Recognition of Ironic Sentences in Twitter using Attention-Based LSTM

Andrianarisoa Tojo Martini, Makhmudov Farrukh, Hongwei Ge

Department of Computer Science & Technology
Dalian University of Technology
Dalian, P. R. China

***Abstract*—Analyzing written language is an interesting topic that has been studied by many disciplines. Recently, due to the explosive growth of Internet, social media has become an attractive source of searching and getting information for research purposes on written communication. It is true that different words in a sentence serve different purposes of conveying the meaning while they are of different significance. Therefore, this paper is going to employ the attention mechanism to find out the relative contribution or significance of every word in the sentence. In this work, we address the problem of detecting whether a tweet is ironic or not by using Attention-Based Long Short-Term Memory Network. The results show that the proposed method achieves competitive performance on average recall and F1 score compared to the state-of-the-art results.**

***Keywords*—Irony detection; attention; attention mechanism; sentiment analysis; long-short-term memory**

I. Introduction

Nowadays, the Web has become an indispensable source of searching and gaining information because of the quantity and diversity of textual content containing opinions expressed by internet users. Blogs, comments, forums, social networks, reactions or opinions are more and more centralized by search engines. The prodigious measure of data streaming from online social networking and micro-blogging platforms like Twitter, is increasingly attracting the many researchers in the area of sentiment analysis. From these social medias, the automatic detection of irony is, therefore, important for the development of sentiment analysis research, but at the same time it is also an interesting challenge from a cognitive point of view and can help to shed some lights on how human beings use irony as a communicative tool.

Sarcasm and irony are very similar. Generally speaking, irony is employed to convey the opposite meaning of the actual things you say, but its purpose isn't to harm the other person unlike sarcasm which is employed to hurt the other person. According to the Gricean tradition [1], the function of irony is to effectively communicate the opposite of the interpretation of the utterance. However, determining whether a text is ironic or not is a difficult task since the differences between ironic and non-ironic texts are usually extremely delicate. For example, one tweet wrote that "Love this weather #not" is ironic, but a similar tweet which wrote "Hate this weather #not happy" is considered as non-ironic.

In this paper, we introduce the deep learning representation in ironic tweets detection tasks by merging the attention mechanism with the LSTM layers and compare it with the state-of-the-art feature engineering approaches, as we know that state-of-the-art irony and sarcasm detection systems often only rely on deep and sequential neural networks [2] [3].

The Section 2 of this paper is a survey of the related work while Section 3 presents the proposed work by explaining the architecture and the methods used. In Sections 4 and 5 the experiment setup and the results are being respectively discussed. Finally, Section 6 presets the conclusion part.

II. Related Work

Identifying the ironic texts can help to understand the social web better and there are many related applications like sentiment analysis. Irony detecting techniques are important to enhance the performance of sentiment analysis. In [4], authors used the LIBSVM to perform the inductive learning for the training dataset perhaps in accordance with the recent work which has explored the use of Support Vector Machines for text classification with more precise results compared to the other classification techniques.

In [5], authors use Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Attentive RNN in irony detection tasks, and compare the results with the state-of-the-art feature engineering approaches. The first one is Convolutional Neural Network (CNN), which is introduced by [6], and used as a sentence modeling technique in Natural Language Processing (NLP) [7] by using word embedding. Their CNN is applied with one-directional convolutions over the embedded word vectors with multiple filters in various sizes. After applying one-max-pooling over all the outputs filters, the scalars are concatenated together as the encoded vector. The second model is Recurrent Neural Network (RNN), which has been created for the use of sequential data. The Neural Network generates an output vector which considers not only the current input, but also the previous result. The last output vector is taken as the encoded vector.

In [8], the authors made some improvements on previous work [9] by adding some features as well as the word graph similarity score. Each tweet is represented as directed unweighted word graph and the edge between each word is created based on the vicinity window size. Each class in the dataset is represented as directed unweighted graphs. Then a vector is produced after comparing each class graph. And this vector is used as features by machine learning algorithm. The graph is constructed based on a class assignment and then they measure the similarity of a tweet with each class graph.

Some works have also been carried out for detecting satire in English text, for example [10]. Firstly, authors introduce approach to binary classification of satire in English text. Secondly, they propose a list of generalized linguistic features which provide good results on different types of satire corpora. Furthermore, they make available a standard satire corpus which was retrieved from twitter (with user generated tags such as #satire, #satirical). But developed system might not perform very well on time-based satirical posts on social media platforms.

III. Proposed Approach

A. Self-Attention Mechanism

First of all, since the research is concentrated on the attention mechanism, we have to discuss about the Self-Attention Mechanism. Recurrent Neural Networks (RNNs) output their hidden state $\mathrm{h_i}$ as they process a sequence and that hidden state holds a summary of the information in the sequence. We used a self-attention mechanism [11] to amplify the contribution of important words in the final representation.

After using the attention mechanism, we compute r as combination of all h_i (Fig. 1). The weights a_i have been learned by the network and the magnitude of those weights learned signifies the importance of each hidden state in the final representation.

The hidden state at the last time-step is used as the representation of the input. In long sequences case, the Recurrent Neural Network might not be able to hold all the important information in its final hidden state. In order to amplify the contribution of important elements in the final representation, an attention mechanism has been used.

$$r = \sum_{i=1}^{N=1} a_i h_i \qquad (1)$$

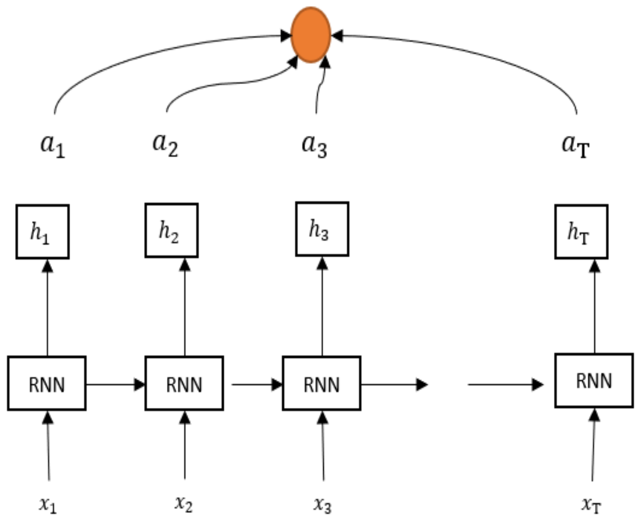


Fig. 1. Attention RNN.

B. Preprocessing

We've used a text processing tool called Ekphrasis presented by [12], which can perform tokenization, word normalization, word segmentation (for splitting hashtags) and spelling correction, using word statistics from two big corpora namely English Wikipedia and Twitter.

1) Tokenization: Tokenization is the initial preprocessing stage which makes it the foundation for the latter stages. Therefore, it will certainly make an effect of the feature's quality studied by the network. Tokenization in Twitter is full of challenges for that various usage of vocabulary and expressions are here and there. Of course, some of the challenges came from the dilemma of projecting the whole expression or simply taking its tokens. To rise to this challenge, Ekphrasis recognized the markup, emoticons, emojis, dates, acronyms, censored words and words with emphasis.

2) Normalization: Apart from the method of tokenization, we also make some adjustment on certain selected tokens, such as spelling correction, words normalization and sedimentation. Furthermore, we also figure out what kinds of tokens should be omitted, normalized and surrounded together with those that should be replaced with special tags such as URLs, emails and @user.

C. Attention-based LSTM Model Description

The framework of our attention-based LSTM network is illustrated in (Fig. 2). Next, we will introduce each layer in our model from bottom to top in detail.

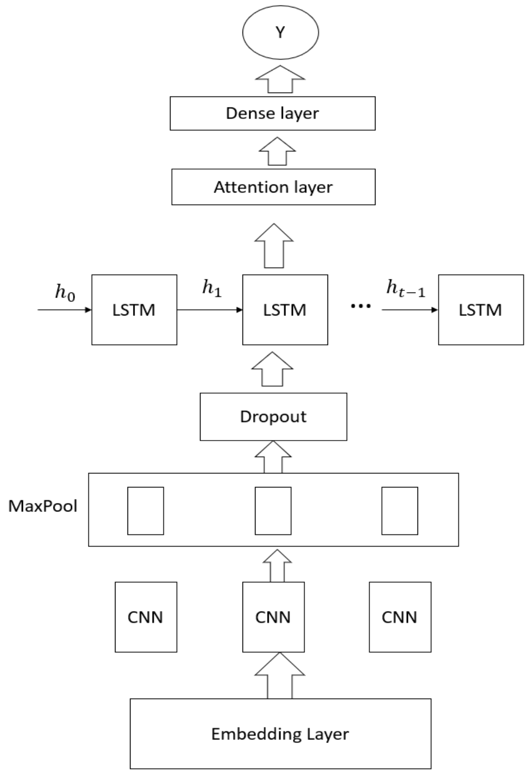


Fig. 2. Architecture of LSTM with Attention Mechanism.

3) Embedding Layer: This process happens just right after the pre-processing. Word embedding techniques aim to use continuous low-dimension vectors representing the features of the words [13], which tweets are transformed into a sequence of words $S = (s_1, \ldots, s_N,), S \in \mathbb{R}^{N \times d}$, where N is the number of a tweet, and d denotes the dimension of a word vector [14]. We use Word2Vec [13] as the vector representation of the words in tweets.

4) Convolutional and Max-Pooling Layers: After getting the pre-trained word vectors "word2vec" from the word embedding Layers, we train a convolutional neural network, followed by a max-pooling layer. The goal of convolution is to extract the input feature, and pooling is to subsample the output of the convolution matrix. The regular way to do pooling is by applying a max operation to the result of each filter. There are two reasons to use a max-pooling layer in our research. First, by doing elimination of any non-maximal values, it reduces computation for upper layers. Second, the max-pooling layer can extract the local dependency within different regions to keep the most salient information.

5) LSTM Layer: The next layer in our model is LSTM layer. LSTM is kind of RNN which has been introduced firstly by [15]. For LSTM, Cell state (c_t) are connected to three gates which are forget gate (f_t), input gate (i_t) and output gate (o_t) respectively. Fig. 3 illustrates the architecture of a standard LSTM.

More formally, each cell in LSTM can be computed as follows:

$$X = \begin{bmatrix} h_{t-1} \\ x^t \end{bmatrix} \quad (2)$$

$$f_t = \theta\,(W_f \cdot X + b_f) \quad (3)$$

$$i_t = \theta\,(W_i \cdot X + b_i) \quad (4)$$

$$\tilde{C}_t = \tanh(W_c \cdot X + b_c) \quad (5)$$

$$C_t = f_t * C^{t-1} + i_t * \tilde{C}_t \quad (6)$$

$$o_t = \theta(W_o X + b_o) \quad (7)$$

$$h_t = o_t * \tanh(C_t) \quad (8)$$

Where $W_f, W_i, W_o, W_c \in \mathbb{R}^{d \times 2d}$ are the weighted matrices and $b_f, b_i, b_c, b_o \in \mathbb{R}^d$ are biases of LSTM to be learned during training, parameterizing the transformations of the input, forget and output gates respectively. θ is the sigmoid function and $*$ stands for element-wise multiplication, x^t includes the inputs of LSTM cell unit.

This layer is used to capture long-range contextual information from tweets. At time step i, a hidden state h_i is generated which contains both previous and future context information. Since different words and phrases serve different purposes to irony detection, we propose to design an attention layer after the LSTM layer to help our model focus on important words and contexts.

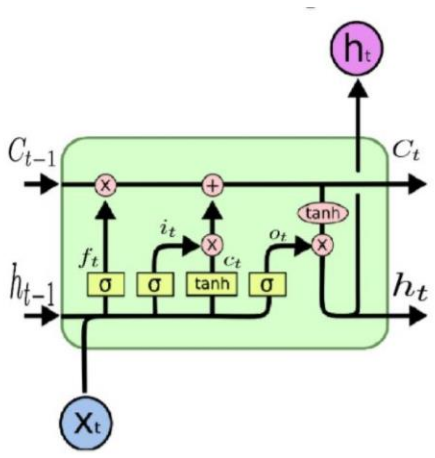


Fig. 3. Architecture of Cell in LSTM.

6) Attention Layer: The input of the attention layer is the hidden state vector h_i at each time step. The attention weight m_i for this time step can be computed as:

$$m_i = tanh(h_i) \quad (9)$$

$$\hat{\alpha}_i = w^T m_i + b \quad (10)$$

$$\alpha_i = \frac{exp(\hat{\alpha}_i)}{\Sigma_j exp(\hat{\alpha}_i)} \quad (11)$$

Where w and b are the parameters of the attention layer. The output of attention layer at the i_{th} time step is formulated as follows:

$$r_i = \alpha h_i \quad (12)$$

IV. Experimental Setup

First of all, let's talk about the datasets. The dataset used consists of 355k English tweets (43k ironic and 312k in literal sentiment sense, we named it dataset1. Another dataset collected by Ghosh [2] contains 18k sarcastic tweets (which can be used on irony) and 21k regular tweets. In order to collect the most data for dasatest1, we used the Twitter API (https:// dev.twitter.com/) to stream tweets from Twitter by using hashtags #irony, #sarcasm and #not as key word. And the data was cleaned by using the preprocessing method from the section 3 (which means that ironic hashtags, such as #not, #sarcasm, #irony, in the dataset have been removed), it was labeled 1 for ironic texts and 0 for normal.

As for the implementation, our model is implemented in Keras library. We conducted the experiment with different values for the LSTM hidden state size and for the dropout probability, obtaining best results for a dropout probability of 0.5 and 128 units for the hidden vector. The table below (Table I) shows the repartition of the collected dataset, we trained 80% of the provided data as training set and 20% as test set. Since the data is kind of voluminous, we only use the number of epochs as 3. Cross entropy and Adam are used as the loss function and optimization algorithm of the output layer.

TABLE I. COUNTS AND PERCENTAGES OF IRONIC AND NON-IRONIC OF THE TWEETS COLLECTED AND TEST-TRAIN SET

	Non-Ironic	**Ironic**	**Total**
Training set	249800 (88%)	34382 (12%)	284182
Test set	62501 (88%)	8545 (12%)	71046
Collected data	312193 (88%)	43035 (12%)	355228

V. FINAL RESULT AND DISCUSSION

A. *Results*

Tables II and III show the results of the experiments after using both LSTM approach and Attention Based approach, and compare them to the state models presented by [2] . We only report the average Precision (Avg.Prec), Recall (Avg.Rec), and F1 scores (Avg.F1).

Table II below presents a comparison of the results trained on the collected dataset (dataset1), we observe that our model with Attention based LSTM almost outperforms every model than other models, except the model which is a combination of CNN, LSTM, and DNN introduced by [2], it outperforms our model at the precision by 0.4% margin but they both got the same results on the F1 score. As for the proposed model with just LSTM, it performs the lowest performance in every evaluation.

As for Table III, we show that the performance of our system can outperform some of the baseline methods on the Ghosh dataset [2] but got outperformed by the CNN, LSTM, and DNN model.

TABLE II. COMPARISON OF OUR METHOD TO BASELINE USING DATASET1

Model		**Avg. Prec**	**Avg. Rec**	**Avg. F1**
Our model	Attention based LSTM	0.836	0.883	0.859
	LSTM	0.703	0.805	0.751
Ghosh	CNN + LSTM + DNN (with dropout)	0.84	0.876	0.857
	LSTM+ LSTM	0.734	0.842	0.784
	CNN+CNN	0.716	0.804	0.758

TABLE III. COMPARISON OF OUR METHOD TO BASELINE USING GHOSH DATASET

Model		**Avg. Prec**	**Avg. Rec**	**Avg. F1**
Ghosh	CNN + LSTM + DNN (with dropout)	0.899	0.91	0.904
	LSTM+ LSTM	0.854	0.871	0.862
	CNN+CNN	0.856	0.879	0.868
Our model	LSTM	0.777	0.859	0.816
	Attention based LSTM	0.875	0.894	0.884

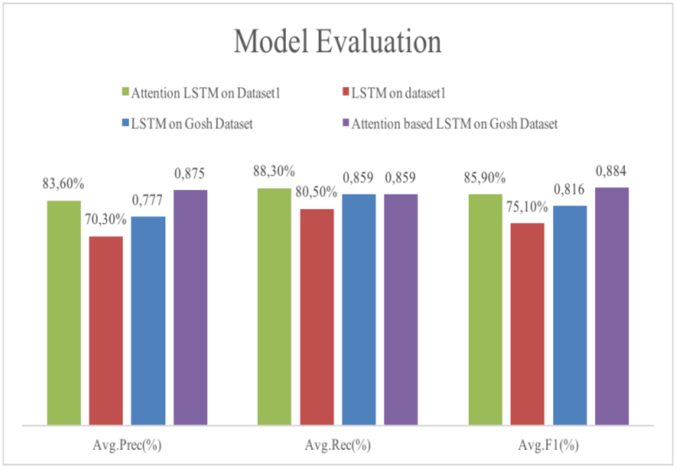


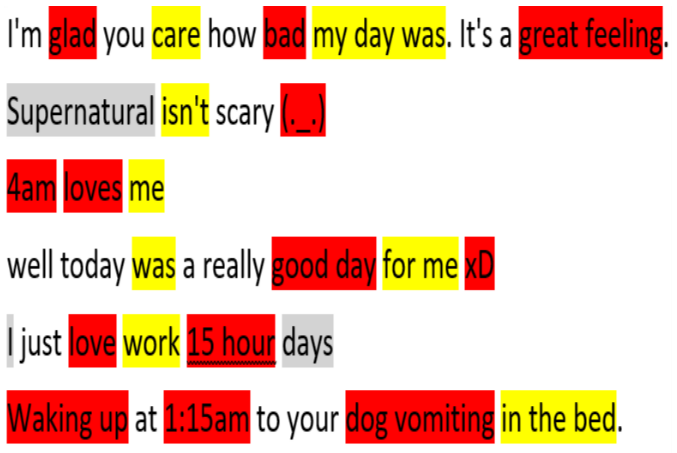
Fig. 4. Attention Architecture with LSTM with Attention Mechanism.

Fig. 4 shows that when using the Attention Mechanism on the LSTM layer, the model performs better than the one that doesn't use it. The Attention based Model makes an improvement on the Precision by more than 9%, around 3 to 8% on Recall and more than 8% on F1 score.

B. *Discussion*

1) *Attention visualization:* In the following figure (Fig. 5), we are going to get a closer look at the degree showing how much attention mechanism will better the performance of irony detection.

According to the given figure, there are some certain usage of language such as apparent emotional words, old topics, emojis, punctuation, numerals and sometimes slang and ungrammatical expressions attaining much more focus in the internet which makes it the biggest factor in case of the contribution to irony detection. The network is going to study the significance of certain words, it targets at finding out what factors will make a difference when it comes to the final ironical decision. As shown in the figure, the reddish color is used to highlight attention weights and the color gradients are there to make a distinction between the heavy weights of attention and the light one.


I'm glad you care how bad my day was. It's a great feeling.

Supernatural isn't scary (._.)

4am loves me

well today was a really good day for me xD

I just love work 15 hour days

Waking up at 1:15am to your dog vomiting in the bed.

Fig. 5. Attention Visualization.

VI. Conclusion

In this paper, we proposed a Long Short-Term Memory (LSTM) with attention mechanism model to detect English ironic sentences from Twitter. The proposed model got competitive result compared to the state-of-the-art models without using further feature engineering. The results showed that our model performs better on the collected dataset, especially on the recall and f1 score. On the Ghosh [2] dataset, our Attention-Based model outperformed the CNN and LSTM model proposed by [2] but couldn’t outperform the model with a combination of CNN, LSTM, and DNN. Finally, in the discussion part we show that the attention vectors generated by our attention layer can capture specific words which are very useful to decide for the training, it can decide whether the tweet selected is ironic or not. In a future work, we would like to explore how to make full usages of the attention mechanism on text sentiment analysis.

Acknowledgment

This work was supported by the National Natural Science Foundation of China No. 61471084, No. U1608253, and the open program of State Key Laboratory of Software Architecture No. SKLSA2016B-02.

References

[1] S. Chapman, "Logic and Conversation," in Paul Grice, Philosopher and Linguist, London, Palgrave Macmillan UK, 2005, pp. 85-113.

[2] A. Ghosh and D. T. Veale, "Fracking Sarcasm using Neural Network," in Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis , San, 2016.

[3] M. Zhang, Y. Zhang and G. Fu, "Tweet Sarcasm Detection Using Deep Neural Network," in COLING, 2016.

[4] T. Ahmad, H. Akhtar, A. Chopra and M. W. Akhtar, "Satire Detection from Web Documents Using Machine Learning Methods," 2014 International Conference on Soft Computing and Machine Intelligence, pp. 102-105, 2014.

[5] Y.-H. Huang, H.-H. Huang and H.-H. Chen, "Irony Detection with Attentive Recurrent Neural Networks," in ECIR, 2017.

[6] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, pp. 2278-2324, 11 1998.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014.

[8] U. Ahmed, L. Zafar, F. Qayyum and M. Arshad Islam, "Irony Detector at SemEval-2018 Task 3: Irony Detection in English Tweets using Word Graph," in Proceedings of The 12th International Workshop on Semantic Evaluation, New, 2018.

[9] G. Giannakopoulos, V. Karkaletsis, G. Vouros and P. Stamatopoulos, "Summarization System Evaluation Revisited: N-gram Graphs," ACM Trans. Speech Lang. Process., vol. 5, pp. 5:1--5:39, 10 2008.

[10] A. N. Reganti, T. Maheshwari, U. Kumar, A. Das and R. Bajpai, "Modeling Satire in English Text for Automatic Detection," in 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 2016.

[11] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," CoRR, vol. abs/1409.0473, 2014.

[12] C. Baziotis, N. Pelekis and C. Doulkeridis, "DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis," in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) , Vancouver, 2017.

[13] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," CoRR, vol. abs/1301.3781, 2013.

[14] Y. Zhang, J. Wang and X. Zhang, "YNU-HPCC at SemEval-2018 Task 1: BiLSTM with Attention based Sentiment Analysis for Affect in Tweets," in Proceedings of The 12th International Workshop on Semantic Evaluation, New, 2018.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, pp. 1735-1780, 1997.