# Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection

**Chapter** · May 2019

DOI: 10.1007/978-3-030-14070-0_69

**2 authors:**

Paul Mandal
California State University, Fullerton
**1** PUBLICATION **1** CITATION
SEE PROFILE

Rakeshkumar V Mahto
California State University, Fullerton
**15** PUBLICATIONS **28** CITATIONS
SEE PROFILE

# 69 Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection

Paul K. Mandal and Rakeshkumar Mahto

## 69.1 Introduction

Sarcasm detection has being a difficult problem in traditional Natural Language Processing (NLP)/Artificial Intelligence (AI). As described by Pozzi and colleagues [1], "*The difficulty in recognition of sarcasm causes misunderstanding in everyday communication and poses problems to many NLP systems*." Due to the importance and complexity of the problem, many automatic sarcasm detection techniques were reviewed in [2]. The approaches presented for sarcasm detection in [2] were rule-based AI, statistical based AI, and machine learning based AI. However, the rule-based AI in sarcasm detection shows an inability in understanding the context or meaning of words [1–2]. Additionally, the rule-based AI is quite onerous to program. The rule-based modeling for NLP presented in [3] was only able to understand active voice sentences. In this paper, we present a unique deep neural network-based sarcasm detection technique in the news headlines that weight in the advantages of convolution neural network (CNN) and long short-term memory layer (LSTM).

The rest of this paper is organized as follows. Section 69.2 describes the dataset used in this paper. Section 69.3 presents detail CNN-LSTM architecture we used for sarcasm detection. Section 69.4 highlights the results obtained after applying CNN-LSTM architecture. Finally, Sect. 69.5 concludes the paper.

P. K. Mandal
Department of Computer Engineering, California State University, Fullerton, CA, USA
e-mail: pmandal@csu.fullerton.edu

R. Mahto (✉)
Department of Computer Engineering, California State University Fullerton, Fullerton, CA, USA
e-mail: ramahto@fullerton.edu

## 69.2 Description of Dataset

The dataset consists of 26,709 news headlines. Of these headlines, 43.9% are satire, and 56.1% are real as shown in Table 69.1. Each record consists of three attributes. The first was a Boolean variable indicating whether the headline is sarcastic or not. The second was the news headline itself. The third was the URL of the article. Since the goal was to determine whether a news headline was sarcastic or not, we omitted the URL from our model (Figs. 69.1 and 69.2).

The Natural Language Tool Kit (NLTK) library has a variety of methods that greatly simplify data preprocessing. First, the tokenizer function was used to split each headline into a vector of words. Then, NLTK's PorterStemmer method was used to stem the words. Finally, a frequency list of the entire corpus was created. Later, a dictionary of the 10,000 most common words and assigned each word a number corresponding to its index in the dictionary. Figure 69.3, illustrates the word cloud for the sarcastic and non-sarcastic words. Each headline was then zero padded or truncated so that it was 20 words long in order to feed them into the Neural Network.

## 69.3 Proposed CNN-LSTM Architecture

The architecture that proposed in this paper consists of an embedding layer, a CNN, and a bidirectional LSTM on word-level vector encodings of news headlines. This network architecture leverages advantages of the LSTM proposed in [4] and the CNN described by [5].

S. Latifi (ed.), *16th International Conference on Information Technology-New Generations (ITNG 2019)*,
Advances in Intelligent Systems and Computing 800,
https://doi.org/10.1007/978-3-030-14070-0_69

### 69.3.1 Embedding

The first layer of our network architecture accepts the news headlines as a sequence of word indices corresponding to the dictionary outlined in Sect. 69.3. The embedding layer accepts the inputted headlines and encodes each word into a vector of size *e*. In the proposed network, sequences are 20 words long, and the embedding size is 128. Thus, this layer will output a matrix of size $20 \times 128$.

### 69.3.2 Convolution

The output of the embedding layer is then fed into a 1-dimensional (1-D) convolution layer. For the 1-D convolutional layer, 32 filters with a kernel size of 7 were used. This layer will perform 1-D convolution on words rather than the 2-D convolutional windows commonly applied on image data which would cut our embedding vectors (and thus our words) into pieces.

The CNN layer allows viewing word combinations equivalent to the kernel size. This permits the neural network to have a sense of context when words are used with other words. In the proposed case, the kernel size is 7. Therefore the filter will establish 7-word combinations. Each neuron uses a rectified linear unit (ReLU) in order to learn non-linear features. It is worth noting that if a linear activation function is used, then the subsequent convolutional layers would be redundant.

### 69.3.3 Max Pooling

The output from the proposed convolutional layer then undergoes 1-dimensional max pooling. This layer converts each kernel size of the input into a single output by selecting the maximum value observed in each kernel. There are other variants of pooling such as min pooling or average pooling, but most often the most substantial value in the kernel tells the most about the data. Pooling is used to reduce overfitting; this allows to add more layers to proposed architecture and in turn allows the neural network to extract higher-level features.

### 69.3.4 Convolution

The output from the max pooling layer is now fed into another 1-dimensional convolutional layer. For this layer, 32 filters are again used and chose a kernel size of 7. To feed in the data, the output of the max pooling layer is padded in order to apply the filer of size 32 (note that zero padding does not affect the data in any other way).

It is more difficult to describe what associations this second convolutional layer will build. However, one can intuitively conceptualize that if the outputs of the first convolutional layer are groupings of words, then this second convolutional layer will output groupings of phrases. The activation function for this layer was also a ReLU.

### 69.3.5 Bidirectional LSTM

Technically, this layer can be viewed as two layers. First, the output of our convolution is duplicated. One of these sets is reversed. We now have one set in “chronological” order and another in “reverse” order. One LSTM is trained on the chronologically ordered set and another on the reversed set. The output of these two LSTM’s is then merged together.

It is worth noting that if an RNN is trained on a chronological set of language data, it can also extract a set of useful features from the reverse order. The idea of a Bidirectional LSTM is to learn features from both the chronological and reverse orderings of language data. Although the data has already passed through two CNN’s and isn’t “chronological” in the conventional sense, the LSTM can still take advantage of both orderings of the output. A recurrent dropout of 0.5 is used.

**Table 69.1** Type of datasets used

| Statistics/dataset | Headlines |
|---|---|
| Total number of records | 26,709 |
| Number of sarcastic records | 11,725 |
| Number of non-sarcastic records | 14,984 |

### 69.3.6 Output, Loss Function, and Hyperparameters

Since the news headlines used in this work, ultimately fall under the categories of sarcastic or not sarcastic, our output

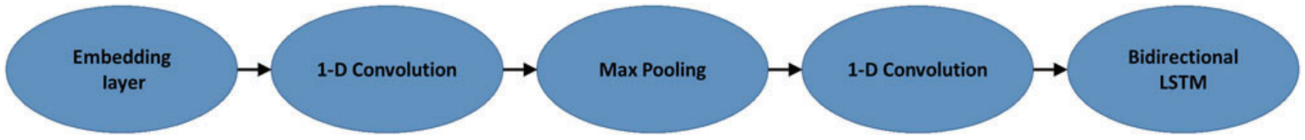


**Fig. 69.1** CNN-LSTM based architecture

layer is a single sigmoid neuron trained with loss function binary cross-entropy. To train the neural network, a sarcastic headline is represented as a 1 and a real headline as a 0. The output of the sigmoid corresponds to its confidence in deciding whether the headline is sarcastic or not.

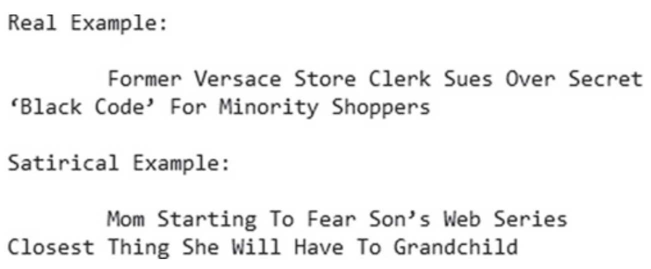
The neural network is trained using the optimizer Adam. RMSprop and stochastic gradient descent also used but did not perform as well. The optimal batch size was 128.

## 69.4 Results

This deep CNN-LSTM with word embedding was able to achieve an accuracy of 86.16% as shown in Fig. 69.4. Many models were tested before developing this "optimal" model. The primary challenge was to design a larger model that would not overfit. More data would have allowed us to design more complex models.

### 69.4.1 Comparison of Other Models

Currently, no other works have trained neural networks on this dataset. As a baseline, a feedforward neural network with 32 ReLU neurons in the first hidden layer, 4 ReLU neurons in the second neurons in the second hidden layer, and a single sigmoid neuron in the output layer was trained.



```
Real Example:

        Former Versace Store Clerk Sues Over Secret
'Black Code' For Minority Shoppers

Satirical Example:

        Mom Starting To Fear Son's Web Series
Closest Thing She Will Have To Grandchild
```

**Fig. 69.2** Example of real and satirical text analyzed

This yielded an accuracy of 84.56%. Later, another feedforward neural network is trained with three hidden units with ReLU neurons. The first two hidden layers consisted of 32 neurons with a dropout of 0.5; The third hidden layer was composed of four neurons. The output neuron was a single sigmoid neuron. This network architecture had an accuracy of 84.92%.

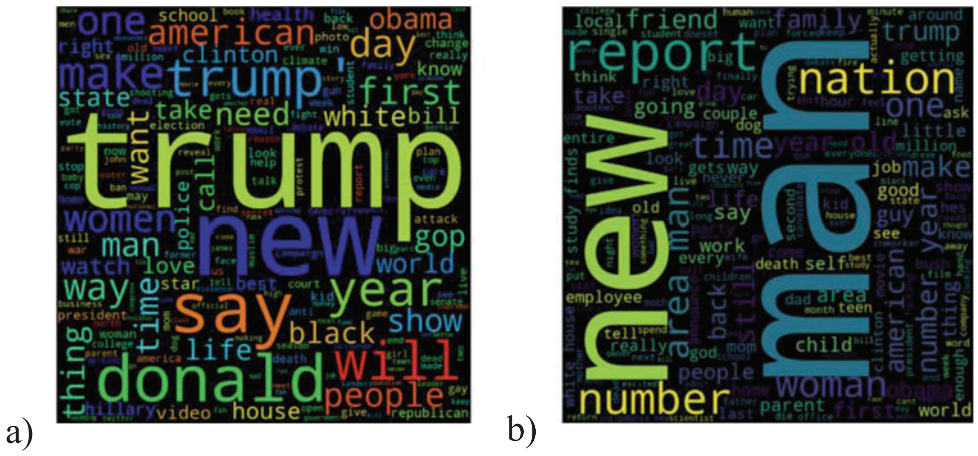
Various LSTM architectures without an embedding layer were tested. None of them achieved an accuracy above 75%. It should be noted that models with recurrent dropout performed much better than those without it. A network architecture with an embedding layer of 32 features and an LSTM with 32 units had an 85.26% accuracy. A similar architecture with an embedding layer of 256 achieved an accuracy of 85.52%.

Using 1-dimensional convolution yielded further gains. Architecture with an embedding layer of 128 features, a bidirectional LSTM with 32 neurons, and a convolutional layer with 32 filters and a kernel size of 7 with global max pooling achieved an accuracy of 86.04%. The next architecture that performed better is the one proposed in this paper.

### 69.4.2 Layer Performance

A few things were consistent regardless of the network architecture that we designed. The first is that weight regularization (whether it be the L1 norm, L2 norm, or any other variant) did not help for any of our network architectures. Secondly, all of our models served to benefit from dropout if they were big enough. For LSTM and CNN layers, embedding proved to be invaluable in boosting accuracy. Furthermore, using convolution before our bidirectional LSTM yielded further improvements.

**Fig. 69.3** Word cloud for (**a**) non-sarcastic text (**b**) sarcastic text



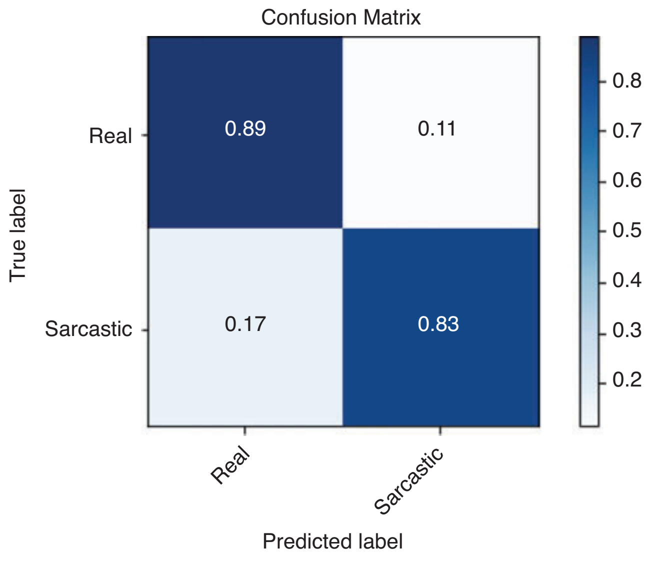


**Fig. 69.4** Confusion matrix to determine the accuracy of our model

## 69.5 Conclusion

In this paper, we presented a CNN-LSTM based sarcasm detection from the news headlines with 86.16% accuracy.

A severe limitation in our experiments was the lack of data at hand. More data would not only yield a higher accuracy on the current model, but also lead to the development of more advanced architectures. Furthermore, we did not experiment with pretrained word embeddings. Significant practical gains could be made with word2vec or other pretrained models.

## References

1. Pozzi, F.A., Fersini, E., Messina, E., Liu, B.: Sentiment Analysis in Social Networks. Morgan Kaufmann, Burlington, MA (2016)
2. Joshi, A., Bhattacharyya, P., Carman, M.J.: Automatic sarcasm detection: a survey. ACM Comput. Surv. **50**(5), 73
3. Bajwa, I., Choudhary, M.: A rule based system for speech language context understanding. J. Donghua Univ. **23**(6), 39–42 (2006)
4. Rahman, L., Mohammed, N., Azad, A.K.A.. A new LSTM model by introducing biological cell state. In: 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), pp. 1–6. (2016)
5. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751. Doha, Qatar (2014)