Journal of Data Intelligence, Vol. 2, No. 4 (2021) 448–461
© Rinton Press

SENTIMENT MINING AND ANALYSIS OVER TEXT CORPORA VIA COMPLEX DEEP LEARNING NEURAL ARCHITECTURES

TERESA ALCAMO
DMI Department, University of Palermo
Palermo, Italy
teresa.alcamo@community.unipa.it

ALFREDO CUZZOCREA
iDEA Lab, University of Calabria
Rende, Italy
alfredo.cuzzocrea@unical.it

GIOVANNI PILATO
ICAR - CNR, Italian National Research Council
Palermo, Italy
giovanni.pilato@icar.cnr.it

DANIELE SCHICCHI
ITD - CNR, Italian National Research Council
Palermo, Italy
daniele.schicchi@itd.cnr.it

We analyze and compare five deep-learning neural architectures to manage the problem of irony and sarcasm detection for the Italian language. We briefly analyze the model architectures to choose the best compromise between performances and complexity. The obtained results show the effectiveness of such systems to handle the problem by achieving 93% of F1-Score in the best case. As a case study, we also illustrate a possible embedding of the neural systems in a cloud computing infrastructure to exploit the computational advantage of using such an approach in tackling big data.

Keywords: Sentiment Mining; Sentiment Analysis; Text Corpora; Deep Learning; Neural Architectures; Complex Architectures

1. INTRODUCTION

With the growth of Web 2.0 and the development of social media, sentiment analysis has played a growing and increasingly important role. The sentiment expressed in comments, reviews, feedback, and so on can provide valuable reference points for many different purposes ranging from marketing to political issues.

Sentiment analysis is a research field which has been deeply studied in the past. Its main objective is to analyse people's opinions, sentiments, attitudes and emotions conveyed by written texts. In fact, through microblogs (e.g., Twitter) users can express their opinions over many different topics, make suggestions, and communicate positive or negative sentiments regarding products of every-day life. The relevance of sentiment analysis has heightened simultaneously with the growth of active web users, who writing their thoughts allows investigating the people common opinion concerning the different type of subjects such as politics and commercials. Another relevant problem in this context is represented by *fake news* (e.g.,

[8]) that can arise in such posts. While this problem is interesting at now, it is not central in our research, thus left as future work.

The sentiment can usually be divided into positive and negative. From this perspective, it can be seen as a classification task with three values: positive, negative, or neutral.

The huge amount of data has made impractical the handmade analysis of the texts, therefore researchers have been focused their attention on developing efficient methodologies to make automatic this task. Rudimentary tools analyse the text literally meeting difficulties when the real meaning of the text is hidden by means of linguistic devices such us *irony* and *sarcasm.* The classical definition of such figures of speech is "saying the opposite of what you mean" [7]. There is a little difference between irony and sarcasm since the latter is usually used for taunting or humiliating someone of something, thus it can be considered as a sub-class of irony.

Irony and sarcasm are two complex linguistic phenomena that are present in everyday life interaction[16]. Sarcasm and irony are very similar; however, sarcasm has a specific target who is the object of the sarcastic statement, while irony does not have such a target [25].

On the other hand, they are difficult to tackle from the computational perspective for automated text understanding and human-machine interaction.

Both the situation and the speaker's mental state, like attitude, knowledge, and intentions, play a key role in irony and sarcasm detection: they should be taken into account to understand sarcastic content in a sentence [29] and they could be useful to improve the effectiveness of conversational agents provided of associative and intuitive capabilities [38]. Moreover, sarcasm detection in written texts is an even more difficult task even for humans [39], since the tone of the voice usually suggests the sarcastic intent, which is lost in written texts.

Thus, the automatic identification of both irony and sarcasm has become a critical task to tackle because it affects the performances of systems that provide sentiment analysis.

In this paper, we tackle the problem of irony-sarcasm detection by exploiting Deep Learning (DL) methodologies. Our study is focused on the Italian language since, to the best of our knowledge, there are only a few works available through the relevance of the topic. We compare many DL systems by combining architectures such as Recurrent Neural Network (RNN), Bi-direction RNN, Convolutional Neural Network (CNN) and Attention mechanism. In details, we started exploiting a DL system [35] created for dealing with the sarcasm detection task and then we changed the network architecture by subtracting layers in order to evaluate the degree of influence of such components to the system performances. The presented systems are trained by exploiting a corpus composed of ironic and non-ironic examples extracted respectively from *Spinoza* and from seven famous newspapers, Twitter accounts. The experiments are computed by exploiting repeatedly the K-Fold cross-validation methodology to make balanced the dataset. Results show high suitability of such systems to face the problem by achieving 93% of F1-Score in the best case.

The paper is structured as follows. In section 1 a case of study on possible embedding in a cloud computing infrastructure is illustrated; in section 2the literature related to the subject is briefly reviewed; in section 3the core modules (i.e pre-processing and the model) and parameters which characterize the presented systems are introduced; section 4 and section 5 respectively describe the methodology exploited to test our systems and the analysis of the results. Finally, conclusions and future perspectives are given in section 6.

2. CASE STUDY: FRAMEWORK IMPLEMENTATION ON TOP OF CLOUD COMPUTING INFRASTRUCTURES

The system can find a natural field of application in a big data context. This can be accomplished by integrating the system within the core layer of popular big data processing platforms, such as Hadoop, Spark, etc., thus building effective, ad-hoc big data architectures for supporting sarcasm detection though a neural approach.

Figure 1 shows a reference architecture meant to pursue this goal. The proposed architecture can be mapped as a traditional multi-layer software solution, where the following layers can be identified:

Data Source Layer: this layer contains the big data sources that produce the input to be provided to one or more instances of the proposed system. Among the data sources we can mention documents repositories, social networks, web sites and blogs: they are just example sources of realistic application scenarios where the architecture can be effectively used;

Cloud-Based Prediction Layer: in this layer, the classifier systems are deployed on top of a Cloud environment, in order to exploit the versatility of modern Cloud computing infrastructures as well as big data processing platforms. A classifier instance is replicated on every Cloud node as to enforce distributed and parallel processing paradigms. A graph is the outcome of each node of this layer, where each node is the chunk text associated with the label "sarcastic" or "non-sarcastic" and two nodes are linked by an edge if a semantic similarity, computed through a sub-symbolic score function, is above a given threshold experimentally fixed.

Data Staging Layer: this layer identifies the software layer where common and established big data processing platforms are used to stage big data repositories. Furthermore, it gives the proper big data services to support the whole analytics architecture. At this layer, the underlying big data processing platform not only supports the collection of big data sources and proper prediction via the classifier system (thanks to a core integration with the Cloud computing infrastructure) and a semantic similarity between text items, but it also supports the next layer of the architecture where proper big data analytics tasks are delivered;

Big Data Analytics Layer: this layer contains the software components that are dedicated to completely support big data analytics tasks over the irony/sarcasm prediction data produced by the respective layer. Here, several solutions can be explored, including visual big data analytics and so forth.

The proposed Cloud-based architecture makes it possible to realize more efficient applications, capable to be applied on very large datasets, taking advantage of the Cloud computing paradigm. To this end, the most critical aspect consists in how to make "Cloud-able" the classification technique. Thanks to the Cloud support, the system can thus achieve higher degree of scalability over big data that populate real-life applications, which is a relevant feature to take into consideration.

The content of each source is filtered according to a given semantic query and the textual data is then processed by a node containing a specific neural network trained for recognizing sarcasm or irony arising from the analyzed text. Each network can be trained, or updated, independently from the other networks present in the other nodes. The results obtained as a classification outcome are then forwarded and analyzed through the use of the cloud infrastructure, in order to realize, for example a decision support system or a recommendation system.

A schema of the whole approach is depicted in Figure 1

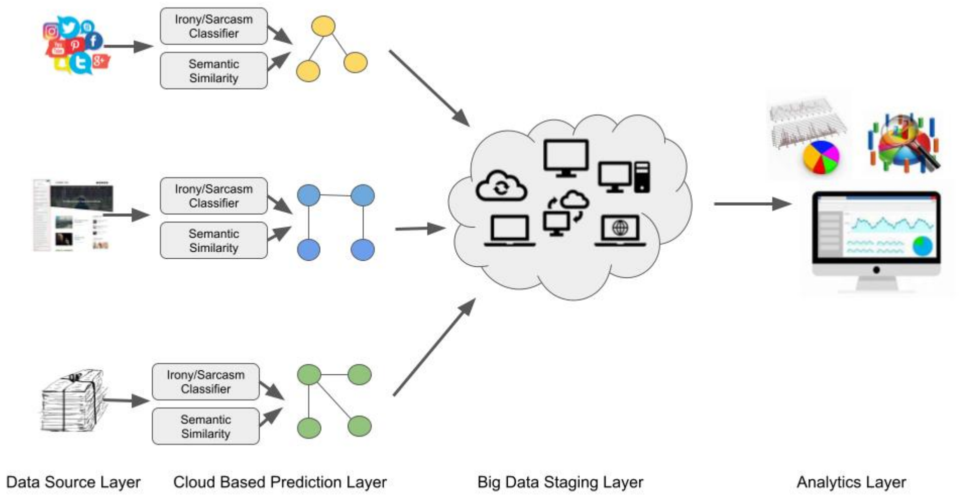

Fig. 1. A schema of the possible implementation of the system embedded in a cloud computing infrastructure.

3. RELATED WORK

The increasing opportunities for accessing technological facilities have led to an enrichment of the web textual contents due to the growth of internet users. Every day, billions of different types of texts are published on the Internet, and the advent of social media has made this process more affordable [17] [36]. In the last years, researchers have investigated the usefulness of internet data by harvesting online resources to build datasets. Such data has been used for creating models capable of accomplishing different tasks resulting in systems capable of achieving high performances.

One of the first attempts to sarcasm detection by using web data is described in [37]. Authors exploit data from Twitter, and Amazon reviews demonstrate the effectiveness of automatically-crawled data to tackle the task. Most of the following approaches have in common the use of the Twitter platform since it is possible to exploit hashtags to filter messages with a specific ironic or sarcastic content. Twitter allows users to write a short text of a maximum of 280 characters, which can be enriched using *hashtags*, which helps to specialize in the intent of the message (e.g., irony, sarcasm, critique, approval).

First achieved results on the sarcasm detection by means of Twitter data have shown the difficulty of the task even for human people. In [22] it has been created a corpus with 2700 tweets labeled in a balanced way as *sarcastic*, *positive sentiment* and *negative sentiment*. It experimented the capability of human being for sarcasm detection by presenting part of such corpus to a group of human judges. The results showed both low agreement and low accuracy in the classification task by proving the complexity of the task. Furthermore, in [30], a set of tweets representative of a sarcastic situation (i.e, express a positive sentiment in a negative situation) and tagged as #sarcastic have been presented to a group of people omitting the hashtag. It was found that less than half of the tweets were classified as sarcastic by human judges, giving an additional proof of the task difficulty. In [32], a study on the effort of the sarcasm detection reveals that the task is easier for documents because the analysis of the context provides further information which simplifies the task.

The automatic detection of sarcasm has been tackled by exploiting different approaches. In [28] has been proposed a methodology based on both the combination of a set of rules for sarcasm identification and hashtags. Experiments concerned a corpus composed of general-topic tweets resulting in a precision of 96.41% . In [4] it has been studied how the inclusion of the author description and the user features could affect the system performances. Experiments showed a positive contribution to the augmented data by obtaining a significant improvement in accuracy. In [16], Latent Semantic Analysis (LSA) and the Truncated Singular Value Decomposition (T-SVD) have been utilized for classifying sarcasm using a Support Vector Machine (SVM). The system was tested for five different corpora reporting a precision up to 71%. The problem has been widely examined as a sub-task of the sentiment analysis, which has been deeply studied in the past. In [2] was developed an ML model for the sentiment analysis of tweets regarding football employing three classifiers: SVM, multinomial Navies Bayes and random forest. In [31], support vector regression is utilized for predicting vehicle sales using the analysis of Twitter data and stock market values. In [6] it is proposed a multi-class sentiment analysis system which aims to identify precisely the sentiment within the tweet by associating labels representing all the existing sentiments.

The relevant results achieved by Deep Learning on Natural Language Processing (NLP) have inspired researchers to use such methodologies for tackling the subject task. In [20], a specific type of neural network based on convolutional and recurrent layers have been proven to achieve high performances in classification task overcoming recursive SVM. Such work has contributed to built the model presented in [35] for improving sentiment analysis and sarcasm detection. In [19], a pre-trained DL model is used to extract sentiment, emotion, and personality features for developing a sarcasm detection system. For what concerns the Italian language, most of the work has been tackled the automatic detection of irony, which is a linguistic device close to the sarcasm one. It is possible to assume that sarcasm is a sub-class of irony. In [5], features such as frequency of words, synonyms, the ambiguity of a word, parts of speech are used to train a Decision Tree algorithm for the irony detection. EVALITA-2018 has challenged researchers to develop an automatic system for *a)* detecting autonomously the irony and *b)* for detecting sarcasm as sub-class of irony. In [21], a bidirectional recurrent neural network based on Gated Recurrent Unit and convolutional layers have been presented for the competition. It achieves a 61% of F1-Score for the task a) while a 46.5% for the second task.

4. AN INNOVATIVE METHODOLOGY FOR SUPPORTING SENTIMENT MINING AND ANALYSIS BASED ON DEEP LEARNING NEURALE ARCHITECTURES

4.1. *Preprocessing*

The preprocessing phase is an essential phase of every ML-based system whose main task is to make the input suitable for the analysis by the model. Such a task takes place by transforming the input data in order to structure it coherently with the model features, preserving as much as possible the original amount of information, and cutting off biased elements.

In the NLP, a common way to organize the input text is as a *sequence* of *tokens*: a token can be thought as a base unit of the text and the policy to select them may affect the system performances. An efficient *tokenization* procedure should identify as tokens exclusively parts of the input which are rich in information in order to allow the system to achieve its objective. In this case, we have chosen to select *words* and *punctuation symbols* as tokens. This choice allows us to have a complete representation of the input text taking into account also its syntactical structure.

After the *tokenization*, a proper padding operation has been performed to make uniform the length of the tweet to the average length (i.e 15 tokens). Then the input data is represented in a vectorial form by meeting the specifications of the core model. Such representation is turned out by exploiting a pre-trained version of an efficient *embedding* tool called FastText [24] a library for the efficient learning of word representations and sentence classification. For each input phrase, every token is mapped to a 300-dimensional space by organizing it as a sequence of multidimensional vectors. Finally, for each input phrase, the output of the preprocessing phase is a matrix of dimension $L\ x\ N$, L is the number of tokens, and N is the *embedding* dimension (i.e., 300).

4.2. *Main Model*

The main model (from now on BILSTM+CONV) exploited to tackle the problem is similar to the one introduced in [35]. It is a Deep Neural Network that mixes both the Recurrent and Convolutional architectures and the Attention mechanism. The peculiarity of the model is to take into account a specific set of computed punctuation-based features that have been proven helping to improve the system performances. Such features (i.e., Number of exclamation marks (!) in the tweet, Number of question marks (?) in the tweet, Number of periods (.) in the tweet, Number of capital letters in the tweet, Number of uses of *or* in the tweet) often indicate a figurative text and symbolic clued which help to recognize if the tweet is sarcastic.

The matrices of real numbers outputted by the Preprocessing phase are given as input to the *input* layer of the Model architecture. This layer organize the data in order to allow the analysis by the next layer that is based on a Long Short Term Memory (LSTM) units [23].

LSTM is an RNN that has been created to overcome the weakness of the classical RNN systems. Researchers have developed the LSTM formulation to face a well-known problem named *vanish gradient* [23], which brings the network training to an impasse. The LSTM unit examines a sequence element by element. It changes its *internal state* (hidden state) based on what it has already analyzed, the final output may be either a single vector representing the final network state or a vector of hidden states assumed during the computation time T: $Y = [h_1, h_2, \ldots, h_t]$, $t \in T$.

The formulation of a LSTM unit, named *memory unit*, is described in by the following equations:

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

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 σ is the sigmoid function.

The Bidirectional LSTM layer (BiLSTM) combines the hidden states of two different LSTM layers, the first one which analyses the sequence in a straightforwardly (from the beginning to the end) and the second one which examines the sequence in a reverse manner (from the end to the beginning). In our case, the output of such layer is the concatenation of the hidden states assumed by the layer for each time step $t \in T$:

$$Y = [h_{1,f}, h_{2,f-1}, h_{3,f-2}, \ldots, h_{f,1}] \tag{2}$$

where $h_{i,j}$, $i, j \in T$, is the concatenation of the hidden state at time i of the straightforward examination, and the hidden state at time j of the reverse analysis, f is the last time step.

The output of BiLSTM is given as input to the Attention layer, which exploits the sequence vectors Y for discovering the relevance of the input sequence elements. The Attention output (context-vector), is computed as a weighted sum of the *attention weights* and the output vectors Y. The mathematical formulation is introduced in [18], and it is inspired by [3, 40]. The *context-vector* is then concatenated to the computed auxiliary features in order to create a one-dimensional vector C which contains the analysis of the BiLSTM layer extended by Pragmatic features.

The last part of the model is characterized by a series of Convolutional layers (convNet) which aims to learn the features exploitable for the classification of the input data using a Dense layer.

The vector resulting from the concatenation of context-vector and the auxiliary features is provided as input to the convNet. Each trained involving filter is convolved with C over a single dimension in order to compute the local features for each possible word by exploiting a non-linear activation function. Each convolutional layer output is given as input to the next one except for the last output, for which is applied a max-pooling operation.

Finally, the max-pooling output becomes the input of a Dense layer activated by the softmax function, which gives as output the probability that the input text belongs to either the *sarcastic* or *non-sarcastic* class.

A complete schema of the model architecture is given in Figure 2.

4.3. *Secondary Models*

The main model has been compared with a set of DL systems inspired by the main one but removing specific DL layers to carry on performance analysis. We have created four alternative architectures at different degrees of complexity[a]by investigating, beyond the contribution of layers, the setting that allows the network to infer the input space as best as possible. The architectures we propose examine only tokens as described in section 4.1without considering additional features. We avoided to use auxiliary features in secondary models because we want to investigate how different types of DL models can tackle such a task by leveraging only on words and punctuation symbols, formulating the research question if the auxiliary features are valuable. A description of secondary models is given below.

S-LSTM: S-LSTM stands for *simple LSTM*, and it is a DL architecture composed of two layers. The first one is an RNN composed of LSTM neural units that deals with examining the input sequence by outputting a vector computed based on both element and their arrangements. The final output of such layer is derived by exploiting the internal feedback loops of the LSTM that allow the network to consider both long-term (i.e among long distance elements) and short-term (i.e among short distance elements) relationships. The second layer is a fully connected neural network composed of two neurons activated by softmax function. The softmax function, applied to the output of the network, maps its value in a range between 0-1 in order to its sum is the unit. This means that the network's final output is the probability that the input text belongs to one of the two classes.

ATT-LSTM: ATT-LSTM stands for *attentional LSTM.* This architecture is comparable to the S-LSTM one, in fact, it differs only for the Attention mechanism added after the LSTM layer. The attention layer computes attention weights that support the network to weigh the input sequence elements efficiently to perform the subject task. In this case, the attention mechanism is the same exploited for the main model (see sec.), and it leverages only the recurrent neural network's internal states [18]. ATT-LSTM is an architecture utilized in the past for tackling several task such as *autonomous text complexity evaluation* for Italian and English language [34][33].

BI-LSTM: BI-LSTM stands for *bidirectional LSTM.* It is composed of two layers; the first one is a Bidirectional LSTM, composed of two LSTM layers that relate elements of the input sequence respectively in a forward and a backward way. Examining the input sequence in these opposite directions allows increasing the amount of input information available to analyze. Moreover, it supports the recurrent neural network to track information regarding the past and the future that otherwise would be harder to *remember* by the network. The Bidirectional LSTM outputs two vectors that have to be merged by choosing a *merging policy*, in this case, we have chosen to concatenate them. The computed output will be then passed as input to a dense layer composed of two neurons and activated by using softmax function.

BI-ATT-LSTM: BI-ATT-LSTM stands for *bidirectional attentional LSTM* and it is a BI-LSTM enriched by an attention layer. This network architecture is the most powerful within the secondary models capable of inferring the input space accurately though it might cause overfitting. The rationale of the BI-ATT-LSTM is to relate the methodology of BI-LSTM layer to support the network to overcome its memory limits with the attention mechanism that weight the internal state taken by the LSTMs during the sequence analysis. It has been used in the past to tackle problems of different application domains [18] and its training should

[a]complexity is related to the number of stacked layers and the number of parameters to be learned.

be carried on carefully to allow the network to well generalize the solution for the given task.

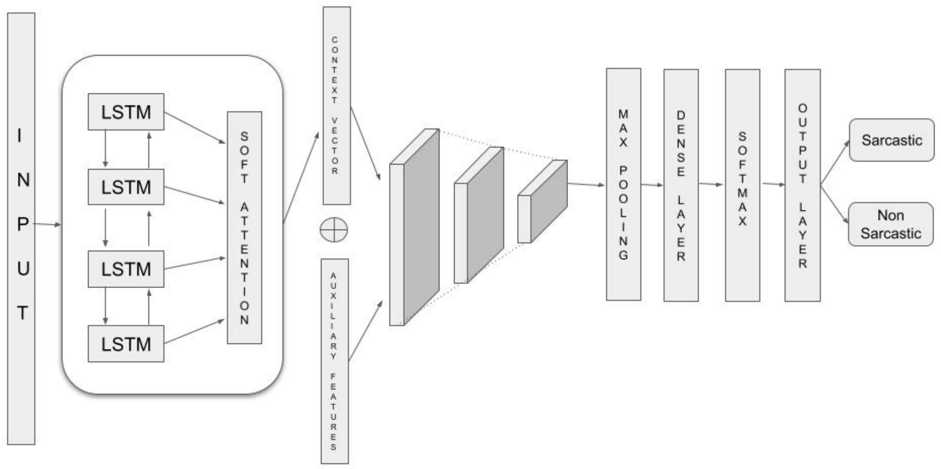

Fig. 2. Visual representation of the Model architecture. From the top down: the input Layer, the Bidirectional LSTM layer, Attention layer, the concatenation of the auxiliary features to the context-vector, a series of Convolutional layers, Max pooling, Dense layer activated by the softmax function and the output layer. The $\bigoplus$ is the concatenation operation.

4.4. *Parameters*

The parameters of the main DL system have been computed empirically and by taking inspiration from [35]. The secondary models parameters are set to the same values of the main model ones.

In particular, the BiLSTM has a number of LSTM cells equal to 500. The weights of the network are initialized randomly by using a Gaussian distribution of the random values to be generated. A standard deviation of 0.01 has been set. The dropout regularization technique will allow the recurrent network to remove 20% of the weights, while, for the convolutional layers 40%. For the optimization, we use Adam with a learning rate of 0.0001.

For the sake of simplicity, they are listed in table .

5. EXPERIMENTAL ASSESSMENT AND RESULTS

5.1. *Corpus*

Although the automatic identification of the irony for the Italian language is a relevant task, it is a new research area that has barely been studied in the past. The lacking attention to the topic has led to the slow growth of the research field, which is characterized by few available resources for developing and testing classification systems. To accomplish the task, we exploit a dataset created for the automatic identification of irony for the Italian language presented in [5]. It is a collection of ironic texts (3,185) harvested from both the *Spinoza* blog and

Table 1. List of the model parameters.

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

the twitter account of *spinozait*. *Spinoza*[b]is a popular blog which main content is satire on politics that is widely approval as ironic resource. As counterpart, the corpus contains 22.295 non-ironic tweets retrieved from the account of seven popular daily newspapers: *Corriere della Sera*, *Gazzetta dello Sport*, *Il Messaggero*, *Repubblica*, *Il Resto del Carlino*, *Il Sole 24 ore*, and *La Stampa*. The choice of utilizing the newspaper tweets as negative examples is due to the similarity of topics tackled by both *Spinoza* and the newspapers, which are mainly *politics* and *news*. Furthermore, *Spinoza* uses a writing style close to the one used by the newspapers.

5.2. *Experiments*

The testing phase aims at measuring the performance of the network to associate input texts to the relative classes. It has taken into account two main problems: the unbalancing of the dataset and the low number of tweets available for training and testing systems. Although the number of positive examples (3,185) is much less than the negative ones (22.295), we have decided to exploit the whole dataset. To do so, we ran experiments repeatedly by coupling positive examples with subsets of negative examples. In details, let C be the amount of positive examples, we repeat experiments by pairing the first C negative example to the C positive example. We create a second subset of data by pairing the second C negative examples to the entire set of positive example. After i steps, we create the subset by taking negative elements from the position $i * C$ to $i * C + C - 1$ and pairing such data to the entire set of positive examples.

For tackling the low number of available texts, we have relied on the well-known cross-validation methodology named K-Fold, which has been proven capable of generalizing the knowledge acquired by the system when the data is insufficient. K-Fold method partitions the dataset in K subsets. Alternately, each sub-set is taken as test-set and the rest of the data is utilized to train the model. For each trained model, its performance are computed on the testset. Finally, the measures are averaged.

In this case, for each pair negative examples-positive examples, a 10-Fold is applied, and Precision, Recall, Accuracy, and F1-Score measures are computed. Partial results are averaged over the executed runs.

[b]www.spinoza.it

We have set the model parameters as described in Section 4.4, and the models performances are evaluated for a variable number of epochs: 1-50. The best achieved results are shown in table 2.

Table 2. Averaged systems results over runs.

Model Name	Epochs	Accuracy	Recall	Precision	F1-Score
BILSTM+CONV	37	0.932	0.940	0.926	0.933
S-LSTM	37	0.626	0.630	0.452	0.512
ATT-LSTM	48	0.931	0.937	0.926	0.931
BI-LSTM	40	0.928	0.933	0.923	0.929
BI-ATT-LSTM	38	0.932	0.936	0.929	0.932

5.3. *Discussion*

The results shown in table 2 highlight overall good performances achieved by the systems. Almost all the tested models reach comparable values of Accuracy and F1-Score. The BILSTM+CONV neural network has been introduced to tackle specifically the sarcasm detection, and it is the most complex system we have used since it incorporates all the DL architectures described in the paper, furthermore, it examines auxiliary features computed ad-hoc. Despite its complexity, experiments evidence that it achieves the same accuracy value of other systems and a slightly better value of F1-Score. Its best competitor is the BI-ATT-LSTM, which achieves the closest values of accuracy and F1-Score but relying on a *simpler* architecture. The latter does not exploit the CNN, and it uses only a dense layer. Except for the S-LSTM, which achieves the worst performances, other systems are capable of achieving results near to the highest ones by increasing the number of training epochs. While the BI-ATT-LSTM and the BILSTM+CONV need of *only* 37 epochs of training, the BI-LSTM and ATT-LSTM have to be trained respectively for 40 and 48 epochs.

Results show that considering in the tweets the number of exclamation marks, the number of question marks, the number of periods, the number of capital letters, the number of uses of *or* do not affect the performance of the network. Indeed, although such the auxiliary features are taken into account by the BILSTM+CONV model only, the compared architectures achieve the same results by exploiting only words and punctuation marks. The results are comparable not only by considering the F1-Score, but in all the computed measures. In other words, we can affirm that their performance are almost indistinguishable.

In the Italian context, we can conclude that the experimentation that leverages on Spinoza corpus and tweets taken from a bunch of newspapers, confirms that auxiliary features do not support the classification of ironic-sarcastic texts.

Under the computational effort perspective, the compared performances shown in Table 2 suggest to do choose as favourite model the ATT-LSTM one. Indeed, despite a greater number of epochs needed to achieved the best results, its architecture is less complex. Choosing a *lighter* architecture allows us to think applications that can be embedded in mobile devices avoiding to use a main workstation and classical rest-based services. Such applications might support users by bridging the developed irony-sarcasm detection DL system with other applications, such us the ones related to social medias. In this way, sarcams-irony might be recognized automatically by offering a real time support to who has troubles in comprehending such a linguistic devices.

In [5], a decision tree has been trained, and experiments have been conducted by exploiting the 10-Fold methodology. Although there are minors differences between the methodologies of system testing, the results are still comparable. The authors of [5], compute Recall, Precision, F1-Scores by exploiting a 10-Fold methodology as well. The comparison shows a +22% value of F1-Score achieved by our best model.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have analyzed and compared five deep learning methodologies for identifying irony and sarcasm autonomously. An accurate testing phase has been turned out to compare such systems. It shows that almost all the systems are capable of achieving high performances, and it evidences the contribution of DL layers for tackling this problem. Results suggest that the combination of Bidirectional LSTM, Attention mechanism and a fully connected layer is a good compromise between complexity and performances, while a simple model such as an S-LSTM is not enough to tackle the problem. Future works will consider applying these results for the text complexity evaluation, to contribute to the enrichment of the automatic text complexity evaluation system by improving state of art results [26][27]. Another line of research consists in taking into account *data approximation paradigms* (e.g., [11, 15, 14, 13, 12, 10, 9, 1]), which may turn useful in dealing with the complexity of emerging big data environments.

ACKNOWLEDGEMENTS

The authors would thank Prof. Giosué Lo Bosco for the fruitful discussions on the approach.

References

1. Ahn, S., Couture, S. V., Cuzzocrea, A., Dam, K., Grasso, G. M., Leung, C. K., McCormick, K. L., and Wodi, B. H. A fuzzy logic based machine learning tool for supporting big data business analytics in complex artificial intelligence environments. In 2019 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2019, New Orleans, LA, USA, June 23-26, 2019 (2019), IEEE, pp. 16.
2. Aloufi, S., and El Saddik, A. Sentiment identification in football-specific tweets. *IEEE Access* 6 (2018), 7860978621.
3. Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, 2014.
4. Bamman, D., and Smith, N. A. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media* (2015).
5. Barbieri, F., Ronzano, F., and Saggion, H. Italian irony detection in twitter: a first approach. In *The First Italian Conference on Computational Linguistics CLiC-it (2014)*, vol. 28.
6. Bouazizi, M., and Ohtsuki, T. Multi-class sentiment analysis in twitter: What if classification is not the answer. *IEEE Access* 6 (2018), 6448664502.
7. Butler, H. E., et al. *The Instituto Oratoria of Quintilian: With an English Translation*, vol. 1. W. Heinemann, 1953.
8. Campan, A., Cuzzocrea, A., and 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, December 11-14, 2017 (2017), IEEE Computer Society, pp. 44534457.

9. Ceci, M., Cuzzocrea, A., and Malerba, D. Effectively and efficiently supporting roll-up and drill-down OLAP operations over continuous dimensions via hierarchical clustering. *J. Intell. Inf. Syst.* 44, 3 (2015), 309333.
10. Cuzzocrea, A. Improving range-sum query evaluation on data cubes via polynomial approximation. *Data Knowl. Eng.* 56, 2 (2006), 85121.
11. Cuzzocrea, A., and Chakravarthy, S. Event-based lossy compression for effective and efficient OLAP over data streams. *Data Knowl. Eng.* 69, 7 (2010), 678708.
12. Cuzzocrea, A., Furfaro, F., and Sacca, D. ' Hand-olap: A system for delivering OLAP services on handheld devices. In *6th International Symposium on Autonomous Decentralized Systems (ISADS 2003)*, 9-11 April 2003, Pisa, Italy (2003); IEEE Computer Society, pp. 8087.
13. Cuzzocrea, A., and Matrangolo, U. Analytical synopses for approximate query answering in OLAP environments. In *Database and Expert Systems Applications, 15th International Conference*, DEXA 2004 Zaragoza, Spain, August 30-September 3, 2004, Proceedings (2004); F. Galindo, M. Takizawa, and R. Traunmuller, Eds., vol. 3180 of *Lecture Notes in Computer Science*, Springer, pp. 359370.
14. Cuzzocrea, A., Sacca, D., and Serafino, P. ' A hierarchy-driven compression technique for advanced OLAP visualization of multidimensional data cubes. In *Data Warehousing and Knowledge Discovery, 8th International Conference*, DaWaK 2006, Krakow, Poland, September 4-8, 2006, Proceedings (2006); A. M. Tjoa and J. Trujillo, Eds., vol. 4081 of *Lecture Notes in Computer Science*, Springer, pp. 106119.
15. Cuzzocrea, A., and Serafino, P. LCS-hist: taming massive high-dimensional data cube compression. In *EDBT 2009, 12th International Conference on Extending Database Technology*, Saint Petersburg, Russia, March 24-26, 2009, Proceedings (2009); M. L. Kersten, B. Novikov, J. Teubner, V. Polutin, and S. Manegold, Eds., vol. 360 of *ACM International Conference Proceeding Series*, ACM, pp. 768779.
16. Di Gangi, M., Lo Bosco, G., and Pilato, G. Effectiveness of data-driven induction of semantic spaces and traditional classifiers for sarcasm detection. *Natural Language Engineering* 25 (2019), 257285.
17. DAvanzo, E., Pilato, G., and Lytras, M. Using twitter sentiment and emotions analysis of google trends for decisions making. *Program* (2017).
18. Felbo, B., Mislove, A., Sgaard, A., Rahwan, I., and Lehmann, S. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (Copenhagen, Denmark, Sept. 2017)*, Association for Computational Linguistics, pp. 16151625.
19. Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.* (Conf. Publ. No. 470) (1999), vol. 2, pp. 850855 vol.2.
20. Ghosh, A., and Veale, T. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis* (2016), pp. 161169.
21. Giudice, V. Aspie96 at ironita (evalita 2018): Irony detection in italian tweets with character-level convolutional rnn. *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA18)* (2018), 160165.
22. Gonzalez-Ib anez, R., Muresan, S., and Wacholder, N. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (2011), pp. 581586.
23. Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning.* MIT Press, 2016.
24. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. Learning word vectors for 157 languages. *CoRR* abs/1802.06893 (2018).
25. Kreuz, R. J., and Glucksberg, S. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of experimental psychology: General* 118, 4 (1989), 374.
26. Lo Bosco, G., Pilato, G., and Schicchi, D. A neural network model for the evaluation of text

complexity in italian language: a representation point of view. *Procedia computer science* 145 (2018), 464470.

27. Lo Bosco, G., Pilato, G., and Schicchi, D. A sentence based system for measuring syntax complexity using a recurrent deep neural network. In *2nd Workshop on Natural Language for Artificial Intelligence, NL4AI 2018*, (2018), vol. 2244, CEUR-WS, pp. 95101.
28. Maynard, D. G., and Greenwood, M. A. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC 2014 Proceedings* (2014), ELRA.
29. McDonald, S. Exploring the process of inference generation in sarcasm: A review of normal and clinical studies. *Brain and language* 68, 3 (1999), 486506.
30. Oraby, S., Harrison, V., Reed, L., Hernandez, E., Riloff, E., and Walker, M. Creating and characterizing a diverse corpus of sarcasm in dialogue. arXiv preprint arXiv:1709.05404 (2017).
31. Pai, P.-F., and Liu, C.-H. Predicting vehicle sales by sentiment analysis of twitter data and stock market values. *IEEE Access* 6 (2018), 5765557662.
32. Reyes, A., and Rosso, P. On the difficulty of automatically detecting irony: beyond a simple case of negation. *Knowledge and Information Systems* 40, 3 (2014), 595614.
33. Schicchi, D., Pilato, G., and Lo Bosco, G. Attention-based model for evaluating the complexity of sentences in english language. In *20th ieee mediterranean eletrotechnical conference* (2020), p. in press.
34. Schicchi, D., Pilato, G., and Lo Bosco, G. Deep neural attention-based model for the evaluation of italian sentences complexity. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)* (2020), pp. 253256.
35. Son, L. H., Kumar, A., Sangwan, S. R., Arora, A., Nayyar, A., and Abdel-Basset, M. Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE Access* 7 (2019), 2331923328.
36. Terrana, D., Augello, A., and Pilato, G. Facebook users relationships analysis based on sentiment classification. In *2014 IEEE International Conference on Semantic Computing* (2014), IEEE, pp. 290296.
37. Tsur, O., Davidov, D., and Rappoport, A. Icwsma great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *fourth international AAAI conference on weblogs and social media* (2010).
38. Vassallo, G., Pilato, G., Augello, A., and Gaglio, S. Phase coherence in conceptual spaces for conversational agents. *Semantic computing* (2010), 357371.
39. Wallace, B. C., Kertz, L., Charniak, E., et al. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Volume 2: Short Papers) (2014), pp. 512516.
40. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (San Diego, California, June 2016)*, Association for Computational Linguistics.