Towards A Deep-Learning-Based Methodology for Supporting Satire Detection

Alfredo Cuzzocrea*
iDEA Lab, University of Calabria, Rende, Italy & LORIA, Nancy, France
alfredo.cuzzocrea@unical.it

Giosué Lo Bosco
Dept of Computer Science, University of Palermo & IEMEST, Palermo, Italy
giosue.lobosco@unipa.it

Mariano Maiorana
Dept of Computer Science, University of Palermo, Palermo, Italy
mariano.maiorana@community.unipa.it

Giovanni Pilato
ICAR-CNR, Palermo, Italy
giovanni.pilato@cnr.it

Daniele Schicchi
Istituto Euro-Mediterraneo di Scienza e Tecnologia, Palermo, Italy
danieleschicchi@iemest.eu

Abstract

This paper describes an approach for supporting automatic satire detection through effective deep learning (DL) architecture that has been shown to be useful for addressing sarcasm/irony detection problems. We both trained and tested the system exploiting articles derived from two important satiric blogs, Lercio *and* IlFattoQuotidiano*, and significant Italian newspapers.*

1 Introduction

Satire is a way of criticizing people (or ideas) by ridiculing them on political, social, and morals topics (e.g., [18]. Most of the time, such a language form is utilized to influence people's opinions. It is a figurative form of language that leverages comedic devices such as *parody* (i.e to imitate techniques and style of some person, place or thing), *exaggeration* (i.e to represent something beyond normality make it ridiculous), *incongruity* (i.e to present things that are absurd concerning the context), *reversal* (i.e to present the opposite of normal order), *irony/sarcasm* (i.e to say something that is the opposite of what a person mean). Moreover, satire masks emotions like irritation and disappointment by using ironic content.

The easy way of denouncing political and societal problems exploiting humor has brought consensus to satire that has been widely accepted. It leads people to constructive social criticism, to participate actively in the socio-political life, representing a sign of democracy. Unfortunately, the ironic nature of satire tends to mislead subjects that can believe the humorous news as they were real; therefore, satirical news can be deceptive and harmful.

Detecting satire is one of the most challenging computational linguistics tasks, natural language processing, and social multimedia sentiment analysis. It differs from irony detection since satire *mocks* something or someone, while irony is intended to be a way for causing laughter. Tackling such a task means both to pinpoint linguistic entities that characterize satire and look at how they are used to express a more complex meaning.

As satirical texts include figurative communication for expressing ideas/opinions concerning people, sentiment analysis systems may be negatively affected. In this case, satire should be adequately addressed to avoid performances degradation of such systems, mainly if sar-

*This research has been made in the context of the Excellence Chair in Computer Engineering — Big Data Management and Analytics at LORIA, Nancy, France

DOI reference number: 10.18293/DMSVIVA2021-016

casm/irony is used [1]. Moreover, reliably detecting satire can benefit many other research areas where figurative language usage can be a problem, such as *Affective Computing* [15]. An autonomous way of detecting satire might help computers interpret human interaction and notice its emotional state, improving the human-computer experience.

In this paper, we tackle automatic satire detection through effective deep learning (DL) architecture that has been shown to be effective for addressing the sarcasm/irony detection problem. The Neural Network (NN) exploits articles derived from two important satiric blogs, *Lercio* and *IlFattoQuotidiano*, and major Italian newspapers. The dataset has been specifically created for the task, and it includes news concerning similar topics. Experiments show an optimal performance achieved by the network that is capable of performing well on satire recognition. The network demonstrates the ability to detect satire in a context where it is not marked as in IlFattoQuotidiano. In fact, in this special case, news are so realistic that they seem to be true. [15]. An autonomous way of detecting satire might help computers interpret human interaction and notice its emotional state, improving the human-computer experience.

On the other hand, studying these techniques as combined with the emerging *big data trend* (e.g., [11, 10, 6, 8, 9]) is an interesting challenge.

The extended version of this paper appears in [7].

2 The Overall Proposed Methodology

Recognizing *satire* can be modeled as a classification task subdividing *satiric* and *non-satiric* articles in two different classes. Such a task has been widely tackled by using machine learning algorithms, and it has been shown that it is important to consider various aspects related to the application domain. For what concerns the subject problem, many factors should be taken into account: the way the text is represented and how it is structured (sec. 2.1), the model's architecture for tackling the task and its tuning (sec 2.2 and 2.3). Le Hoang Son et al. [13] have introduced a deep learning model that promises optimal performances for detecting sarcasm/irony. We believe that such a network can also help recognizing the main aspects of the satire; a detailed description is given in sec. 2.2.

2.1 Preprocessing

The preprocessing phase deals with the input arrangement to make it analyzable to the model as best as possible. Most of the time, the text is changed by removing punctuation marks, stop-words, etc. In this case, since the articles have been harvested from online resources we focused on the removal of the *author's name*, *HTML tags*, *hyperlinks*, and *hashtags*. Subsequently, the input text is split into tokens (i.e., words and punctuation marks) using NLTK [1]. To level out the lengths of the articles, we have analyzed the cumulative frequency of the length of the texts, and then we have selected a value $L = 4500$ words such that we considered 95% of the entire set of articles. Finally, each token is mapped to a 300-dimensional space by a pre-trained embedding tool that relies on FastText [3, 12]. Therefore, each article is represented by a matrix of real values of size $(L, 300)$. We crop texts longer than L, and we pad with 0s texts that are shorter.

2.2 Architecture

The network's architecture is inspired from the one presented by Le Hoang Son et al [13], that exploits *Bidirectional Long Short Term Memory* (BiLSTM), *Soft Attention Mechanism*, *Convolutional* NNs, and *Fully Connected* NNs. Moreover, such a model consider five different auxiliary characteristics that have been shown to be relevant to sarcasm/irony detection: number of exclamation marks (!), number of question marks (?), number of periods (.), number of capital letters, number of uses of *or*. A complete model representation is given in figure 1.

2.2.1 Input Layer

The first network's layer is the *Input* layer which manage the pre-processed text in order to allow the analysis by the BiLSTM.

2.2.2 BiLSTM Layer

BiLSTM is composed of two LSTM layers which examine respectively the input sequence in *forward* (from the first token x_0 to the last one x_T) and *backward* (from the last token x_T to the first one x_0) ways. LSTM *cell*, is a neural unit created specifically for overcoming the vanish/exploding gradient problem [2] that affects the training phase by using the backpropagation through time algorithm. The *cell* is composed of a set of *gates* (i.e input, forget, and output gate) which control the flow of information. The *forget* gate deals with choosing the information part should be kept and what should be gotten rid, the *input* gate proposes new information that is worth to be considered, and the *output* gate mix the contributes given by both the *input* and *forget* gates for creating the final cell's output. LSTM cell leverages two *feedback* loops (i.e internal and external) which allow to track the sequence of elements the cell has already analyzed through a sequence of internal states $h_1, \ldots, h_T$. The final output of the LSTM cell is its final internal state that is strictly dependent of the previous ones. The formulation of

[1] www.nltk.org

a LSTM unit, named *memory unit*, is described in by the following equations [14]:

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\ s_t &= f_t \odot s_{t-1} + i_t \odot c_t \\ h_t &= tanh(s_t) \odot o_t \end{aligned}$$

where f_t, i_t, o_t are respectively the input, forget and output gates, the $\odot$ is the element-wise multiplication, the b_f, b_i, b_o, b_c are bias vectors, while tanh is the hyperbolic tangent and sigma is the sigmoid function.

The analysis of the input text in these two opposite directions create two representation of the input sequence: straight and reversed. BiLSTM layer merges the output of the two LSTM layers into a single output by concatenating them. The final vector, if examined through the soft attention, allow the network to capture the salient words considering the input text totally.

2.2.3 Soft Attention Layer

The Soft Attention is a mechanism that weight the input sequence elements on the basis of their relevance for the classification task, suggesting on what elements leverage for classifying the input correctly. It exploits the sequence of LSTM states during the examination of the input sequence.

The attention layer's output is the *context-vector*. It is computed as the weighted sum of the *attention weights* α_t and the LSTM's states $h_0, \ldots, h_T$. The approach is described by the following formulas, considering w_α the weights matrix:

$$\begin{aligned} z_t &= h_t w_\alpha \\ \alpha_t &= \frac{e^{z_t}}{\sum_{i=1}^{T} e^{z_i}} \\ c &= \sum_{i=1}^{T} \alpha_i h_i \end{aligned}$$

In this case, the context-vector c is extended by concatenating the auxiliary features. Finally, one-dimensional vector C which contains the analysis of the BiLSTM layer and the Pragmatic features becomes the input of the next convolutional layer.

2.2.4 Convolutional Layer

We stacked three convolutional layers for the feature learning. Each convolving filter of size s slides over the input vector to compute a localized feature vector v_j for each possible word through a nonlinear activation function. For each filter, a transition matrix T is generated. Such a matrix is iteratively applied to a part of the input vector to compute the features as following :

$$v_j = f(\langle T, F_{j:j+s-1} \rangle + b_a)$$

where $\langle \cdot, \cdot \rangle$ is the inner product, $F_{g,l}$ is the part of the input vector which includes elements from position g to position l, b_a is a bias related to the specific filter, and f is a non linear function.

The output of the convolutional layers is a vector of features $v = v_1, v_2, \ldots, v_{n-s+1}$ where n is the length of the input vector.

A max-pooling layer then processes the convolutional layer's output. Such a layer extracts the largest computed feature for each filter, considering only the most relevant ones. The output layer then analyzes the output vector that included the selected features.

2.2.5 Output Layer

The output layer is a Fully Connected NN activated by Softmax. Such a layer takes as input the features extracted by the max-pooling layer. Employing the Softmax activation function computes the probability that the input text belongs to the either *satiric* or *non-satiric* class.

2.3 Parameters

Hyperparameters have been chosen empirically and taking inspiration from [13, 1]. Different tries have shown that taking a small learning rate and using a small minibatch coupled with Dropout regularization factors helps the network improve its performance by diminishing the loss. A complete list of them can be found in table 1.

Table 1. List of the model's hyperparameters.

Embedding size	300
LSTM neurons	500
Batch size	10
Convolutional layers	3
Kernel size	3
Convolutional activation function	ReLU
Dropout BiLSTM Dropout ConvNet	0.2 0.4
Optimization algorithm	Adam
Learning rate	0.0001
Dense layer neurons	350

3 Conclusion and Future Works

Satire aims at criticizing either something or someone leveraging on comedic devices. Its automatically detec-

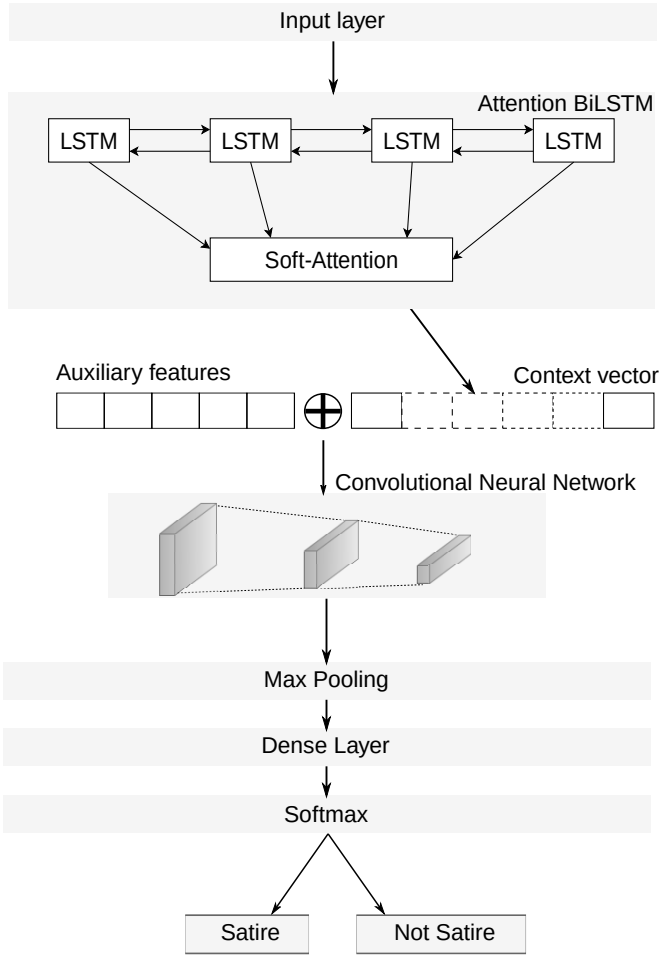

Figure 1. The representation of the Neural Network's architecture. The first layer manages the input in order to make it available for analysis. BiLSTM layer analyses the input in the forward and backward way to give a complete representation of the text. The attention mechanism is exploited for detecting the most relevant words for accomplishing the classification task. Its output is concatenated to the auxiliary features and then it is given as input to the convolutional layer. Such a layer extract prominent features, which are processed by a fully connected layer activated by softmax.

tion is a non-trivial task that have to consider the components it is composed such as *parody*, *exaggeration*, *reversal*, *irony/sarcasm* which often are related to stand-alone research topics.

In this paper, we have introduced a powerful DL model that tackles the satire detection problem by examining lexical, syntactical, and auxiliary features. To support the analysis by the system, we exploited an effective pre-trained embedding tool based on FastText.

Future work will further analyze the network's behavior by exploiting incremental data [5] and clustering [4]. Moreover, we are going to study how satire might affect the text comprehension [16] and if it might be reproduced through automatic creative processes [17].

References

[1] Teresa Alcamo, Alfredo Cuzzocrea, Giosue Lo Bosco, Giovanni Pilato, and Daniele Schicchi. Analysis and comparison of deep learning networks for supporting sentiment mining in text corpora. In *22th International Conference on Information Integration and Web-based Applications and Services (iiWAS2020)*, 2020.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[4] G. Casalino, G. Castellano, and C. Mencar. Incremental adaptive semi-supervised fuzzy clustering for data stream classification. In *2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–7, 2018.

[5] Gabriella Casalino, Ciro Castiello, Nicoletta Del Buono, and Corrado Mencar. A framework for intelligent twitter data analysis with non-negative matrix factorization. *International Journal of Web Information Systems*, 2018.

[6] Alfredo Cuzzocrea. Improving range-sum query evaluation on data cubes via polynomial approximation. *Data & Knowledge Engineering*, 56(2):85–121, 2006.

[7] Alfredo Cuzzocrea, Giosue Lo Bosco, Mariano Maiorana, Giovanni Pilato, and Daniele Schicchi. A novel approach for supporting italian satire detection through deep learning. In *Flexible Query Answering Systems - 134th International Conference,*

FQAS 2021, Bratislava, Slovakia, September 19–24, 2021, Proceedings, Lecture Notes in Computer Science. Springer, 2021.

[8] Alfredo Cuzzocrea and Ugo Matrangolo. Analytical synopses for approximate query answering in olap environments. In *International Conference on Database and Expert Systems Applications*, pages 359–370. Springer, 2004.

[9] Alfredo Cuzzocrea, Rim Moussa, and Guandong Xu. Olap*: effectively and efficiently supporting parallel olap over big data. In *International Conference on Model and Data Engineering*, pages 38–49. Springer, 2013.

[10] Alfredo Cuzzocrea, Domenico Saccà, and Paolo Serafino. A hierarchy-driven compression technique for advanced olap visualization of multidimensional data cubes. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 106–119. Springer, 2006.

[11] Alfredo Cuzzocrea and Paolo Serafino. Lcs-hist: taming massive high-dimensional data cube compression. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 768–779, 2009.

[12] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[13] Le Hoang Son, Akshi Kumar, Sangwan Raj Saurabh, Anshika Arora, Anand Nayyar, and Mohamed Abdel-Basset. Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE Access*, 7:23319–23328, 2019.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Rosalind W Picard. *Affective computing*. MIT press, 2000.

[16] Daniele Schicchi, Giosué Lo Bosco, and Giovanni Pilato. Machine learning models for measuring syntax complexity of english text. In *Biologically Inspired Cognitive Architectures Meeting*, pages 449–454. Springer, 2019.

[17] Daniele Schicchi and Giovanni Pilato. Wordy: a semi-automatic methodology aimed at the creation of neologisms based on a semantic network and blending devices. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 236–248. Springer, 2017.

[18] Aman Sinha, Parth Patekar, and Radhika Mamidi. Unsupervised approach for monitoring satire on social media. In Prasenjit Majumder, Mandar Mitra, Surupendu Gangopadhyay, and Parth Mehta, editors, *FIRE '19: Forum for Information Retrieval Evaluation, Kolkata, India, December, 2019*, pages 36–41. ACM, 2019.