A Novel Approach for Supporting Italian Satire Detection Through Deep Learning

Gabriella Casalino[1], Alfredo Cuzzocrea[2(✉)], Giosué Lo Bosco[3,5], Mariano Maiorana[3], Giovanni Pilato[4], and Daniele Schicchi[5]

[1] Department of Computer Science, University of Bari, Bari, Italy
gabriella.casalino@uniba.it
[2] iDEA Lab, University of Calabria, Rende, Italy
alfredo.cuzzocrea@unical.it
[3] Department of Mathematics and Computer Science, University of Palermo, Palermo, Italy
giosue.lobosco@unipa.it, mariano.maiorana@community.unipa.it
[4] ICAR-CNR, National Research Council of Italy, Palermo, Italy
giovanni.pilato@cnr.it
[5] Department of Science for Technological Innovation, Euro-Mediterranean Institute of Science and Technology, Palermo, Italy
danieleschicchi@iemest.eu

Abstract. Satire is a way of criticizing people (or ideas) by ridiculing them on political, social, and morals topics often used to denounce political and societal problems, leveraging comedic devices such as parody exaggeration, incongruity, etc.etera. Detecting satire is one of the most challenging computational linguistics tasks, natural language processing, and social multimedia sentiment analysis. In particular, as satirical texts include figurative communication for expressing ideas/opinions concerning people, sentiment analysis systems may be negatively affected; therefore, satire should be adequately addressed to avoid such systems' performance degradation. This paper tackles 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 *IlFattoQuotidaino*, and significant Italian newspapers. Experiments show an optimal performance achieved by the network capable of detecting satire in a context where it is not marked.

Keywords: Satire detection · Deep learning · Natural language processing

1 Introduction

Satire is a way of criticizing people (or ideas) by ridiculing them on political, social, and morals topics. Most of the time, such a language form is utilized to influence people's opinions. It is a figurative form of language that leverages

T. Andreasen et al. (Eds.): FQAS 2021, LNAI 12871, pp. 170–181, 2021.
https://doi.org/10.1007/978-3-030-86967-0_13

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. Another yet interesting paradigm consists in coupling these aspects with the emerging *big data trend*, as to cover some advanced topics such as graph analytics (e.g., [10]) and cybersecurity (e.g., [7]).

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 sarcasm/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* [17]. 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 *Il Fatto Quotidaino*, 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 Il Fatto Quotidaino. In fact, in this special case, news are so realistic that they seem to be true [17]. An autonomous way of detecting satire might help computers interpret human interaction and notice its emotional state, improving the human-computer experience.

The structure of this paper is organized as follows: in Sect. 2 we present the satire detection topic showing some related works. Section 3 describes the

methodology we used to tackle the task and it contains details for replicating the system. Finally Sects. 5 and 6 respectively show experiments and comments on results. Finally, we give the conclusion and future works in Sect. 7.

2 Related Works

Satire is a fascinating topic extensively studied in literature [13,16,18]. Its automatic detection is an emerging task that has been introduced by Burfoot and Baldwin [6], where they presented their first solution: a Support Vector Machine-based system that relied on bag-of-words features.

Although the importance of the task, it has been tackled mainly for the English language. The Italian one, like other languages, have suffered the prominence of such a language. Few works tackle the satire detection problem for Italian text and a lack of resources to support the development of a computational system for such a problem. Nonetheless, Barbieri and Saggion [2] have proposed a language-independent framework for detecting satiric Tweets. They harvested data from popular satirical Twitter accounts. Then they have trained a Support Vector Machine for the classification task by taking into account both language-independent intrinsic word features and language-dependent word-based features. The experimentation reveals the effectiveness of their system for English, Spanish and Italian Tweets. To the best of our knowledge, it is the most relevant work that tackles the Italian language problem, but it differs from ours because of the different type of text we address. Our system analyzes full articles extracted from online *satirical journals* that are different from Tweets in several respects.

A key element of satire is *irony*. Automatically detecting irony is a related research topic that has been widely studied in the past [1,3,11]. The literature concerned such a task is a good support for the satire detection one, but it is limited because of the principal characteristic of the satire: it is utilized for mocking something of someone. Therefore, they are different tasks that have to be dealt with separately.

3 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 (Sect. 3.1), the model's architecture for tackling the task and its tuning (Sect. 3.2 and 3.3). Le Hoang Son et al. [14] 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 Sect. 3.2.

3.1 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 [5,12]. 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.

3.2 Architecture

The network's architecture is inspired from the one presented by Le Hoang Son et al. [14], 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 Fig. 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.

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 [4] 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 a LSTM unit, named *memory unit*, is described in by the following equations [15]:

[1] www.nltk.org.

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

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.

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.

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.

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.

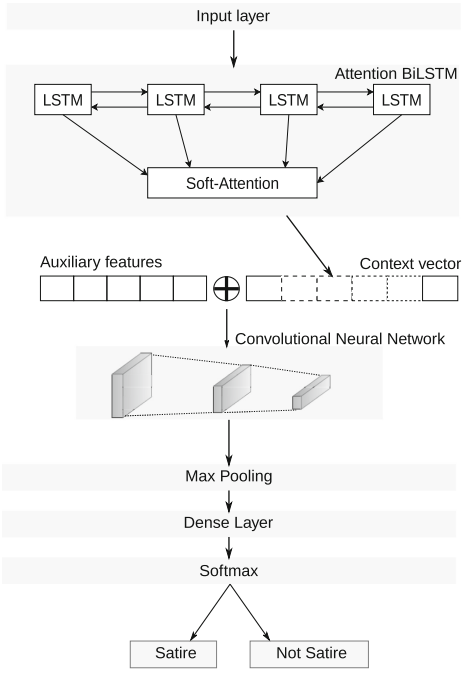

Fig. 1. The representation of the Neural Network's architecture. The first layer manages the input in order to make it available for analysis. BiLSTM layer analyses the input in the forward and backward way to give a complete representation of the text. The attention mechanism is exploited for detecting 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.

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	0.2
Dropout ConvNet	0.4
Optimization algorithm	Adam
Learning rate	0.0001
Dense layer neurons	350

3.3 Parameters

Hyperparameters have been chosen empirically and taking inspiration from [1, 14]. 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.

4 Experiments

4.1 Corpus

Satiric texts has been extracted from two famous Italian sites named *Lercio*[2] and *IlFattoQuotidaino*[3]. Both of them include satiric news written according to the *news satire* format.

The gathering process has been carried out by exploiting the RDF Site Summary (RSS): a technology that makes it possible to distribute web contents in a standardized, computer-readable format. We collected 7354 articles (2195 full articles and 5159 texts related to satiric *memes*) from Lercio and 943 full articles from IlFattoQuotidaino. Afterward, we used the RAKE algorithm [19] to extract a set of keywords from the satiric texts. Finally, such keywords have been used to filter newspaper articles from the *Event Registry*[4] site. A complete set of the RAKE's parameters used are given in Table 2.

Event Registry collects and annotate news published by over 30'000 news publishers, and it provides tools for selecting articles on the basis of *topic*, *language*, *keywords*, *article location* and so on. We focused on *Italian current news* that best matched the keywords extracted from the satiric news.

The final dataset is a mixture of satiric news extracted from Lercio and IlFattoQuotidaino coupled with Newspaper articles which concern similar topics.

[2] www.lercio.it.
[3] www.ilfattoquotidaino.it.
[4] https://eventregistry.org/.

Table 2. RAKE's parameters used for extracting keywords from the satiric texts.

Parameter	Value
Mininum characters	3
Maximum words	2
Language	It
Mininum frequency	2
Stopwords	NLTK stopwords

Table 3. Averaged system results on Lercio and regular newspapers.

Measures	Epochs			
	5	10	15	20
Accuracy	.9445	.9620	**.9885**	.9855
Recall	.9620	.9820	.998	**.9920**
Precision	.9294	.9442	**.9793**	.9792
F1-Score	.9455	.9628	**.9886**	.9856

5 Experiments

The network has been trained and tested exploiting a dataset composed of 1'000 Lercio's full articles and 1'000 full regular articles extracted from newspapers. We leveraged on cross-validation methodology K-Fold with $K = 5$. We evaluate the network's behavior for each fold by using Accuracy, Recall, Precision, and F1-Score. The final network's behavior has been evaluated by averaging the partial results achieved for each fold. Moreover, we used 400 articles extracted from IlFattoQuotidaino *only* for testing.

Table 3 shows the results computed considering the averaged performance of the network on Lercio and regular newspapers on varying of the number of epochs. Similarly, table 4 contains the performances of the network on IlFatto-Quotidaino.

5.1 Baseline Models

The main model was compared with a set of competitors derived rearranged its layers. To doing so, it makes clear whether the problem's complexity is overestimated leaving room for the usage of fewer complex models. We trained and tested five different NNs systems according to the methodology used for the main model. We assembled the baselines leveraging on LSTMs, Convolutional NNs, Attention Mechanism, and Fully Connected NNs as follows:

- *sAttBiLSTMConvNet* is the main model as described in Sect. 3.2;
- *sBiLSTMConvNet* is structured as the main model except for the attention mechanism. The auxiliary features are concatenated to the output of the

Table 4. Averaged system results on IlFattoQuotidaino and regular newspapers.

Measures	Epochs			
	5	10	15	20
Accuracy	.9512	.955	.9887	**.9912**
Recall	.9325	**.9975**	.9975	.990
Precision	.9688	.9194	.9803	**.9925**
F1-Score	.9503	.9569	.9888	**.9912**

Table 5. Averaged systems results on Lercio and regular newspapers: models comparison.

Model	Accuracy	F1-score
sAttBLSTMConvNet	**.9885**	**.9886**
sBiLSTMConvNet	.9445	.9442
S-LSTM	.5930	.7107
BiLSTM	.9580	.9670
BiLSTM-Att	.9305	.9344
SVM	.8389	.8579

BiLSTM layer, and the resulting vector is given as input to the convolutional layer;

- *S-LSTM* is a simple NN that relies on the Input layer that manages the input articles linked to an LSTM cells layer. The last layer analyzes the input sequence only in the forward direction. The final output is then processed by a fully connected NN activated by softmax;
- *BiLSTM* is a NN composed of the Input layer and a Bidirectional LSTM layer. The output of such a layer is then given as input to the final fully connected layer that is activated by the softmax activation function. Auxiliary features are concatenated to the output of the Bidirectional LSTM layer.
- *BiLSTM-Att* is a BiLSTM that exploits the Attention Mechanism and that output the final result through the fully connected layer activated by softmax. Auxiliary features are concatenated to the context vector.

Moreover, we used a Support Vector Machine for tackling the satire classification problem. In this case, the articles are represented as the averaged vector computed exploiting the tokens embedding described in Sect. 3.1.

Table 5 compares the results achieved by the models on Lercio considering 5-Fold cross-validation. It contains the best performance achieved by each model. For what concern the *sAttBLSTMConvNet* we have chosen the model trained for 15 epochs.

6 Discussion

The testing phase is made harder by the lack of data and benchmarks that standardize experimentation. Furthermore, few models tackle the same tasks; therefore it is hard to carry on an extended comparison. Although such difficulties, the cross-validation methodology allows us to measure the network's performances and generalizing capabilities beyond the training set. Moreover, many different full models' are tested in the same way.

Experiments' verdict outlines a relevant performance achieved by the sAttBLSTMConvNet for tackling the task of *satire detection*. The model achieves 98.9% of accuracy and F1-Score on Lercio achieving respectively +3% and +2% compared to the second-best model *BiLSTM*.

LSTM is the worst model; it is not capable of detecting the satire as well as other models, suggesting the need for more efficiency to approximate the input space and to find the right threshold between approximation and generalization. The other models achieve relevant performance, which is almost comparable to the ones achieved by sAttBLSTMConvNet. Experiments show the importance of the attention mechanism, which get decreasing the sBiLSTMConvNet's performance. Surprisingly, a simpler model as BiLSTM achieves the second most relevant results.

Another significant contribution is given by relating the sAttBLSTMConvNet's performance to the analysis of the corpora. Lercio and IlFattoQuotidiano treat satire in different ways: the former makes use of *exaggeration* heavily, the latter tends to use a *soft* satire. From a human-being perspective, satire is easy to recognize in Lercio but more complex in IlFattoQuotidaino since their authors write articles similar to those published in regular newspapers. We believed that the satire included in the IlFattoQuotidaino news could be detected mainly by human-beings since their articles are often mistaken for *real* news or *fake* news. Results in Table 4 show a different trend, further demonstrating the primary model's effectiveness.

7 Conclusion and Future Works

Satire aims at criticizing either something or someone leveraging on comedic devices. Its automatically detection 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. The system has been widely tested, and results show its capability to detect the soft satire and the more marked one.

Future work will further analyze the network's behavior by exploiting incremental data [9] and clustering [8]. Moreover, we are going to study how satire might affect the text comprehension [20] and if it might be reproduced through automatic creative processes [21].

Acknowledgement. Gabriella Casalino acknowledges funding from the Italian Ministry of Education, University and Research through the European PON project AIM (Attraction and International Mobility), nr. 1852414, activity 2, line 1.

References

1. Alcamo, T., Cuzzocrea, A., Lo Bosco, G., Pilato, G., Schicchi, D.: 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. Barbieri, F., Ronzano, F., Saggion, H.: Do we criticise (and laugh) in the same way? Automatic detection of multi-lingual satirical news in Twitter. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
3. Barbieri, F., Saggion, H.: Automatic detection of irony and humour in Twitter. In: ICCC, pp. 155–162 (2014)
4. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994). https://doi.org/10.1109/72.279181
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. CoRR abs/1607.04606 (2016)
6. Burfoot, C., Baldwin, T.: Automatic satire detection: Are you having a laugh? In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pp. 161–164 (2009)
7. Campan, A., Cuzzocrea, A., Truta, T.M.: Fighting fake news spread in online social networks: actual trends and future research directions. In: 2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, 11–14 December 2017, pp. 4453–4457. IEEE Computer Society (2017)
8. Casalino, G., Castellano, G., Mencar, C.: Incremental adaptive semi-supervised fuzzy clustering for data stream classification. In: 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), pp. 1–7 (2018). https://doi.org/10.1109/EAIS.2018.8397172
9. Casalino, G., Castiello, C., Del Buono, N., Mencar, C.: A framework for intelligent twitter data analysis with non-negative matrix factorization. Int. J. Web Inform. Syst. **14**(3), 334–356 (2018)
10. Cuzzocrea, A., Song, I.: Big graph analytics: the state of the art and future research agenda. In: Proceedings of the 17th International Workshop on Data Warehousing and OLAP, DOLAP 2014, Shanghai, China, 3–7 November 2014, pp. 99–101. ACM (2014)
11. Di Gangi, M.A., Lo Bosco, G., Pilato, G.: Effectiveness of data-driven induction of semantic spaces and traditional classifiers for sarcasm detection. Nat. Lang. Eng. **25**(2), 257–285 (2019). https://doi.org/10.1017/S1351324919000019
12. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
13. Highet, G.: Anatomy of Satire. Princeton University Press, Princeton (2015)
14. Hoang Son, L., Kumar, A., Raj Saurabh, S., Arora, A., Nayyar, A., Abdel-Basset, M.: Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. IEEE Access **7**, 23319–23328 (2019)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

16. Hodgart, M.J.C.: Die Satire, vol. 42. Transaction Publishers (1969)
17. Picard, R.W.: Affective Computing. MIT Press, Cambridge (2000)
18. Pollard, A.: Satire, vol. 6. Taylor & Francis (2017)
19. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic keyword extraction from individual documents. Text Min.: Appl. Theory **1**, 1–20 (2010)
20. Schicchi, D., Lo Bosco, G., Pilato, G.: Machine learning models for measuring syntax complexity of English text. In: Samsonovich, A.V. (ed.) BICA 2019. AISC, vol. 948, pp. 449–454. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-25719-4_59
21. Schicchi, D., Pilato, G.: WORDY: a semi-automatic methodology aimed at the creation of neologisms based on a semantic network and blending devices. In: Barolli, L., Terzo, O. (eds.) CISIS 2017. AISC, vol. 611, pp. 236–248. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-61566-0_23