNN_{fm}} \\
\text{CNN}(\mathbf{we_{n\times d}}, 2) &= y^1_{\mathbf{n}\times\text{CNN}_{\mathbf{fm}}}
\end{aligned} \tag{7}$$

Finally, we use two dense layers. The first one with h^2 hidden units, and the second one matching the number of labels with *sigmoid* activation function.[4]

We show the architecture of our model in Figure 1. Furthermore, we summarise the formal definition in Equation 7.

IV. Evolutionary optimisation of hyper-parameters

The increasing complexity of deep learning models in SA are becoming harder the right configuration of the hyper-parameter values of the neural networks. In this section we describe our proposal grounded in the use of a evolutionary algorithm for tuning the hyper-parameters of a deep learning method.

Evolutionary algorithms (EA) are based on the natural evolution of species, which allows to keep promising individuals, that is, best solutions to our problem. The main steps of these types of algorithms are: (1) Initialisation of a random population, (2) evaluation of the population, (3) selection of the parents, (4) crossover and mutation and (5) replacement of

[4]We decided to use the sigmoid function instead of softmax as activation function of the last layer because the sigmoid function reached better results in previous experiments.

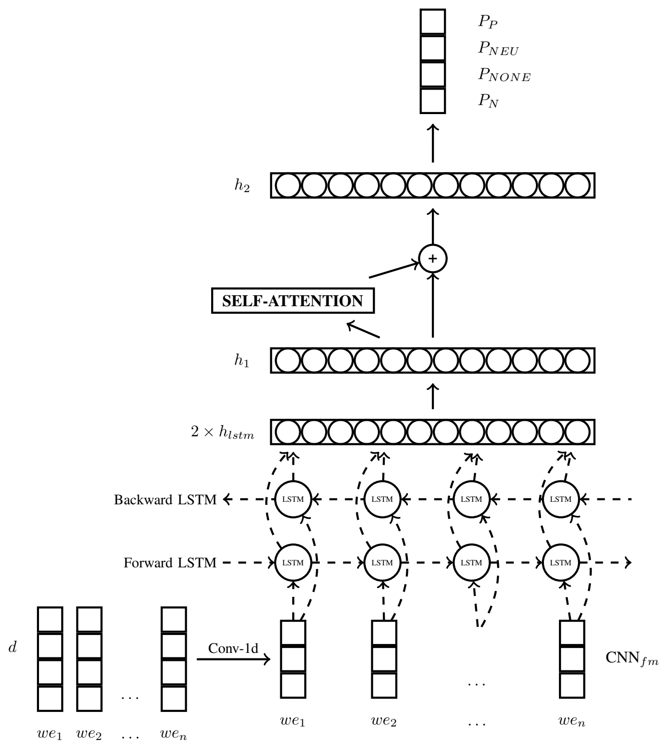

Fig. 1. Architecture of the deep learning model.

the current population by a new generation with individuals selected between parents and offspring. The algorithm is iteratively run until the stopping condition is satisfied.

According to the CEC competition,[5] L-SHADE [38] is in the state of the art of EAs. This algorithm is able to work with large populations of individuals, and it has a mechanism to linearly reduce the population size. The population size of our problem is not large, hence the L-SHADE algorithm is not the most suitable. Consequently, we used the SHADE algorithm [11], which lacks of the linear population size reduction method.

We define as the population of EA the hyper-parameters of some of the layers of our deep learning configuration proposed, specifically: (1) dropout rate (dr^1, dr^2 and dr^3), (2) regularisation rate (L2) (L_2r^1 and L_2r^2) and (3) number of units in the network layers (CNN$_{fm}$, h_{LSTM}, h^1 and h^2). Thus, each individual shows a candidate combination of these network hyper-parameters. Figure 2 shows an example of an individual of the population.

110	0.5	64	64	0.0005	0.5	32	0.002	0.5
CNN$_{fm}$	dr^1	h_{LSTM}	h^1	L_2r^1	dr^2	h^2	L_2r^2	dr^3

Fig. 2. Example of an individual of the population.

[5]http://www3.ntu.edu.sg/home/EPNSugan/index_files/cec-benchmarking.htm

Differential evolution (DE) evolves a population of NP D-dimensional individual vectors towards the global optimum. We represent as $x_{i,G}$ the individual i at generation G and called it target vector. The initial population should ideally cover the entire search space by randomly distributing each parameter of an individual vector with uniform distribution between prescribed upper and lower parameters bounds.

The main operations of the SHADE algorithm are described as what follows.

A. Mutation Operation

Trying to generate diversity in our population, we create a new population which will be crossed in a next step with the current individuals. We define the next mutation operation for this task. For each target vector $x_{i,G}$ we generate a mutant vector $v_{i,G}$. We use the mutation strategy `DE/current-to-best/1`, which generates a mutant vector using differences between the target vector and the best individual and other random individuals of the current population (see Equation 8).

$$v_{i,G} = x_{i,G} + F(x_{best,G} - x_{i,G}) + F(x_{r_1^i,G} - x_{r_2^i,G}) \quad (8)$$

where the subscripts r_1 and r_2 are random and mutually different integers generated in the range $[1, NP]$, F is a positive factor for scaling differential vectors and $x_{best,G}$ is the individual vector with best fitness value in the population at generation G.

B. Crossover Operation

The idea of diversity is needed for seeking the solution. However, we need a balance between exploration of the search-space and exploitation of the current population. Thus, after the mutation operation, crossover operation is used on the individual $x_{i,G}$ and its corresponding mutant vector $v_{i,G}$ to generate a trial vector $u_{i,G}$, which could be seen as a new individual that allows to keep both properties noted before. For each parameter of the trial vector, we choose between the corresponding parameter of $x_{i,G}$ or $v_{i,G}$ depending on crossover rate (CR):

$$u_{i_j,G} = \begin{cases} v_{i_j,G} & \text{j = K or } rand_{i_j}[0,1] \leq CR \\ x_{i_j,G} & \text{otherwise} \end{cases}$$

where CR is a value within the range [0,1), K is a randomly chosen integer in the range $[1, D]$. To ensure that the trial vector $u_{i,G}$ will differ from its corresponding vector $x_{i,G}$ we add the condition $j = K$. As result, we obtain the off-spring population.

C. Selection Operation

It selects the best individuals from the population in order to generate a better offspring. The objective function value of each trial vector is compared to its corresponding target vector in the current population. If the trial vector improve the objective function value, the trial vector will replace the target vector for the next generation. Otherwise, the target vector will remain in the population for the next generation. The selection operation is grounded in the Equation 9

$$x_{i,G+1} = \begin{cases} x_{i,G} & f(x_{i,G}) \leq f(u_{i,G}) \\ u_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

D. Parameters self-adaption

The performance of the original DE algorithm is highly dependent on the parameters settings (CR and F). It may require a huge amount of computation time. SHADE can automatically adapt the parameters settings during evolution. For this purpose, SHADE introduce success and failure memories to store different values of F and CR within a fixed number of previous generations, hereby named learning period (LP). After the initial LP generations, the probabilities of choosing different parameters values is given by Equation 10.

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \tag{10}$$

where K is the total number of values that we can choose, and $S_{k,G}$ represents the success rate of the trial vectors generated by the k_{th} value and successfully entering the next generation within the previous LP generations with respect to generation G. Equation 11 defines $S_{k,G}$.

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,G}}{\sum_{g=G-LP}^{G-1} ns_{k,G} + \sum_{g=G-LP}^{G-1} nf_{k,G}} + \epsilon \tag{11}$$

where ns and nf are the successful and failures for a certain value in a certain generation.

E. Restart mechanism

When an iteration of the evolution is performed, it is possible that our solutions may get stuck in a local search space. Accordingly, we propose to use a restart mechanism in order to avoid to reach a local optimum. When many generations pass without an improvement of the best solution, we opt to restart the population, keeping the best so far. It allows to move our search to new points of the search-space and and test new solutions that could not be evaluated without this approach.

F. Objective function

We need an objective function for evaluating the candidate solutions and select best ones. For that, we design a fitness function based on the following requirements:

- We fixed a model (same for each individual).
- Given an individual, each individual parameter is placed adequately in this model.
- We train the model for those values.
- We get predictions and calculate **Macro-F1** (see Section V-C) over a training set.
- With the purpose of minimising the previous value, we use $1 - MacroF1$ as fitness function for this individual.

Thus, we lead on the evolution of the population towards to the solution with the best results for **Macro-F1** over the evaluation set.

V. Experiments

In this section we show the experiments carried out with our proposed deep learning hyper-parameter tuning based on an EA. We first introduce the dataset used in our experiments, InterTASS Corpus (see Section V-A). Subsequently, we detail the set of pre-train vector of word embeddings used to represent the input tweets (see Section V-B). Then, we compare the results obtained with our method to the ones given by the neural network using the hyper-parameters defined by hand. We also compare our models to the highest ranked model in Task 1 of TASS-2018 Workshop (see Section V-C).

A. InterTASS Corpus

The InterTASS Corpus was presented in the *TASS-2018 Workshop* for Task 1, polarity classification at tweet level. The sentiment of the tweets of the corpus are annotated in a scale of 4 levels of polarity intensity: positive (P), Negative (N), neutral (NEU) and no opinion (NONE). The InterTASS Corpus is divided into three datasets: Training (1008 tweets), Development (506 tweets) and Test (1899 tweets). The distribution among the different labels is shown in Table I.

TABLE I
Size of each class in each subset of the InterTASS corpus.

	Training	Dev.	Test
P	317	156	642
NEU	133	69	216
N	416	219	767
NONE	138	62	274

According to Table I, the size of the training set is not large, and the distribution of the classes is not balanced, because there is a big difference among the classes P and N and the classes N and NONE. Thus, this two facts will make harder the classification and the optimisation of the model. According to [39], the imbalanced of the data in machine learning may be smoothed by oversampling the minority class. Hence, we slightly reduced the imbalance of the corpus conducting an oversampling method, which consisted in duplicating the instances from the two minority classes. The distribution among the classes in the training set after the oversampling is shown in Table II.

TABLE II
Distribution of labels after oversampling the minority labels.

	No Oversample	Oversample
P	317	317
NEU	133	266
N	416	416
NONE	138	276

As we can see from the distribution of classes in table II, NEU and NONE are still the minority classes. However, the difference with the majority classes has decreased. Establishing the percentage of oversampling is a difficult task, since the amount of data from the minority classes must be increased without losing the representativity of the dataset. For that reason, we choose low oversampling ratios.

B. Word Embeddings

As we indicated in Section III-A, each word is represented with a vector from a set of pre-trained set of word embeddings. Since the aim is to classify data from Twitter, we trained the embeddings on a set of tweets written in Spanish[6] and using the FastText method [40]. This set of embeddings have a vector representation for some meta-tokens of Twitter, such as: mentions (@user), emojis[7] and for the hashtags of the embeddings training set.

The dimension of the vectors given by these embeddings is $d = 100$. Since we used TensorFlow for developing our deep learning method, we had to define a fixed size for the input of the neural network, and to used a zero-padding approach for those tweets shorter and larger to the pre-defined size. The longest tweet in the training set contained 35 tokens, so shorter tweets were filled using padding and truncated in the case of finding a longer tweet in the validation or test set. As the length of the embeddings is 100 and the length of the tweets was set to be 35, the input of the model is a 35×100 matrix.

C. Results

In this section we present the results of the evaluation, that was conducted using the standard evaluation measures in text classification tasks, specifically: F1 score and Accuracy. The F1 is the harmonic mean of the Precision and the Recall, and it provides a trade off among them. Since we are facing up a multi-class classification problem, we used the macro-average version of F1.

We define a set of default values for the hyper-parameters of our deep learning model. Those values were used to configure out the model that was not optimised by the EA algorithm, and they were also used as the initial values of the neural model that was optimised. Table III shows the default hyper-parameters values, which are similar of other deep learning models from the state of the art in SA in Twitter. Likewise, Table III shows the hyper-parameter values returned by the EA algorithm. Some of those values are far from the default ones, and we highlight the value for the second layer of dropout (dr^2) that is a very uncommon rate value for a dropout layer, which is usually about 0.5. We also stress out the value for the output units of the biLSTM layer, which is far away from the default value, and it significantly reduces the number of trained parameters of the neural network. Likewise, the size of the output dimension of the CNN was also shortened. Consequently, the SHADE algorithm also optimised the complexity of the neural network.

[6]The tweets to train the set of word embeddings are totally different from the tweets of the training set of InterTASS corpus.

[7]Thre is an embedding vector for each emoji.

TABLE III
HYPER-PARAMETER VALUE BEFORE AND AFTER USING EVOLUTIONARY ALGORITHM.

	Starting point	After tuning
CNN_{fm}	128	108
h_{lstm}	64	28
h^1	32	21
h^2	16	21
dr^1	0.35	0.471887870
dr^2	0.35	0.0706515485
dr^3	0.5	0.509543630
$L_2\ r^1$	0.0001	0.000410222222
$L_2\ r^2$	0.001	0.00173633267

The objective function of the SHADE algorithm was configured out to optimise the F1 score on the validation set. Table IV shows the results reached by the non-optimised deep learning method, our baseline, and the optimised model.

TABLE IV
RESULTS OBTAINED WITH THE DIFFERENT MODELS.

	Macro-F1	Accuracy
Baseline Model	0.41870	0.60242
Our proposal	0.48352	0.56398

According to Table IV, there is an improvement of more than 6 points in the Macro-F1 after tuning the hyper-parameters with the SHADE algorithm. The use of evolutionary algorithms to tune the hyper-parameters proves to be successful as it improves the Macro-F1 of the initial model. However, the value of the Accuracy in the optimised model is slightly lower than the one reached by the baseline. This is an expected behaviour because of the imbalanced nature of the data. Although the total number of true positives in all the classes is slightly lower in the optimised model, the trade off of correctly tweets classified in all the classes is better in the optimised model as we show in Section VI.

Finally, we use McNemar statistical test [41] in order to study if there are significant differences among the non-optimised model and the optimised one. The test returned that our proposal is significantly better with a p-value of 0.001 ($p < 0.001$).

Table V shows the position of our proposal in the competition TASS 2018. The first two ranked models elirf-es-run-1 [7] and retuyt-lstm-es-1 [42] are based on deep learning methods, but both of them are grounded in the use of data augmentation techniques. Moreover, the elirf-es-run-1 system also uses external knowledge, such as lists of opinion bearing words in order to enrich with sentiment information of the

TABLE V
RESULTS OF *InterTass-2018 Workshop task 1.*

	Macro-F1	Accuracy
elirf-es-run-1	0.503	0.612
retuyt-lstm-es-1	0.499	0.549
Our proposal	0.484	0.564
atalaya-ubav3-100-3-syn	0.476	0.544
retuyt-svm-es-2	0.473	0.584

vectors of word embeddings. In contrast, our proposal does not use any amount of external knowledge, and it only uses to train the model the training data. Furthermore, we only duplicated the instances of the minimum classes, which is a less sophisticated data augmentation technique than the one used in retuyt-lstm-es-1. Nevertheless, the results of our optimised proposal are close to the first ones.

VI. Analysis

In this section we study the performance of the experiments explained in the previous section in a more exhaustive way.

In order to explain the results obtained in table IV, we compute the F1 score for each class. We aim to analyse the increases and decreases of the F1 score for each class, which shows the behaviour of the evolutionary algorithm in the task of optimising the F1 score. The F1 for each class can be found in Table VI.

TABLE VI
Results obtained with the different models showing F1 by class.

	Baseline Model	Our Model
$F1_P$	0.6691	0.6485
$F1_{NEU}$	0	0.1909
$F1_N$	0.6886	0.67452
$F1_{NONE}$	0.3171	0.4202

Regarding the base model, we see a low performance of the two minority labels (NEU and NONE). We highlight that the base model does not classify any tweet as NEU, which means that the model is not be able to learn anything about this label. Likewise, the performance on the NONE label is also reduced, which means that the base model is over-fitted to the labels with more instances. The main improvement of the optimised model is that it improves the performance of the classification in the two minority classes, which improves the performance of the overall system. Consequently, the macro-F1 score of the optimised model is higher, as we indicated in Section V-C.

To go further into this analysis, we examine the behaviour of both models in specific tweets of the different classes. On the first place, we observe that there are some tweets of the majority classes (P and N) that the base model labels correctly and the proposed model does not. We show some of these examples in the table VII. We highlight that the proposed model misclassifies the tweets with the minority classes and, it does not misclassifies among the two majority classes (P and N).

In the same way, there are several examples of tweets of the minority classes (NEU and NONE) that the proposed model labels correctly while the base model does not. We show some examples in table VIII.

This analysis explains the behaviour of the Macro-F1 and accuracy in Table IV. The baseline model (non-optimised) labels correctly more instances but ignoring minority classes while the optimised model deals better with imbalance by giving more importance to minority classes. This illustrates the importance of choosing a good evaluation measure. Depending on the problem there are evaluation measures that are more representative than others. In our problem, the measure Macro-F1 measures the performance of the models in a more representative way since it takes into account the imbalance. Therefore, according to this measure, we can conclude that the optimised model has provided better results for the imbalanced classification problem.

VII. Conclusions

In this paper, we have stress out the difficulty of defining the right hyper-parameters of deep learning method, which makes harder as the complexity of the network increases. We claim that evolutionary algorithms may be used to optimise the value of those hyper-parameters, and we thus propose the use of the SHADE algorithm in order to optimise a self-attentive neural network. We evaluate our proposal in the task of SA in Twitter, specifically of tweets written in Spanish from the InterTASS corpus.

The results show how our optimised proposal allows to improve the performance of the global model and the performance on each of the four classes of the dataset. Likewise, the resultant configuration of the neural network has less parameters than the non-optimised, which is also positive in the sense than optimised the efficiency of the model. Therefore, we conclude that evolutionary algorithms, in our case the SHADE algorithm, are suitable for optimising the configuration of neural networks, broadly speaking, for tuning the hyper-parameter values of deep learning methods. Accordingly, this results open a research line for the meta-learning of hyper-parameters and neural networks, where there a lot of room of improvement.

As future work, we will study the performance of evolutionary algorithms for optimising the number of encoding and classification layers. Likewise, we will evaluate the model with data augmentation approaches to study the synergy between both methodologies.

Acknowledgements

This work was supported by proyect TIN2017-89517-P, by the Spanish “Ministerio de Economía y Competitividad”, the project DeepSCOP-Ayudas Fundación BBVA a Equipos de Investigación Científica en Big Data 2018 and a grant from the Fondo Europeo de Desarrollo Regional (FEDER). Eugenio Martínez Cámara was supported by the Spanish Government Programme Juan de la Cierva Formación (FJCI-2016-28353).

References

[1] D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, “The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation,” *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 2, pp. 5:1–5:29, Aug. 2018. doi: 10.1145/3185045. [Online]. Available: http://doi.acm.org/10.1145/3185045

[2] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Found. Trends Inf. Retr.*, vol. 2, no. 1-2, pp. 1–135, Jan. 2008. doi: 10.1561/1500000011

TABLE VII
TWEETS OF MAJORITY CLASSES CORRECTLY LABELED BY BASE MODEL. WE SHOW THE ORIGINAL TWEET IN SPANISH AND ITS ENGLISH TRANSLATION.

Original Tweet	Translated Tweet	Label	$\text{Label}_{\text{base}}$	$\text{Label}_{\text{ev.}}$
Gracias a toda la gente que dio RT a mi mensaje de buscar clan. He conseguido ser suplente adc en DragonsZGaminG. Con ganas	Thanks to all the people who gave RT to my clan search message. I have managed to be adc substitute at DragonsZGaminG, I am looking forward to it.	P	P	NEU
Está tocando Bolbora en mi pueblo y yo en la cama con 38 de fiebre	Bolbora is playing in my city and I'm in bed with a fever of 38	P	P	NONE
@user lo sé, lo sé ... la sigo desde hace tiempo jajajaja es de mis favoritas	@user I know, I know... I've been following her for a long time jajaja she's one of my favourites	N	N	NEU
@user la experiencia en mi centro es que los docentes (en la medida de nuestras posibilidades) las llevamos a cabo habitualmente	the experience in my school is that we teachers (to the best of our ability) carry them out regularly	N	N	NONE

TABLE VIII
TWEETS OF MINORITY CLASSES CORRECTLY LABELLED BY OUR PROPOSED MODEL. WE SHOW THE ORIGINAL TWEET IN SPANISH AND ITS ENGLISH TRANSLATION.

Original Tweet	Translated Tweet	Label	$\text{Label}_{\text{base}}$	$\text{Label}_{\text{ev.}}$
Venga.. en el próximo tweet os muestro el equipo con el que haré el Modo Carrera en FIFA 17	Alright... In the next tweet I'm showing the team I'm using in the Manager Mode in FIFA 17	NONE	P	NONE
@user sí, las classicas de un día es lo que tienen	@user yes, one day's classical is what it means	NONE	P	NONE
@user mejor a 3. El lunes más primeras Impresiones	@user 3 is better. On Monday more first impressions	NONE	P	NONE
¿Me podrían recomendar alguna película antigua que les guste?	Could anyone recommend me any old film that you like?	NONE	P	NONE
@user yo solo puse las evos del eevee	@user I just put the evee's evolutions	NONE	N	NONE
Al igual solo es estrés, perdón	Maybe it is just stress, sorry @user	NONE	N	NONE
Luego lo más seguro haga periscope, alguien me va a ver?	I will surely be in live in periscope later, is anyone seeing me?	NONE	N	NONE
Estoy sola en un banco	I'm alone in a bench	NONE	N	NONE
Cada vez estoy más Chetado en el LOL	I'm getting more and more cheated in LOL	NEU	P	NEU
@user pero si ya estás pillada	@user but you are already mad	NEU	P	NEU
Soy una atrevida ay todos me lo dicen	I'm daring ay everyone tell me	NEU	P	NEU
@user pero no es lo mismo	@user but it is not the same	NEU	P	NEU
@user de aquí nace una bonita amistad Pero bueno, si la decepción es solo por el software ni tan mal! @user @user	@user a nice friendship grows from here.Anyway, if the disappointment is only because of the software, it's alright! @user @user	NEU	N	NEU
Esta noche hasta el culete	Tonight I'll be off my face	NEU	N	NEU
Mejorando esos dobles, añana más. Lástima no poder mostrar los steps de pantalla (Por ahora)	My doubles are improving, more tomorrow. It's a pity that I cannot show the screen steps (by the moment)	NEU	N	NEU
a mí me sigues y no soy guapo	You follow me and I'm not handsome	NEU	N	NEU

[3] E. Martínez-Cámara, M. T. Martín-Valdivia, L. A. Ureña López, and A. Montejo-Ráez, "Sentiment analysis in Twitter," *Natural Language Engineering*, vol. 20, no. 1, p. 1–28, 2014. doi: 10.1017/S1351324912000332

[4] S. Mohammad, S. Kiritchenko, and X. Zhu, "NRC-canada: Building the state-of-the-art in sentiment analysis of tweets," in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, Jun. 2013, pp. 321–327. [Online]. Available: https://www.aclweb.org/anthology/S13-2053

[5] L. Hurtado, F. Pla, and D. Buscaldi, "ELiRF-UPV at TASS 2015: Sentiment analysis in twitter," in *Proceedings of TASS 2015: Workshop on Sentiment Analysis at SEPLN co-located with 31st SEPLN Conference (SEPLN 2015)*. Alicante, Spain: Spanish Society for Natural Language Processing, 2015, pp. 75–79.

[6] M. Cliche, "BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017. doi: 10.18653/v1/S17-2094 pp. 573–580. [Online]. Available: https://www.aclweb.org/anthology/S17-2094

[7] H. L. González, José-Ángel and F. Pla, "ELiRF-UPV at TASS 2018: Sentiment analysis in twitter based on deep learning," in *Proceedings of TASS 2018: Workshop on Semantic Analysis at SEPLN (TASS 2018) co-located with 34nd SEPLN Conference (SEPLN 2018)*. Sevilla, Spain: Spanish Society for Natural Language Processing, 2018, pp. 37–44.

[8] A. Ambartsoumian and F. Popowich, "Self-attention: A better building block for sentiment analysis neural network classifiers," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018. doi: 10.18653/v1/P17 pp. 130–139. [Online]. Available: https://www.aclweb.org/anthology/W18-6219

[9] N. Majumder, S. Poria, A. Gelbukh, M. S. Akhtar, E. Cambria, and A. Ekbal, "IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3402–3411. [Online]. Available: https://www.aclweb.org/anthology/D18-1377

[10] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 13:1–13:35, Sep. 2015. doi: 10.1145/2792984. [Online]. Available: http://doi.acm.org/10.1145/2792984

[11] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013. doi: 10.1109/CEC.2013.6557555 pp. 71–78.

[12] M. C. Díaz-Galiano, M. A. García-Cumbreras, M. García-Vega, Y. Gutiérrez, E. M. Cámara, A. Piad-Morffis, and J. Villena-Román, "TASS 2018: The strength of deep learning in language understanding tasks," *Procesamiento del Lenguaje Natural*, vol. 62, pp. 77–84, 2019. doi: 10.26342/2019-62-9

[13] S. Tabik, D. Peralta, A. Herrera-Poyatos, and F. Herrera, "A snapshot of image pre-processing for convolutional neural networks: case study of MNIST," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 555–568, 2017.

[14] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter:

Understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, ser. WebKDD/SNA-KDD '07. New York, NY, USA: ACM, 2007. doi: 10.1145/1348549.1348556. ISBN 978-1-59593-848-0 pp. 56–65. [Online]. Available: http://doi.acm.org/10.1145/1348549.1348556

[15] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury, "Micro-blogging as online word of mouth branding," in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '09. New York, NY, USA: ACM, 2009. doi: 10.1145/1520340.1520584. ISBN 978-1-60558-247-4 pp. 3859–3864. [Online]. Available: http://doi.acm.org/10.1145/1520340.1520584

[16] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Jul. 2002. doi: 10.3115/1118693.1118704 pp. 79–86. [Online]. Available: https://www.aclweb.org/anthology/W02-1011

[17] P. Turney, "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews," in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002. doi: 10.3115/1073083.1073153 pp. 417–424. [Online]. Available: https://www.aclweb.org/anthology/P02-1053

[18] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford University, Stanford, CA, USA, Tech. Rep. CS224N Project Report, 2009.

[19] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 151–160. [Online]. Available: https://www.aclweb.org/anthology/P11-1016

[20] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu, "Combining lexicon-based and learning-based methods for twitter sentiment analysis," HP Laboratories, USA, Tech. Rep. HPL-2011-89, 2011.

[21] E. Martínez Cámara, M. A. García Cumbreras, M. T. Martín Valdivia, and L. A. Ureña López, "SINAI-EMMA: Vectors of words for sentiment analysis in twitter," in *Proceedings of TASS 2015: Workshop on Sentiment Analysis at SEPLN co-located with 31st SEPLN Conference (SEPLN 2015)*. Alicante, Spain: Spanish Society for Natural Language Processing, 2015, pp. 41–46.

[22] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment," 2010.

[23] A. Jungherr, P. Jürgens, and H. Schoen, "Why the pirate party won the german election of 2009 or the trouble with predictions: A response to tumasjan, a., sprenger, t. o., sander, p. g., & welpe, i. m. "predicting elections with twitter: What 140 characters reveal about political sentiment"," *Social Science Computer Review*, vol. 30, no. 2, pp. 229–234, 2012. doi: 10.1177/0894439311404119

[24] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1 – 8, 2011. doi: 10.1016/j.jocs.2010.12.007

[25] J. Wehrmann, W. E. Becker, and R. C. Barros, "A multi-task neural network for multilingual sentiment classification and language detection on twitter," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: ACM, 2018. doi: 10.1145/3167132.3167325. ISBN 978-1-4503-5191-1 pp. 1805–1812.

[26] F. M. Luque and J. M. Pérez, "Atalaya at TASS 2018: Sentiment analysis with tweet embeddings and data augmentation," in *Proceedings of TASS 2018: Workshop on Sentiment Analysis at SEPLN co-located with 34th SEPLN Conference (SEPLN 2018)*. Sevilla, Spain: Spanish Society for Natural Language Processing, 2018, pp. 29–35.

[27] J. Kapočiūtė-Dzikienė, R. Damaševičius, and M. Woźniak, "Sentiment analysis of lithuanian texts using traditional and deep learning approaches," *Computers*, vol. 8, no. 1, 2019. doi: 10.3390/computers8010004. [Online]. Available: https://www.mdpi.com/2073-431X/8/1/4

[28] N. Chen and P. Wang, "Advanced combined lstm-cnn model for twitter sentiment analysis," in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Nov 2018. doi: 10.1109/CCIS.2018.8691381 pp. 684–687.

[29] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. doi: 10.3115/v1/D14-1181 pp. 1746–1751. [Online]. Available: https://www.aclweb.org/anthology/D14-1181

[30] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, Feb. 2012.

[31] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959.

[32] A. Cano, A. Zafra, and S. Ventura, "Speeding up the evaluation phase of gp classification algorithms on gpus," *Soft Computing*, vol. 16, no. 2, pp. 187–202, Feb 2012. doi: 10.1007/s00500-011-0713-4

[33] I. Loshchilov and F. Hutter, "Cma-es for hyperparameter optimization of deep neural networks," in *Proceedings of ICLR 2016 - Workshop Track)*, 2014, pp. 1746–1751.

[34] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001. doi: 10.1162/106365601750190398

[35] T. Tanaka, T. Moriya, T. Shinozaki, S. Watanabe, T. Hori, and K. Duh, "Automated structure discovery and parameter tuning of neural network language model based on evolution strategy," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2016. doi: 10.1109/SLT.2016.7846334 pp. 665–671.

[36] Y. Nalçakan and T. Ensari, "Decision of neural networks hyperparameters with a population-based algorithm," in *Machine Learning, Optimization, and Data Science*, G. Nicosia, P. Pardalos, G. Giuffrida, R. Umeton, and V. Sciacca, Eds. Cham: Springer International Publishing, 2019. ISBN 978-3-030-13709-0 pp. 276–281.

[37] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *International Conference on Learning Representations (ICLR 2017)*, 2017. [Online]. Available: https://openreview.net/forum?id=BJC_jUqxe

[38] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014. doi: 10.1109/CEC.2014.6900380 pp. 1658–1665.

[39] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018.

[40] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. doi: 10.1162/tacl_a_00051. [Online]. Available: https://doi.org/10.1162/tacl_a_00051

[41] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun 1947. doi: 10.1007/BF02295996. [Online]. Available: https://doi.org/10.1007/BF02295996

[42] L. Chiruzzo and A. Rosá, "RETUYT-InCo at TASS 2018: Sentiment analysis in spanish variants using neural networks and svm," in *Proceedings of TASS 2018: Workshop on Sentiment Analysis at SEPLN co-located with 34th SEPLN Conference (SEPLN 2018)*. Sevilla, Spain: Spanish Society for Natural Language Processing, 2018, pp. 57–63.