Applying Recurrent Neural Networks to Sentiment Analysis of Spanish Tweets

Aplicación de Redes Neuronales Recurrentes al Análisis de Sentimientos sobre Tweets en Español

Oscar Araque, Rodrigo Barbado, J. Fernando Sánchez-Rada y **Carlos A. Iglesias**
Intelligent Systems Group, Universidad Politécnica de Madrid
Av. Complutense 30, 28040 Madrid
o.araque@upm.es, rodrigo.barbado.esteban@alumnos.upm.es
{jfernando, carlosangel.iglesias}@upm.es

Abstract: This article presents the participation of the Intelligent Systems Group (GSI) at Universidad Politécnica de Madrid (UPM) in the Sentiment Analysis workshop focused in Spanish tweets, TASS2017. We have worked on Task 1, aiming to classify sentiment polarity of Spanish tweets. For this task we propose a Recurrent Neural Network (RNN) architecture composed of Long Short-Term Memory (LSTM) cells followed by a feedforward network. The architecture makes use of two different types of features: word embeddings and sentiment lexicon values. The recurrent architecture allows us to process text sequences of different lengths, while the lexicon inserts directly into the system sentiment information. The results indicate that this feature combination leads to enhanced sentiment analysis performances.
Keywords: Deep Learning, Natural Language Processing, Sentiment Analysis, Recurrent Neural Network, TensorFlow

Resumen: En este artículo se presenta la participación del Grupo de Sistemas Inteligentes (GSI) de la Universidad Politécnica de Madrid (UPM) en el taller de Análisis de Sentimientos centrado en tweets en Español: el TASS2017. Hemos trabajado en la Tarea 1, tratando de predecir correctamente la polaridad del sentimiento de tweets en español. Para esta tarea hemos propuesto una arquitectura consistente en una Red Neuronal Recurrente (RNN) compuesta de celdas *Long Short-Term Memory* (LSTM) seguida por una red neuronal prealimentada. La arquitectura hace uso de dos tipos distintos de características: *word embeddings* y los valores de un diccionario de sentimientos. La recurrencia de la arquitectura permite procesar secuencias de texto de distintas longitudes, mientras que el diccionario inserta información de sentimiento directamente en el sistema. Los resultados obtenidos indican que esta combinación de características lleva a mejorar los resultados en análisis de sentimientos.
Palabras clave: Aprendizaje Profundo, Procesamiento de Lenguaje Natural, Análisis de Sentimientos, Red Neuronal Recurrente, TensorFlow

1 Introduction

Recent developments in the area of deep learning are strongly impacting sentiment analysis techniques. While traditional methods based on feature engineering are still prevalent, new deep learning approaches are succeeding and reduce the need of labeled corpus and feature definition. Moreover, traditional and deep learning approaches can be combined obtaining improved results (Araque et al., 2017).

This paper describes our participation in TASS 2017 (Martínez-Cámara et al., 2017). Taller de Análisis de Sentimientos en la SEPLN (TASS) is a workshop that fosters the research of sentiment analysis in Spanish for short text such as tweets. The first task of this challenge, Task 1, consists in determining the global polarity at a message level. The dataset for the evaluation of this task

ISSN 1613-0073

considers annotated tweets with 4 polarity labels (P, N, NEU, NONE). P stands for positive, while N means negative and NEU is neutral. It is considered that NONE means absence of sentiment polarity. This task provides a corpus, which contains a total of 1514 tweets written in Spanish, describing a diversity of subjects.

We have faced this challenge as an opportunity to evaluate how these techniques could be applied in the TASS domain, and their results compared with the traditional techniques we used in a previous participation in this challenge (Araque et al., 2015).

The reminder of this paper is organized as follows. Sect. 2 introduces related work. Then Sect.3 describes the proposed polarity classification model and its implementation, which is evaluated in Sect. 4. Finally, conclusions are drawn in Sect. 6.

2 *Related work*

Many works in the last years involve the use of neural architectures to learn text classification problems and, more specifically, to perform Sentiment Analysis. A relevant example of this are Recursive Neural Tensor Networks (Socher et al., 2013). This architecture makes use of the structure of parse trees to effectively capture the negation phenomena and its scope. A similar work (Tai, Socher, and Manning, 2015) introduces the use of LSTM in tree structures, leveraging both the information contained in these trees and the representation capabilities of gated units. Although parse trees can result very useful in sentiment analysis, many works do not make use of them, as they introduce an additional computation overhead. In (Wang et al., 2015) a data-driven approach is described that learns from noisy annotated data also making use of LSTM units and a error signal processing to avoid the problem of vanishing gradient. Another useful technique is *attention* (Bahdanau, Cho, and Bengio, 2014), that enables weighting the importance of the different words in a given piece of text. Attention has been used in Sentiment Analysis successfully in a recurrent architecture, as presented in (Wang et al., 2016).

In the context of the TASS challenge, it has not been the first time that neural architectures have been proposed for solving the different tasks. In (Vilares et al., 2015), the authors propose a LSTM architecture that is compared to linear classifiers. Also, word embeddings have been leveraged in previous versions, as shown in (Martınez-Cámara et al., 2015). Nevertheless, neural networks have not been thoroughly studied in TASS, and many potentially interesting techniques remain unused.

3 *Sentiment analysis Task*

3.1 Model architecture

The approach followed for the Sentiment Analysis at Tweet level Task consists in a RNN composed of LSTM cells that parse the input into a fixed-size vector representation. The representation of the text is used to perform the sentiment classification. Two variations of this architecture are used: (i) a LSTM that iterates over the input word vectors or (ii) over a combination of the input word vectors and polarity values from a sentiment lexicon.

The general architecture of the model takes as inputs the words vectors and the lexicon values for each word from an input tweet. Then, the inputs are passed through a one-layer LSTM with a tunnable number of hidden units. The generated representation is then used to determine the polarity of the input text using a feedforward layer with softmax activation as output function. The output of this last layer encodes the probability that the input text belongs to each class. Fig. 2 shows this architecture, which is further described as follows:

1. The input vector is the word embedding of each word in a given tweet. It contains word-level information or sentiment word-level information. Each specific case will be described in more detail afterwards.
2. The RNN number of units is chosen during training for optimization purposes. In this work we use a one-layer LSTM to avoid overfitting of the network to the training data.
3. The weight matrix has as input dimension the RNN size, and the number of classes as output dimension. This means that, taking as inputs the last LSTM output, we obtain a vector whose length is the number of classes. This matrix is also optimized during the training process.

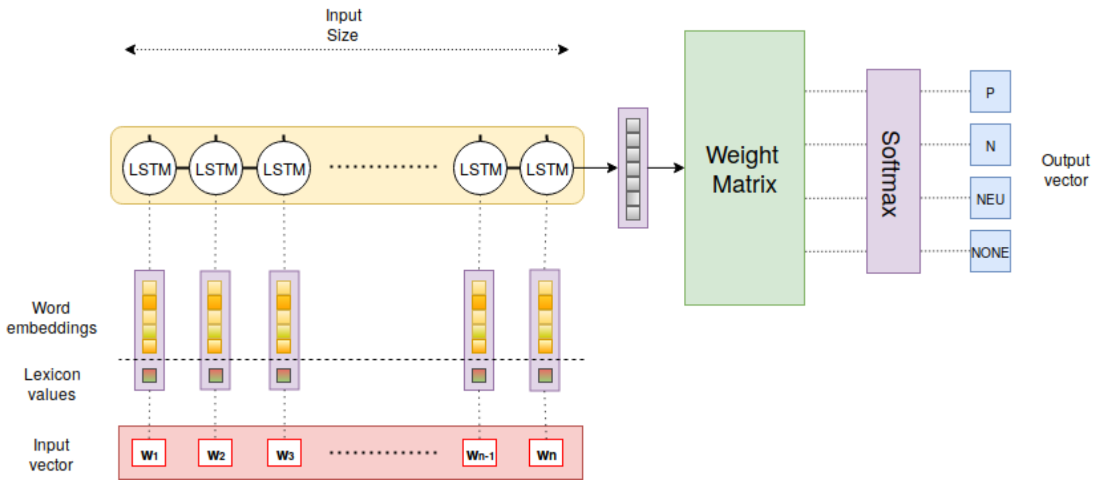

Figure 1: Recurrent Neural Network (RNN) architecture

4. The final probability vector is obtained by passing the result of the previous matrix multiplication through a softmax function, which converts the values of the components of this result vector into probabilities representation. Finally, the predicted label for the tweet is the component of the output vector with the highest probability.

Following, the two types of inputs used are described thoroughly.

3.2 Word-level RNN

For this input, the tweet text is tokenized into word tokens, which are then expressed in a one-hot representation. That is, each token is represented as a $\mathbb{R}^{|V|\times 1}$ vector with all 0s and and one 1 at the index of that token in the sorted token vocabulary. For example, the representations for the tokens *a*, *antes* and *zebra* would appear as:

$$w_{\mathrm{a}} = \left[1\,0\,0\,\cdots\,0\right], w_{\mathrm{antes}} = \left[0\,1\,0\,\cdots\,0\right]$$
$$w_{\mathrm{zebra}} = \left[0\,0\,0\,\cdots\,1\right]$$

We limit the number of words to a certain vocabulary size in order to limit the computational cost of this preprocessing step. Before feeding this data to the network, each tweet is presented by the one-hot representation of all the tokens in the tweet.

3.3 Sentiment word-level RNN

Additionally, we include different sentiment information into the word representations by means of a sentiment lexicon. In this case, a similar approach as the word-level RNN is followed, but instead of using information about the different words contained on each tweet, information about the sentiment of each word is used. In this case, the preprocessing process is modified:

1. First, each tweet is split into tokens.
2. Secondly, a sentiment dictionary is used to map words with sentiment polarity values. In this way, each word is mapped into a positive, neutral or negative value.
3. Finally, the representation of a word consists in its word vector concatenated with its sentiment polarity label.

3.4 Regularization

Given the reduced number of training examples that is available for this task (Sec. 4) a number of regularization techniques has been used in the experiments. Regularization is used in machine learning to control the complexity of a learning model so it does not overfit to the training data and generalized better to the test data.

It is known that Recurrent Neural Network tend to heavily overfit to the training set (Zaremba, Sutskever, and Vinyals, 2014). For this, we employ two regularization techniques to prevent this:

1. L2 regularization (Ng, 2004). This technique is applied on the weights of the feedforward layer of the network. Being W_{MLP} the weights of this layer, this regularization adds to the cost function the following value:

$$\lambda \| W_{MLP}^T W_{MLP} \|$$

where λ is a parameter that represents the importance that is assigned to this regularization in the overall cost function.

2. Dropout (Srivastava et al., 2014; Gal and Ghahramani, 2016b). This strategy consists in randomly setting a fraction of units to 0 at each step of the training process to prevent overfitting. During test time, the outputs are averaged by this fraction. Dropout has been recently found to be theoretically similar to applying a bayesian prior to the network weigths (Gal and Ghahramani, 2016a).

3.5 TensorFlow implementation

TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms (Abadi, Agarwal, and et al., 2015). For implementing the model previously described, first we had to define a computation graph composed by the RNN architecture, matrices and operations needed. Once the graph is defined, the training process consists in iteratively adjusting numerical values in order to reach the best results. This task was done following those ideas:

- The values to optimize are the internal parameters and matrices that form the network. Those are: the word embedding representations, the LSTM internal weights and the feedforward weight matrix used as last layer. At the beginning of training, those values are initialized in a random way using a normal distribution $\sim \mathcal{N}(\mu, \sigma)$, with $\mu = 0$, and are considered variables to be optimized at each training step by TensorFlow.
- Having those variables defined, the training process iteratively modifies them in order to reach the better results. In order to obtain a error signal that can be used to modify the learning parameters we use a cost function which has to be minimized. That minimization problem is solved by applying the gradient descent method via backpropagation (LeCun et al., 2012). In this work we employ the Adam algorithm (Kingma and Ba, 2014).
- In each iteration of the training process, which are known as epochs, data from the training set flows through all the computation graph yielding to a prediction result. The cost metric is computed by comparing the obtained result with the true training labels. When the backpropagation is finished, the variable values are updated and the following iteration proceeds.
- In order to enhance performance, we use early stopping on the accuracy on the development set. That is, for each epoch we monitor the performance of the network in the development set. If it has not improved for a number of epochs (in this work, 3 epochs) the training process is stopped and the model weights are freezed.
- The number of iterations can be chosen as well as other parameters such as the RNN size. For testing new examples, we use as input the test data, passing all the tweets through our model having as a result the vector of probabilities of the class each tweet belongs to, choosing the class with a higher probability value for each tweet.

4 Experimental setup

For the development of Task 1 a training and development dataset is made public, containing 1,514 labeled tweets which belong to the InterTASS corpus. Additionally, we use the TASS2015 edition training dataset that was extracted from the general corpus (García Cumbreras et al., 2016). We train the system with the InterTASS and the TASS General Corpus training datasets, and adjust the hyper-parameters with the InterTASS development set. For the lexicon, we used ElhPolar dictionary (Urizar and Roncal, 2013), as it has been previously used in TASS competitions.

There are three test datasets, one belonging to the InterTASS corpus and two belonging to the General Corpus of TASS: the full version, with all the 60,798 tweets; and the 1k version, that contains a subset of 1,000 tweets.

In order to enhance the classification performance several hyper-parameters have been explored, and the values that yield better performance are selected to be used in the testing phase. The vocabulary size is set to 20,000, with a batch size of 256 and the num-

Model	Corpus	Accuracy	Macro-F1
LSTM + MLP	InterTASS	53.70	37.1
LSTM + MLP	TASS (1k)	60.1	45.6
LSTM + MLP	TASS (Full)	63.1	50.9
LSTM + MLP + Lexicon	InterTASS	56.2	38.7
LSTM + MLP + Lexicon	TASS (1k)	63.6	46.8
LSTM + MLP + Lexicon	TASS (Full)	63.1	49.7

Table 1: Results in TASS 2017

ber of epochs being 20. With this value, the early stopping mechanism stopped the training before its completion. Regarding the size of the layers, the number of dimensions of the word embeddings is set to 16, as well as it is done with the number of units in the LSTM layer. The dimensionality of the feedforward layer is given by the output of the LSTM, which is 16, and the number of classes of the classification task (in this case, 4). Note that these values are smaller than in the usual neural architectures in order to further prevent overfitting. Also, we select the λ parameter to 0.05, and the dropout rate to 0.7.

5 Experimental Results

Table 1 shows the results of the two variations of the proposed model: LSTM+MLP stack with or without lexicon values. In light of this results it is possible to affirm that the used architecture shows promising performances in the task of sentiment analysis of tweets. Although, the achieved performances are below the best in this year challenge. This indicates that further work should be done in order to improve the results.

The experimental results confirm the idea that the introduction of a sentiment lexicon into the word presentations results, in general, beneficial for the final performance. We see this improvement in the InterTASS and 1k corpora. Nevertheless, when attending to the Full corpus, a performance decrease in the Marco-F1 is observed.

6 Conclusions and Future Work

In this paper we have described the participation of the GSI in the TASS 2017 challenge. Our proposal relies on a Recurrent Neural Network architecture for Sentiment Analysis with Long Short-Term Memory cells. This network can be fed with both word vectors and sentiment lexicon values. This approach is able to represent a arbitrarily long sequence of text due to the dynamic recurrent structure of the architecture. Also, several techniques have been used for avoiding overfitting. From the experiments, it is seen that adding a sentiment lexicon can enhance the classification performance.

However, the proposed model does not compare with the best results in the TASS competition. This can be due to a number of reasons, but the training process suggests that overfitting is a relevant issue. Although benefit comes from the use of regularization techniques, the network is not able to largely generalize. To address this, we think that future work in this direction should include the expansion of the training set.

Other possible improvement for future work is doing a better preprocessing of input texts at word level. In addition, Convolutional Neural Networks could be used for feature extraction in combination with the Recurrent Neural Network architecture. This could lead to the computation of most complex features, which could also yield better results.

Acknowledgement

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used in this research. This research work is partially supported through the projects Semola (TEC2015-68284-R), Emo-Spaces (RTC-2016-5053-7), MOSI-AGIL (S2013/ICE-3019), Somedi (ITEA3 15011) and Trivalent (H2020 Action Grant No. 740934, SEC-06-FCT-2016).

Bibliografía

Abadi, M., A. Agarwal, and P. B. et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Araque, O., I. Corcuera, C. Román, C. A.

Iglesias, and J. F. Sánchez-Rada. 2015. Aspect based sentiment analysis of spanish tweets. In *TASS@ SEPLN*, pages 29–34.

Araque, O., I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias. 2017. Enhancing Deep Learning Sentiment Analysis with Ensemble Techniques in Social Applications. *Expert Systems with Applications*, June.

Bahdanau, D., K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Gal, Y. and Z. Ghahramani. 2016a. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gal, Y. and Z. Ghahramani. 2016b. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

García Cumbreras, M. Á., E. Martínez Cámara, J. Villena Román, and J. García Morera. 2016. Tass 2015–the evolution of the spanish opinion mining systems. *Procesamiento del Lenguaje Natural*, 56:33–40.

Kingma, D. and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y. A., L. Bottou, G. B. Orr, and K.-R. Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, pages 9–48.

Martínez-Cámara, E., M. C. Díaz-Galiano, M. A. García-Cumbreras, M. García-Vega, and J. Villena-Román. 2017. Overview of tass 2017. In J. Villena Román, M. A. García Cumbreras, D. G. M. C. Martínez-Cámara, Eugenio, and M. García Vega, editors, *Proceedings of TASS 2017: Workshop on Semantic Analysis at SEPLN (TASS 2017)*, volume 1896 of *CEUR Workshop Proceedings*, Murcia, Spain, September. CEUR-WS.

Martınez-Cámara, E., Y. Gutiérrez-Vázquez, J. Fernández, A. Montejo-Ráez, and R. Munoz-Guillena. 2015. Ensemble classifier for twitter sentiment analysis. In R. Izquierdo, editor, *Proceedings of the Workshop on NLP Applications: completing the puzzle*, number 1386 in CEUR Workshop Proceedings, Aachen.

Ng, A. Y. 2004. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

Socher, R., A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Tai, K. S., R. Socher, and C. D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Urizar, X. S. and I. S. V. Roncal. 2013. Elhuyar at tass 2013. In *Proceedings of the Workshop on Sentiment Analysis at SEPLN (TASS 2013)*, pages 143–150.

Vilares, D., Y. Doval, M. A. Alonso, and C. Gómez-Rodríguez. 2015. Lys at tass 2015: Deep learning experiments for sentiment analysis on spanish tweets. In *TASS@ SEPLN*, pages 47–52.

Wang, X., Y. Liu, C. Sun, B. Wang, and X. Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *ACL (1)*, pages 1343–1353.

Wang, Y., M. Huang, X. Zhu, and L. Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, pages 606–615.

Zaremba, W., I. Sutskever, and O. Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.