Inteligencia Artificial, 21(62) (2018), 1-11
doi: 10.4114/intartif.vol21iss62pp1-11

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Sentiment polarity classification of tweets using an extended dictionary

Vladimir Vargas-Calderón vvargasc@unal.edu.co[1], Nelson A. Vargas Sánchez nelsona.vargass@konradlorenz.edu.co[2,3], Liliana Calderón-Benavides mcalderon@unab.edu.co[2], and Jorge E. Camargo jorgee.camargom@konradlorenz.edu.co[3]

[1]Universidad Nacional de Colombia
[2]Universidad Autónoma de Bucaramanga
[3]Fundación Universitaria Konrad Lorenz

Abstract With the purpose of classifying text based on its sentiment polarity (positive or negative), we proposed an extension of a 68,000 tweets corpus through the inclusion of word definitions from a dictionary of the Real Academia Española de la Lengua (RAE). A set of 28,000 combinations of 6 Word2Vec and support vector machine parameters were considered in order to evaluate how positively would affect the inclusion of a RAE's dictionary definitions classification performance. We found that such a corpus extension significantly improve the classification accuracy. Therefore, we conclude that the inclusion of a RAE's dictionary increases the semantic relations learned by Word2Vec allowing a better classification accuracy.
Resumen Con el propósito de clasificar texto basado en su polaridad de sentimiento (positivo o negativo), en este artículo se propone una extensión de un corpus de 68.000 tweets mediante la inclusión de palabras pertenecientes a definiciones de un diccionario de la Real Academia Española de la Lengua (RAE). Un cojunto de 28.000 combinaciones de 6 parámetros de Word2vec y Máquinas de Soporte Vectorial fueron considerados para evaluar qué tan positivamente se afecta el desempeño de la clasificación. Se encontró que al hacer la expansión propuesta se incrementan las relaciones aprendidas en Word2vec, permitiendo obtener una mayor precisión en la clasificación.

Keywords: Corpus extension, Word2Vec, Support vector machine, Sentiment analysis, Polarity, Semantics.
Palabras Clave: Extensión de corpus, Word2Vec, Máquinas de soporte vectorial, Análisis de sentimientos, Polaridad, Semántica.

1 Introduction

Sentiment analysis is one of the most challenging tasks in machine learning applied to text. One of its most active branches is text classification by sentiment polarity, which refers to the positivity or negativity of the sentiment represented by the text. There are prominent difficulties in classifying text by polarity when text is too short, as it is the case in Twitter in which text does not exceed 140 characters. To overcome these difficulties, this paper presents an extension to the TASS (sentiment analysis workshop of the Spanish society for natural language processing) General corpus [1] (hereafter just corpus). The TASS corpus consists of a set of tagged tweets by polarity, and the extension consists in including definitions from the Spanish Real Academy's dictionary (RAE) to the set of texts from which sentiment analysis algorithms learn. In this work, we compared a sentiment analysis algorithm trained with tweets with another trained with both tweets and RAE's dictionary definitions. The reason for extending the corpus with RAE's dictionary definitions is that they provide semantic relations between words. The technique of extending a corpus composed of short texts is used to expand the context. Applying this technique allows to improve text classification by polarity, as shown by the studies in [2, 3]. Additionally, some text domains grow at such fast rates that it is practically impossible to maintain the training data updated. Therefore, using related domains with tagged text is a solution that has been lately used to overcome scarce tagged data. This is called cross-domain text classification, and has shown preliminary results. Nevertheless, some procedures have been developed and tested, and the results have improved, such as the work of Wang et al. [4], where Wikipedia is used to propagate labels between two different text domains which captures common words and semantic concepts contained in the documents; as well as the work of Mouriño et al. [5], where the concept of cross-language concept matching is proposed to perform cross-language text classification, which is based on the representation of concepts using Wikipedia correspondence of concepts in different languages.

The vast majority of research has been done using text in English. In spite of scarce research using text in Spanish, there has been recent and steady progress. However, it is evident that sentiment analysis research needs more attention on Spanish corpora since Spanish is a more inflected language than English [6]. Few studies, such as the one by Ortega et al. have deepen in Spanish text domain extension [7].

Some other relevant works using Spanish corpora for sentiment analysis have been based on the polarity analysis of lexicons. This is the case of the research by Pablos et al., in which Word2Vec is used to automatically create a lexicon dictionary with associated polarities [8]. Also, Cruz et al. rank lexicons by their impact on the task of text classification by polarity [9].

Other works have tackled some specific problems in the industry, e.g. [10, 11], while some research, such as [12], have enhanced results of binary classification in sentiment analysis, as well as in topic classification using the concept of maximum entropy. Concerning informal text, including the one found on the Internet, Mosquera et al. proposed preprocessing methods to regularize and normalize such texts for a better performance in sentiment analysis tasks.

In this paper, we propose a model based on Word2Vec that allows to represent words as vectors from an extended Twitter corpus with RAE's dictionary definitions. The model detects word's polarities given their occurrences in tweets tagged with positive or negative polarities. The main goal of this study was not to obtain high accuracy in text classification by polarity, but to measure the profit of classification accuracy when RAE's dictionary definitions are included in the training process of Word2Vec.

In section 2 we present the Word2Vec method, as well as review support vector machines. Section 3 shows the details of TASS corpus, the tool built to obtain RAE's dictionary definitions, and also shows a flow diagram of the proposed model for sentiment analysis. In section 4 we give and analyze the main results of our research. Finally, in section 5 we discuss the results and give some perspectives for future research, and we present the conclusions of this paper.

2 Theoretical Framework

In this section we present the tools used in our method of corpus extension. In order to do that we define some key concepts. A corpus is a collection or set of texts (also called documents) that share a topic or common origin. The vocabulary of this corpus is the set of all words contained in the corpus. In general, when texts are informal, there are sequences of characters that are not official words, but have a concrete meaning in a given cultural context. Thus, the concept of word is generalized by the concept of token, which is defined as any sequence of characters with a meaning. We also define the context of a word, referred as target word, as the set of words that are near to the target word. Specifically, the context has a size $2c$, which means that all the words that are at a maximum distance of c of the target word are context words.

Example 1 *Consider the sentence "en este día soleado me siento genial" (I feel great on this sunny day). Let "soleado" (sunny) be the target word. If $c = 2$, then the context of "soleado" is the set of words at a maximum distance of 2 from the target word. Thus the context is {"este", "día", "me", "siento"}.*

2.1 Word2Vec

One of the essential problems in automatic text analysis is to represent it numerically to establish quantitative relationships between words, phrases or longer texts. Mikolov et al. proposed the Word2Vec method, which supports the construction of embedded vector spaces whose dimensions harbor the semantic and syntactic meanings of words [13].

Word2Vec acts as a function $W2V : \mathcal{V} \to \mathbb{R}^N$, where $\mathcal{V} = \{w_i : i = 1, 2, \dots, V\}$ is an ordered set of V words (the way in which the words are ordered does not matter). The words, or tokens w_i are initially represented by hot-vectors, that are the canonical basis of $\mathbb{R}^V$. For instance, if "sausage" is the second word w_2 of the vocabulary

$\mathcal{V}$, then its hot-vector is

$$\boldsymbol{w}_2 = (0, 1, 0, \dots, 0), \tag{1}$$

and has V components. Hence, Word2Vec maps words from the vocabulary $\mathcal{V}$ to hot-vectors of V components, and later on, builds an embedded vector space of N dimensions in which each word w_i is paired with a vector $\boldsymbol{\omega}_i \in \mathbb{R}^N$. In general $N \ll V$, meaning that the embedded vector space has a much lower dimensionality than the number of words in the vocabulary (usually $N/V < 10^{-2}$).

The way in which the embedded vector space is built is via a neural network (NN). This NN learns the co-occurrences of words that share similar contexts. There exist two different possible configurations of the NN used by Word2Vec: CBOW and Skip-gram. CBOW means continuous bag of words, and predicts a target word given context words. This is, suppose that there are sentences in which a target word w_o is present. Beforehand, a context size c is set (to exemplify, suppose $c = 1$). This means that the target word w_o has contexts composed of two context words w_{before} and w_{after}, that are situated before and after w_o, respectively. Thus, the CBOW configuration is a NN whose input are the hot-vectors of w_{before} and w_{after}, and tries to predict the hot-vector of the target word w_o. On the other hand, the Skip-gram configuration takes as the input the hot-vector of the target word w_o and tries to predict its context words w_{before} and w_{after}.

For this work we selected the CBOW configuration that consisted of three neuron layers shown from bottom to top in Figure 1: the input, hidden and output layers. Again, if we suppose that the context has a size of $2c$, then the input layer contains $2c$ rows of V neurons in order to take the hot-vectors of the $2c$ context words. Each row of V neurons is fully connected to a hidden layer of N neurons. These connections are represented by $2c$ $N \times V$ matrices that relate each context word's hot-vector with the N-neurons hidden layer. Finally, the hidden layer is fully connected to the output layer, that contains V neurons. These connections are represented by a $V \times N$ matrix. Ideally, given a context, the NN should predict the target word's hot-vector. In the case of Figure 1, there are 4 context words and the correct target word is "soleado".

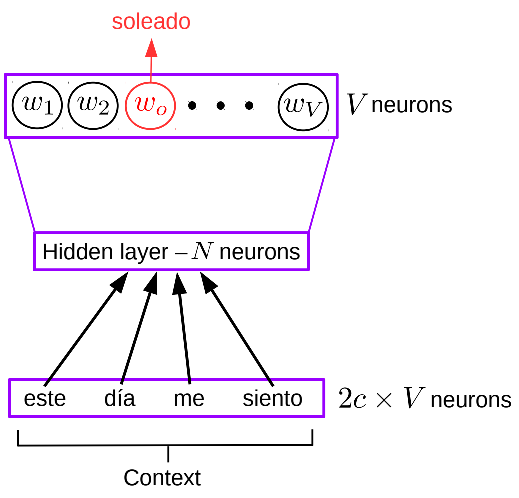

Figure 1: CBOW configuration of Word2Vec's NN using example 1.

The way in which the NN represents a word w_i as a vector in an N-dimensional vector space is by associating the i-th row of the connection matrix between the hidden and output layers with the i-th word. The reason why this works is that words that appear in similar contexts must have similar vector representations. This is achieved by defining the NN's error function as

$$E := -\log P(w_o|\mathcal{C}), \tag{2}$$

where $\mathcal{C}$ is a set of $2c$ words picked from the vocabulary, and $P(w_o|\mathcal{C})$ is the probability that w_o is the target word corresponding to the context $\mathcal{C}$. This probability is defined by a multinomial softmax distribution, which allows words from similar contexts to have similar vector representations in $\mathbb{R}^N$ [14], where the similarity is measured by the inner product.

2.2 Non linear support vector machine

A support vector machine (SVM) is a supervised learning method used for classification. This method consists in learning particular features of numerical objects that belong to different classes. The learning process starts with a training dataset

$$S := \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}, \tag{3}$$

where $\boldsymbol{x}_i \in \mathbb{X} \subset \mathbb{R}^p$ and $y_i \in \mathbb{Y} = \{-1, 1\}$ for all $i = 1, \ldots, m$. Note that there are only two tags because we performed binary classification (positive or negative polarity). Moreover, the observations $\boldsymbol{x}_i$ have p features.

A SVM looks for a pair of parameters $\boldsymbol{w} \in \mathbb{R}^p, b \in \mathbb{R}$ such that the hyperplane

$$\{\boldsymbol{x} \in \mathbb{R}^p : f(\boldsymbol{x}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\} \tag{4}$$

given by a discriminant function f, divides the space $\mathbb{R}^p$ in two semi-spaces: one that contains all the observations with tag -1, and the other one that contains the ones with tag 1. This means that the SVM builds two connected regions in $\mathbb{R}^p$, one for each class. Also, the distance from the hyperplane to the nearest observation's vector from each class must be maximized. These nearest vectors are called support vectors. The method so far presented is called linear SVM, and rarely is useful to classify real data.

To treat real data, Vapnik et al. proposed to map the set of observations $\mathbb{X}$ to a higher-dimensionality space, called the feature space $\mathbb{F} \subset \mathbb{R}^{\tilde{p}}$, where $\tilde{p} > p$, so that the observations are linearly separable in $\mathbb{F}$ [15]. This means that in $\mathbb{R}^{\tilde{p}}$ there exists a hyperplane that divides the space as desired. This map is built with a function $\Phi : \mathbb{R}^p \to \mathbb{R}^{\tilde{p}}$ that allows a re-definition of the discriminant function as

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b, \tag{5}$$

where $\boldsymbol{w} \in \mathbb{R}^{\tilde{p}}$. To make this learning process computationally efficient, a kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') := \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle. \tag{6}$$

If we write $\boldsymbol{w} = \displaystyle\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i$, where $\alpha_i \in \mathbb{R}, i = 1, 2, \ldots, m$, then

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b. \tag{7}$$

The reason why this is computationally efficient is that it is not necessary to explicitly compute Φ in order to compute $k(\boldsymbol{x}, \boldsymbol{x}')$. This is the case of the Gaussian kernel, or radial basis functions kernel (RBF), defined by

$$k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||), \tag{8}$$

where $\gamma \in \mathbb{R}^+$ is a free parameter that represents the influence that a vector $\boldsymbol{x}$ has over a vector $\boldsymbol{x}'$. In other words, since k_{RBF} is a similarity measure between two vectors, then γ is the inverse radius of influence of the support vectors. If this radius is small, then the SVM-RBF allows non-connected regions in $\mathbb{R}^p$ for each class.

It can be shown that the function Φ_{RBF} corresponding to the kernel $k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}')$ maps the observations to a vector space of infinite dimension [16], i.e.:

$$\Phi_{\text{RBF}} : \mathbb{R}^p \to \mathbb{R}^\infty, \tag{9}$$

which eases, in theory, the task of finding the hyperplane (in $\mathbb{R}^\infty$) that divides the observations from different classes.

In our work, we selected the RBF kernel because it is able to separate non-linear data in high-dimensional spaces, which is not necessarily the case of linear kernels. However, in the SVM it is possible to change the kernel by other non-linear kernels.

3 Method and Materials

In this section we show the stages and different materials used in our proposed model of extending the TASS General corpus with RAE's dictionary definitions. The model, and the way in which we measure its advantages with respect to not including RAE's dictionary definitions, can be seen in Figure 2. In sections 3.1-3.4 we explain the sequential flow of our method.

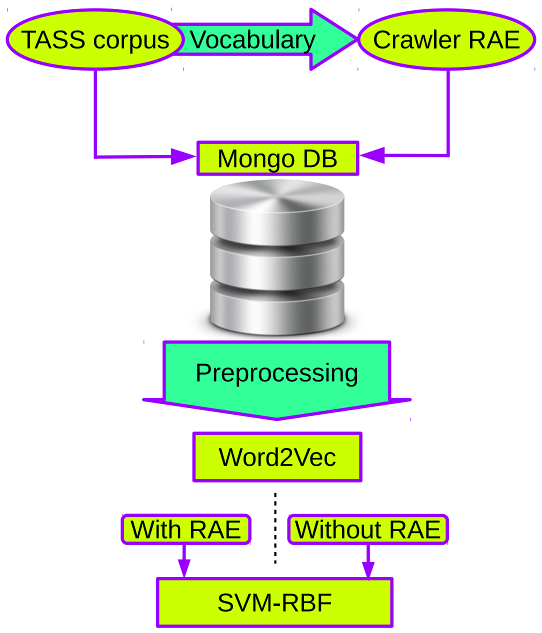

Figure 2: Flow diagram of the proposed model.

3.1 TASS corpus

TASS is a yearly-released workshop organized by the Spanish society for the natural language processing (SEPLN). TASS consists of activities related to sentiment analysis. Participants have access to tagged corpus that contains over 68,000 tweets in Spanish written by around 150 influent people (famous people or celebrities) from the world of politics, economy, media and culture, collected from November, 2011 to March, 2012 [1]. Figure 3 shows the tweet length in characters distribution. The figure exhibits the fact that this group of people tend to use the maximum number of characters to express their opinions.

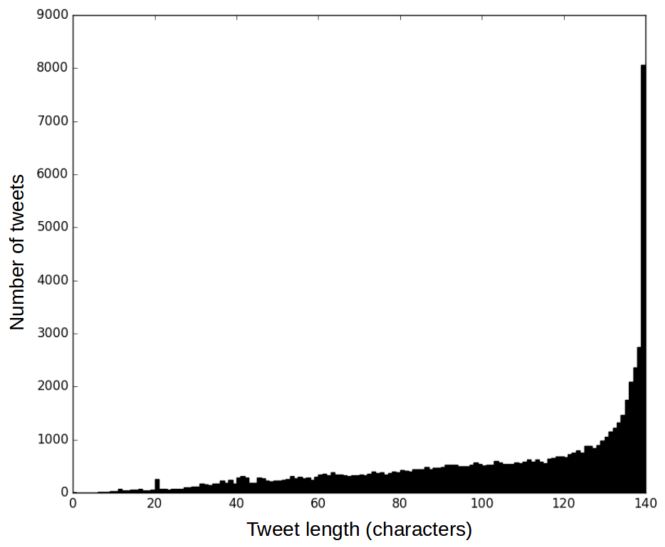

Figure 3: Histogram with tweets length distribution measured in characters.

Approximately 10% of the tweets are tagged with topic and polarity. For our study, the important tag is polarity and comes in 6 classes: strongly positive (P+), positive (P), neutral (NEU), negative (N), strongly negative (N+) and absence of polarity (NONE). In this work, we only used 4 polarity tags and grouped them together in two main classes: the positive class (P+ y P), and the negative one (N+ y N). This means that the set of tagged tweets has around 5100 tweets, corresponding to 70% of the initial tagged tweets.

The corpus is originally written in XML format. To read it and manage its information efficiently, the Python's Beautiful Soup library was used [17]. Additionally, the corpus was stored in a Mongo database using Python's Pymongo library with the objective of reading and writing information rapidly [18].

3.2 Crawler

With the purpose of extending the TASS corpus with RAE's dictionary definitions, a crawler was built in order to obtain such definitions. The process consisted in searching each token from the TASS corpus' vocabulary and to obtain the first definition, which is commonly the most used one. These definitions were stored in a collection of the Mongo database.

In pursuance of building the crawler, Selenium was used [19]. This tool allows to automatize multiple processes in web browsers. In our case, we automatized the process of typing each token in the search bar of RAE's dictionary, and to obtain the definitions of those tokens found in the dictionary.

It is important to point out that most of the tokens are not words present in the RAE's dictionary, as shown in Figure 4. This figure is a heat map, or a 2D histogram, in which dark (light) zones correspond to many (few) tweets, and whose the distribution of number of tokens per tweet, as well as the portion of those tokens found in RAE's dictionary. If a tweet is on the black, green or cyan line means that every, half or a quarter of its tokens are found in RAE's dictionary, respectively. Therefore, Figure 4 shows that most tweets have from a quarter to half of their tokens present in RAE's dictionary.

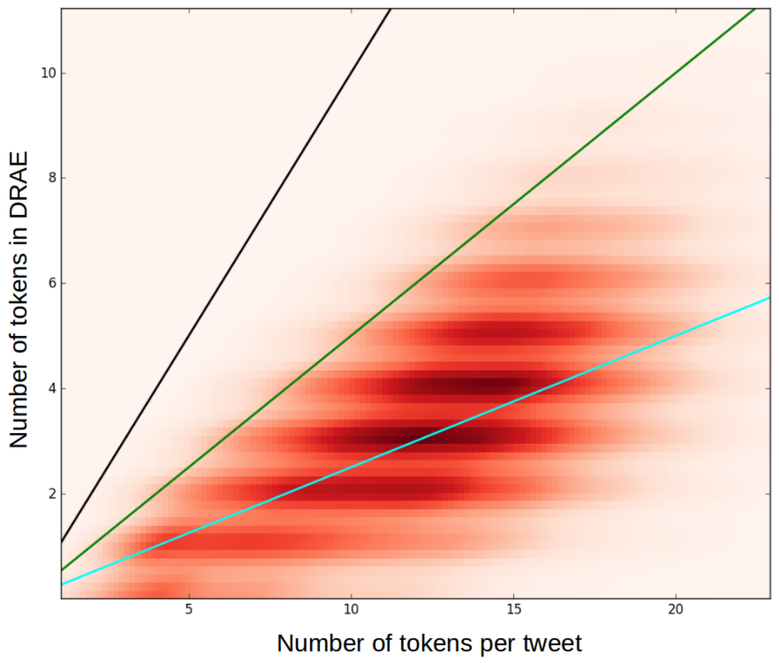

Figure 4: 2D histogram of number of tokens per tweet and the percentage of them that are found in RAE's dictionary (DRAE). Black, green and cyan lines correspond to 100%, 50% and 25%, respectively.

3.3 Preprocessing

The content of each tweet was modified following 5 preprocessing steps: cleaning, automatic correction, regularization, tokenization and lemmatization.

Cleaning

Usually, common words, also called stop-words, do not contribute to the semantic meaning of sentences. Therefore, they are not interesting for natural language processing since they do not supply new information and require computing time for their processing. Normally, these words are articles, pronouns, prepositions, among others. To remove them from the tweets, the NLTK library was used [20].

Automatic correction

In order to avoid words written with mistakes, Language Tool's API for Python was used [21]. Language Tool verifies grammar, orthography and style in a text. This tool allowed us to automatically correct words and structures found in tweets that were badly written.

Regularization

Although uppercase might be useful for some natural language processing applications [22], a good practice to analyze text using Word2Vec is to lowercase all characters. Also, the tokenizer that was used (see section 3.3) in this work required to remove accents from characters because it interprets them as token separators, i.e., words like "canción" (song) were tokenized as ["canci", "on"].

Tokenization

Tokenization is the process of converting a string of characters to a list of tokens. The automation of this procedure is implemented by NLTK's tokenizer TweetTokenizer [20].

Lemmatization

The last step of preprocessing is lemmatization. Here, tokens that are actual words are replaced by their lemma. For example, the token "canciones" was replaced by "canción". This task was performed using the pattern library from the computational linguistics group CLiPS [23]. The objective of this step was to reduce the vocabulary so that sentences containing words with the same lemma would be related by their respective embedded vector.

3.4 Experiments

We designed a model for training Word2Vec and training the SVM-RBF classifier, and measured its performance with the classification accuracy metric. The model depends on some training parameters, denoted by a tuple $\boldsymbol{\theta}$. Therefore, the model is a function $M : \Theta \rightarrow [0, 1]$, where Θ is the set of all possible parameters of Word2Vec and SVM-RBF. Moreover, M takes on values from 0 to 1, where 0 corresponds to 0% of classification accuracy, and 1 to 100%.

The process of training the model with a tuple of parameters and obtaining its corresponding classification accuracy is called the execution of an experiment. Each experiment was carried out in two stages. The first one consists in training Word2Vec's NN, and the second one consists in training and testing the SVM-RBF classifier. Furthermore, Word2Vec's NN training was performed in two different ways: the first one used both tweets and RAE's dictionary definitions as training sentences, and the second one used only tweets (see dashed line zone in Figure 2). This allows to define a first parameter $\theta_1 = 0, 1$, where 0 represents the first way and 1 the second one. A second parameter $\theta_2 = 20, 40, 60, 80, 100$ was called the minimum number of occurrences, and implies that vector representations of tokens that appear less than θ_2 times in all the corpus were not built. $\theta_3 = 3, 5, 7, 10$ was the third parameter, and corresponds to the maximum distance from the target word that a token has to have in order to be in its context (i.e. $\theta_3 = c$). The fourth parameter, $\theta_4 = 30, 60, 90, 120, 150, 190, 220, 250$, was the embedded vector space dimension, this is, the number of neurons in the hidden layer of Word2Vec's NN.

Once we obtained the tokens' vector representations, we built the tweets' vector representations for those tweets that were tagged with polarity in the corpus. This was done by taking the average of the vector representation of the tokens that composed each tweet. However, these construction of tweets' vector representation was not done for every tagged tweet. We introduced a fifth parameter $\theta_5 = 4, 5, 6, 7, 8, 9, 10$ called the minimum number of tokens per tweet, and the vector representations were not built for tweets with less than θ_5 tokens. This means that θ_5 limited the number of tweets that served as observations for the training and testing of the SVM-RBF classifier. In particular, the number of tweets varied from 1,112 to 4,074 (because both θ_5 and θ_2 affect the number of tweets), and these were split in 80% for training and 20% for testing. In order to keep the classes with the same amount of tweets, tweets were randomly eliminated from the class containing the most tweets until both classes had the same number of tweets. It should be noted that from the first stage the tweets were split into these two sets.

With respect to Word2Vec's implementation, Python's gensim library was used [24]. Regarding the SVM-RBF classifier, Python's scikit-learn library was used [25]. The algorithm implemented in this library not only contains the γ parameter, which is the sixth parameter $\theta_6 = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}$, but also has a parameter θ_7 that takes the values $\theta_7 = 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2$. The latter, trades off misclassification of training examples against simplicity of the discriminant surface. Low values of θ_7 generate smooth surfaces, while large values grants the algorithm the power to select more training vectors as support vectors, and build more complex discriminant surfaces.

In conclusion, the space of parameters Θ is the set of all possible tuples $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_7)$ made with the mention values for each parameter. Therefore 56,000 experiments were executed in total. Thus, 28,000 experiments were executed without using RAE's dictionary definitions and 28,000 using it.

We define $\tilde{\Theta}$ as the set of all possible tuples $\tilde{\boldsymbol{\theta}} = (\theta_2, \theta_3, \ldots, \theta_7)$. The reason why we carried out this brute force work of many experiments was the construction and comparison of two classification accuracy distributions defined over $\tilde{\Theta}$: one corresponding to $\theta_1 = 0$ (without RAE) and the other one corresponding to $\theta_1 = 1$ (with RAE). These distributions helped us answer the question of how probable is that $M(\boldsymbol{\theta}|_{\theta_1=1}) > M(\boldsymbol{\theta}|_{\theta_1=0})$ when we choose $\tilde{\boldsymbol{\theta}}$ randomly, where $\boldsymbol{\theta} = \theta_1 \oplus \tilde{\boldsymbol{\theta}}$ ($\oplus$ is used as concatenation symbol). In other words, the distributions determined if it is the case that extending the corpus with RAE's dictionary definitions to the set of training sentences of Word2Vec's NN improve the classification accuracy of polarities.

4 Results and Analysis

The classification accuracy distributions corresponding to experiments that only used tweets, and experiments that used both tweets and RAE's dictionary definitions as training sentences for Word2Vec's NN are shown in Figure 5.

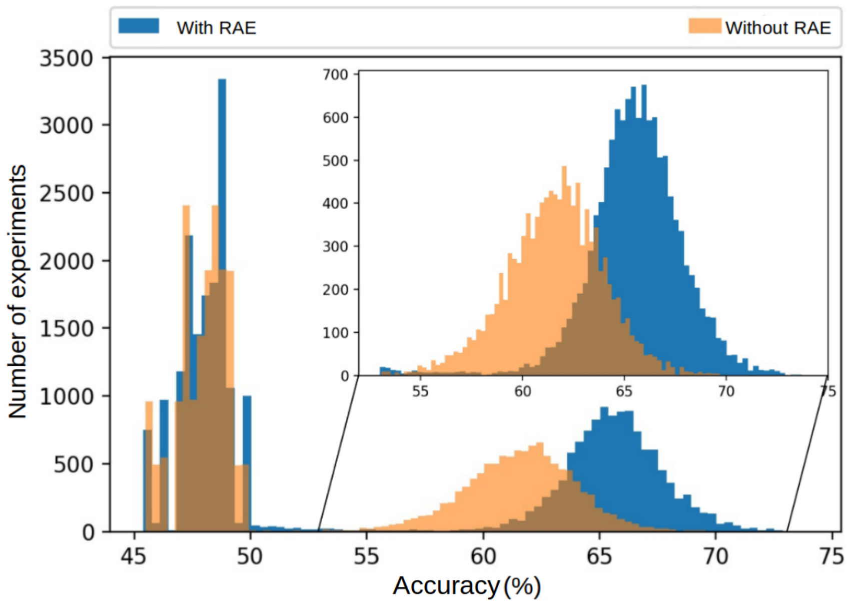

Figure 5: Classification accuracy distributions for experiments with and without RAE's dictionary definitions incorporated to the training sentences of Word2Vec's NN.

Figure 5 has two outstanding regions: the one corresponding to accuracy values greater or equal than 50% and the one with accuracy values lower than 50%. Both regions are interesting, but the first one offers overwhelming hints that using RAE's dictionary definitions within the set of training sentences of Word2Vec's NN improves classification accuracy significantly. In this region, both distributions are gaussian and contain 12,120 and 10,989 experiments that use and do not use RAE's dictionary definitions, respectively. Additionally, the gaussian distributions point out that, in average, using RAE's dictionary definitions as part of the training sentences of Word2Vec's NN improves the accuracy between 4%-5% with respect to the accuracy obtained when one does not use these definitions. Also, it is notable that the width of the distribution corresponding to experiments that incorporate RAE's dictionary definitions is lower than the width of the distribution of experiments that do not include them, which shows less dispersion and greater stability in the results.

The second region is interesting from the mathematical point of view. For two polarities, a random classifier would generate a gaussian distribution of classification accuracy centered at 50%. The data shown in Figure 5 below this percentage are clearly away from a gaussian distribution, and signal that the classifier's performance is significantly worse than a random classifier. A possible explanation of this phenomenon is that for some values of γ and θ_7, the SVM-RBF classifier generates very simple hypersurfaces that prevent a correct discrimination of non connected regions corresponding to a single polarity. This is evident from Figure 6, which shows a histogram of experiments grouped by the value of γ for three intervals of accuracy. The larger γ, the more complex and versatile the hypersurfaces are, and the higher the classification accuracy is (green bars). Conversely, the lower γ, the lower the classification accuracy is (pink bars).

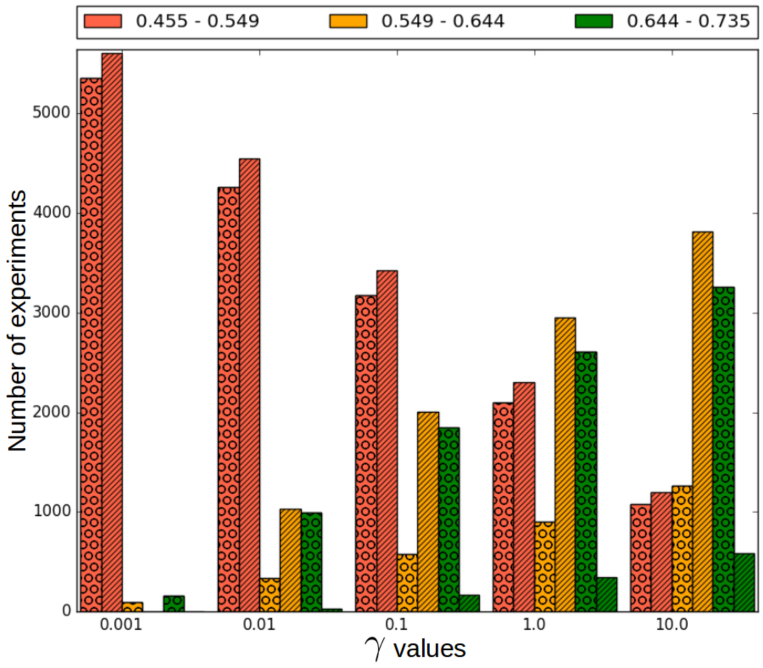

Figure 6: Histogram of number of experiments with different values of γ for three intervals of low (0.455-0.549), medium (0.549-0.644) y high (0.644-0.735) accuracies, shown with pink, yellow and green colors, respectively. Dotted (lined) bars correspond to experiments that use (do not use) RAE's dictionary definitions as part of Word2Vec's NN training sentences.

A similar behaviour was found with the parameter θ_7. Therefore, it is clear that the classifier is very sensible to the parameters $\gamma = \theta_6$ and θ_7. Since the best classification results were obtained with the highest values of θ_6 and θ_7, we built a comparison between experiments that used and did not use RAE's dictionary definitions as part of Word2Vec's NN training sentences, taking into account all the experiments with parameters $\theta_6 = 1, 10$ and $\theta_7 = 10, 100$. The comparison was made in the following way: A histogram of the following set was made:

$$\{M(\boldsymbol{\theta}|_{\theta_1=1}) - M(\boldsymbol{\theta}|_{\theta_1=0}) : \boldsymbol{\theta} \in \Theta \wedge \theta_6 = 1, 10 \wedge \theta_7 = 10, 100\}.$$

In other words, the comparison was made by producing a histogram of the improvement percentage in the classification accuracy when RAE's dictionary definitions were included as part of Word2Vec's NN training sentences. This is, given a tuple of parameters, the histogram shown in Figure 7 tells the probability that the accuracy obtained when using RAE's dictionary definitions is higher than when they are not used.

Figure 7 shows that the improvement percentage in classification accuracy is statistically significant (to 1σ). For the selected subset of experiments, this improvement is, in average, of 3%.

5 Conclusions

The main goal of this paper was to measure the impact on the classification accuracy provided by the inclusion of RAE's dictionary definitions in Word2Vec's NN training sentences. It was found that most of the tweets from the TASS corpus used near the maximum number of characters allowed by Twitter, and that about a quarter to a half of the tokens contained in each tweet were actual words found in RAE's dictionary. The arguments exposed in Section 4 assert that there is a significant improvement in the classification accuracy when RAE's dictionary definitions are included in the Word2Vec's NN training sentences, whenever γ and θ_7 are correctly

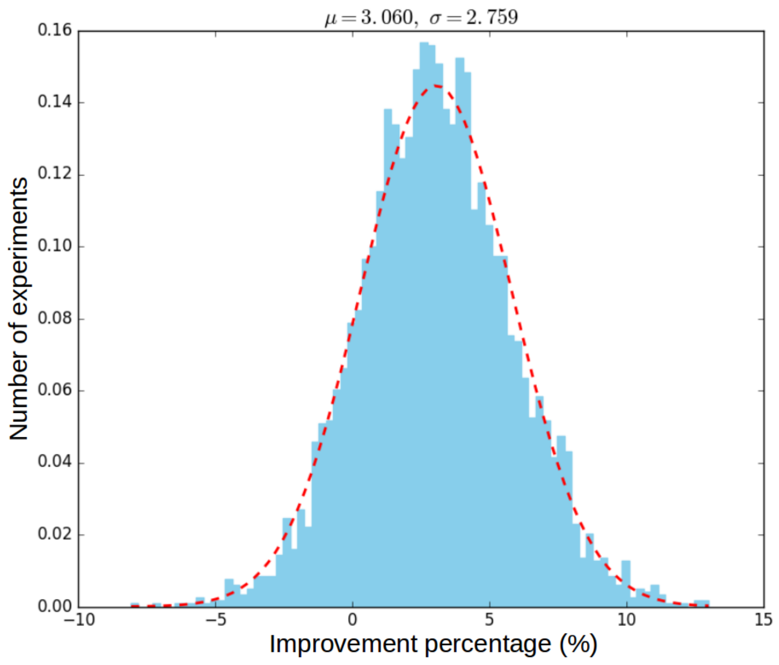

Figure 7: Normalized histogram of the number of experiments by the improvement percentage obtained when RAE's dictionary definitions were included as part of Word2Vec's NN training sentences. The red line shows a gaussian curve with mean 3.060% and standard deviation 2.759%.

selected. Furthermore, it is clear that the accuracy percentages of the binary polarity classification are low and can be improved including a lexicon polarity dictionary in order to build the vector representations of tweets more adequately. Finally, it is important to verify that this improvement does not only occur with the Word2Vec method, but also with other representation models (e.g. GloVe) of words as embedded vectors. Also relevant is to measure the impact of including RAE's dictionary definitions as part of Word2Vec's NN training sentences in different corpora; for instance corpora with longer and better written texts in order to ensure that a larger percentage of these texts' tokens are found in RAE's dictionary, which could be helpful for automatic news analysis like predicting stock prices with financial news articles [26].

References

[1] Eugenio Martínez Cámara, Miguel Á. García Cumbreras, Julio Villena Román, and Janine García Morera. Tass 2015 – the evolution of the spanish opinion mining systems. *Procesamiento del Lenguaje Natural*, 56:33–40, 2016.

[2] Yuan Man. Feature extension for short text categorization using frequent term sets. *Procedia Computer Science*, 31:663 – 670, 2014. 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014. URL: `http://www.sciencedirect.com/science/article/pii/S1877050914004918`, `doi:http://dx.doi.org/10.1016/j.procs.2014.05.314`.

[3] S. Lee, J. Kim, and S. H. Myaeng. An extension of topic models for text classification: A term weighting approach. In *2015 International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 217–224, 2015. `doi:10.1109/35021BIGCOMP.2015.7072834`.

[4] Pu Wang, Carlotta Domeniconi, and Jian Hu. Cross-domain text classification using wikipedia. *IEEE Intelligent Informatics Bulletin*, 9(1):5–17, 2008.

[5] Marcos Antonio Mouriño García, Roberto Pérez Rodríguez, and Luis Anido Rifón. Wikipedia-based cross-language text classification. *Information Sciences*, 406–407:12 – 28, 2017. URL: `http://www.sciencedirect.com/science/article/pii/S0020025517306680`, `doi:https://doi.org/10.1016/j.ins.2017.04.024`.

[6] Antonio Fernández Anta, Luis Núñez Chiroque, Philippe Morere, and Agustín Santos. Sentiment analysis and topic detection of Spanish tweets: A comparative study of NLP techniques. *Procesamiento del Lenguaje Natural*, 50:45–52, 2013.

[7] F. Javier Ortega, José A. Troyano, Fermín L. Cruz, and Fernando Enríquez. Enriching user reviews through an opinion extraction system. *Procesamiento del Lenguaje Natural*, 55:119–126, 2015.

[8] Aitor García-Pablos and Germán Rigau. Unsupervised word polarity tagging by exploiting continuous word representations. *Procesamiento del Lenguaje Natural*, 55:127–134, 2015.

[9] Fermín L. Cruz, José A. Troyano, Beatriz Pontes, and F. Javier Ortega. Ml-senticon: Un lexicón multilingüe de polaridades semánticas a nivel de lemas. *Procesamiento del Lenguaje Natural*, 53:113–120, 2014.

[10] Flor Miriam Plaza-del Arco, M. Teresa Martín-Valdivia, Salud María Jiménez-Zafra, M. Dolores Molina-González, and Eugenio Martínez-Cámara. Copos: Corpus of patient opinions in spanish. application of sentiment analysis techniques. *Procesamiento del Lenguaje Natural*, 57:83–90, 2016.

[11] L. Alfonso Ureña López, Rafael Muñoz Guillena, José A. Troyano Jiménez, and María-Teresa Martín Valdivia. Attos: Análisis de tendencias y temáticas a través de opiniones y sentimientos. *Procesamiento del Lenguaje Natural*, 53:151–154, 2014.

[12] Fernando Batista and Ricardo Ribeiro. Sentiment analysis and topic classification based on binary maximum entropy classifiers. *Procesamiento del Lenguaje Natural*, 50:77–84, 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] Yoav Goldberg and Omer Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method, 2014.

[15] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. URL: `http://doi.acm.org/10.1145/130385.130401`, `doi:10.1145/130385.130401`.

[16] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1st edition, 2000.

[17] Leonard Richardson. Beautiful soup documentation, 2015. URL: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/` [cited February 27, 2017].

[18] MongoDB Inc. Pymongo 3.4.0 documentation. URL: `https://api.mongodb.com/python/current/#about-this-documentation` [cited March 5, 2017].

[19] Sagar Shivaji Salunke. *Selenium Webdriver in Python: Learn with Examples*. CreateSpace Independent Publishing Platform, 1st edition, 2014.

[20] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[21] Steven Myint. Language check. URL: `https://pypi.python.org/pypi/language-check` [cited March 10, 2017].

[22] Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. An impact analysis of features in a classification approach to irony detection in product reviews. *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49, 2014.

[23] Tom De Smedt and Walter Daelemans. Pattern for python. *Journal of Machine Learning Research*, 13:2063–2067, 2012.

[24] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on NewChallenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. ELRA.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Robert P. Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Trans. Inf. Syst.*, 27(2):12:1–12:19, 2009. URL: `http://doi.acm.org/10.1145/1462198.1462204`, `doi:10.1145/1462198.1462204`.