Irony Detection with Attentive Recurrent Neural Networks

Yu-Hsiang Huang, Hen-Hsen Huang, and Hsin-Hsi Chen(✉)

Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
{yhhuang,hhhuang}@nlg.csie.ntu.edu.tw, hhchen@ntu.edu.tw

Abstract. Automatic Irony Detection refers to making computer understand the real intentions of human behind the ironic language. Much work has been done using classic machine learning techniques applied on various features. In contrast to sophisticated feature engineering, this paper investigates how the deep learning can be applied to the intended task with the help of word embedding. Three different deep learning models, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Attentive RNN, are explored. It shows that the Attentive RNN achieves the state-of-the-art on Twitter datasets. Furthermore, with a closer look at the attention vectors generated by Attentive RNN, an insight into how the attention mechanism helps find out the linguistic clues of ironic utterances is provided.

Keywords: Irony detection · Neural networks · Sentiment analysis

1 Introduction

Authors/speakers often use ironic expressions to convey their strong feelings in some situations. An irony says something other than what it meant, or says the opposite of what it meant. For example, the utterance "*I love to be ignored*" means "*I hate to be ignored*" in general understanding. Automatic irony detection aims at realizing people's real intentions. It has many potential applications. In opinion mining and sentiment analysis, the polarities of opinionated expressions in reviews affect readers' decision-making on specific targets. Ironic expressions bring in much stronger comments and thus should have more effects.

Sarcasm and irony are very similar in surface form, but sarcasm ridicules on some victims [6]. In this paper, we focus on the phenomenon of using opposite literal meaning as a mean to strengthen one's point. In other words, we do not distinguish their strict differences. Irony detection is challenging, because, to understand the actual meaning, readers/listeners also need to consider context and background knowledge rather than just interpreting the expressions literally.

In this paper, we will explore Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Attentive RNN in irony detection tasks, and compare them with the state-of-the-art feature engineering approaches. This paper is organized as follows. Section 2 surveys the related work. Section 3 presents how CNN, RNN and Attentive RNN model a sentence and classify it into one of the classes, i.e., Irony or Non-Irony. Section 4 introduces experimental datasets, shows and discusses the results. Section 5 further explains why Attentive RNN achieves the best performance with a case study.

2 Related Work

Reyes et al. [9] first collected an ironic tweet corpus by searching with hashtags to avoid labeling manually. Machine learning techniques with textual features were explored on the irony detection task [2,9]. The hashtag-based approaches are not always suitable for irony corpus construction for all languages. Tang and Chen [10] proposed a method to construct a Chinese irony corpus based on the use of emoticons, linguistic forms, and sentiment polarity. Recently, word embedding [8] is widely adopted in various NLP tasks for its power on semantic similarity between words. Ghosh et al. [3] proposed a maximum-valued matrix-element SVM kernel using word embedding to deal with word sense disambiguation task on an irony corpus.

3 Irony Detection Model

We regard the irony detection task as a binary classification problem. Words in a sentence are represented as a sequence of embedded word vectors with a table lookup in pre-trained vectors. We first encode each sentence into a vector with three different neural network models.

The first one is Convolutional Neural Network (CNN), which is introduced by LeCun et al. [7], and used as a sentence modeling method in NLP [5] with the use of word embedding [8]. Our CNN is applied with one-directional convolutions over the embedded word vectors with multiple filters in various sizes. After one-max-pooling applied over all the filters' outputs, the resulted scalars are concatenated together as the encoded vector.

The second model is Recurrent Neural Network (RNN), which is invented for the use of sequential data. In each instant of time, RNN generates an output vector which considers not only current input, but also the previous processed result (memory). The last output vector is taken as the encoded vector.

Attention mechanism is first used in NLP by Bahdanau et al. [1], and gets popular recently. Our third model is Attentive RNN, which makes a weighted combination of all the output vectors generated from the underlying RNN. In this way, our model takes a global view on all the past information. Given each output vector h_t resulting from time t and let Y be a matrix consisting of output vectors $[h_1;...;h_L]$ by concatenating them together, when the input sentence contains L words, we can get an attended vector h' via the following formula:

$$\alpha = softmax(w^T Y)$$
$$h' = Y\alpha^T$$

where w is a trained vector, and α is the attention weight vector. We get the encoded vector h' by summing up each output vector h_t weighted by the value in dimension t in α.

At the last, the sentence encoded vectors generated by the three neural networks are individually passed to a full-connected layer, along with a softmax layer, to project into the target space of the two classes, i.e., Irony and Non-Irony. The models are trained with the cross-entropy loss as objective function in an end-to-end way to make every part of the model optimized for this task.

4 Experiments

4.1 Datasets and Experiment Setup

The Twitter dataset collected by Ghosh et al. [3] is used in this paper. It includes 198,041 tweets in sarcastic (ironic) sense and 197,917 tweets in literal sentiment sense. Ghosh et al. first collected 37 target words (e.g. "genius") and search tweets containing these target words, i.e., each tweet belongs to one of the target words. Therefore this dataset can be viewed as 37 subdatasets. We take the best model of Ghosh et al., MVME kernel SVM with skip-gram word2vec, as our baseline. In the preprocessing step, the hashtags and username mentions are removed, and hyperlinks and out-of-vocabulary words are treated as special tokens. The skip-gram word2vec vectors pre-trained by Google are used[1], where the embedding would be fine tuned in the training process, and words not shown in the pre-trained vectors are initiated randomly. The CNN model is applied with filter sizes 2, 4, and 6, and each size has 500 filters. The RNN and Attentive RNN models are applied with the Long Short-Term Memory (LSTM) units [4]. All the hidden layers have a dimension size 400 in the three models. Dropout rate [12] is set to 0.2 in the last full-connected layer. RMSProp [11] is used for optimization with learning rate 0.001, rho 0.9, and epsilon 10^{-8}.

4.2 Results

Experimental results are shown in Table 1, including the micro average of Precision, Recall and F1 score among the target words. The maximum and the minimum F1 scores are also provided. We can find that CNN gets a really high average precision (91.5%) over all target words. RNN performs worse than CNN. When attention mechanism is introduced, Attentive RNN makes a progress in F1 score by 5.0% compared to the baseline and beats the other two models. It is interesting to see that the target word with minimum F1 score is "genius" in both models of Ghosh et al. and our Attentive RNN. This is because the target "genius" has less training instances in the dataset. However, the Attentive RNN still gets a better F1 score in this case.

[1] https://code.google.com/archive/p/word2vec/.

Table 1. Performance on Ghosh et al. dataset.

Model	Avg. P(%)	Avg. R(%)	Avg. F1(%)	Max. F1 (Target word)	Min. F1 (Target word)
Ghosh et al.	81.9 ± 3.8	88.1 ± 3.2	84.8 ± 3.0	88.8 (love)	74.2 (genius)
CNN	**91.5** ± 4.1	86.2 ± 5.2	88.6 ± 3.3	93.9 (sweet)	81.6 (interested)
RNN	86.7 ± 4.2	89.9 ± 5.4	88.1 ± 3.5	93.9 (joy)	80.0 (shocked)
Attentive RNN	88.8 ± 4.3	**90.9** ± 2.9	**89.8** ± 2.7	**95.5** (beautiful)	**83.6** (genius)

We also conduct the experiment on Reyes et al. dataset [9]. The Attentive RNN increases the F1 scores to 92.3%, 89.5%, and 89.0% on Irony-{Education, Humour, Politic}, three subdatasets, respectively, which makes an improvement over the performance achieved by Barbieri and Saggion [2].

5 Discussion

In this section, we aim to get a clear insight into how attention mechanism improves irony detection performance.

5.1 Attention Weight Plots

We are interested in which word in a sentence the Attentive RNN will pay more attention to. Figure 1 presents the visualization of the attention weight vectors α generated under the classification process. Word having a higher attention weight is shown with a darker color, indicating that this model pays more attention to that word. We can observe that this model can easily capture the relatively negative terms, for instance, “ignored”, “hurt”, “bad”, and “dumb” in 1i, 1g, 1f, and 1b. Besides, it also captures some swear words such as “fuck”, “suck”, and “shit” in 1a, 1c and 1e. Our model relies on the negative words in a potentially positive sentiment tweet (e.g., sentences with “glad”, “like”, “love”, and “yeah”) to determine whether it is ironic or not. It is interesting that the Attentive RNN

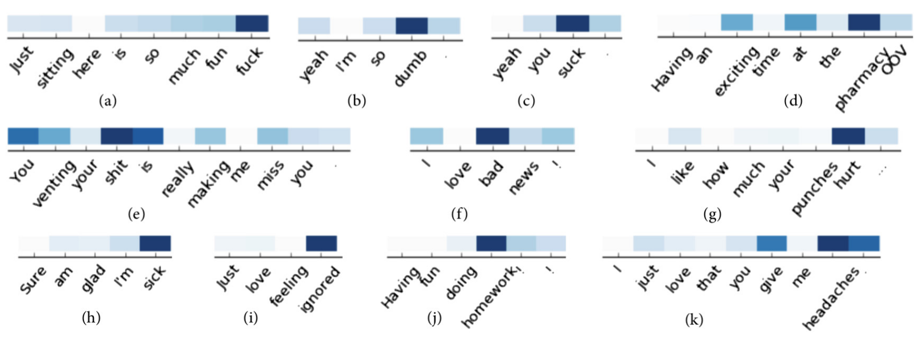

Fig. 1. Attention visualization

regards some words to have a negative sentiment, e.g., "headaches", "pharmacy", "homework" and "sick" in 1k, 1d, 1j, and 1h.

5.2 Attention Word Distribution

We also investigate on what kinds of words are more likely to be noticed by the Attentive RNN. Given a sentence s and an attention weight vector α, we make a ranking on words in s based on their corresponding attention weights in α. We calculate a reciprocal rank for each word w, which is the reciprocal of its ranking position i, and ignore those words with ranking position i higher than 3. We aim to find those words strongly emphasized by our model. We calculate each word's final reciprocal rank over all sentences in the dataset as follows.

$$ReciprocalRank_s^w = \begin{cases} \frac{1}{i}, & i \leq 3 \\ 0, & otherwise \end{cases}$$

$$ReciprocalRank^w = \sum_{\forall s} ReciprocalRank_s^w$$

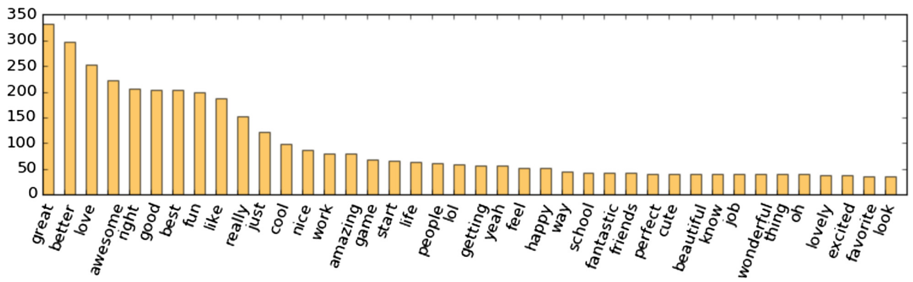

Fig. 2. Top 40 words ordered by the word reciprocal ranks.

Figure 2 shows the top 40 words ordered by their reciprocal ranks. Here the stop-words are removed. We can find some interesting linguistic features in irony sentences with our attention models. The following describes each of them in detail.

Positive Words. We can see that those having higher reciprocal ranks are mostly positive verbs or adjectives such as "like", "love", "best", "awesome", and "fantastic". That implies people are more likely to use positive words to convey their negative sentiment, thus our model pays more attention on them. One of the examples, "*So the debt ceiling was raised... awesome*", supports this view.

Hyperbole. In addition, some interjections ("oh", "yeah", and "lol") and adverbs ("fucking", "really", and "just") get more weight than other words. The phenomenon suggests that people tend to use dramatic utterance to give a hint to their audience what they said may not follow the literal meaning. For instance, "*What a beautiful day to BE IN SCHOOL ALL DAY YEAH.*".

Fact. We can find that there are some nouns in our top 40 words. They are denoted as fact-related words. For fact-related words, we get "work", "game", "life", "school", "people", "friends", and "job". When people complain about those things hard to change or negotiate with, they describe their situations with an ironic utterance to accentuate their disappointment or other negative sentiment, such as, "*School until 6 pm today, got such a beautiful life.*"

6 Conclusion

In this paper, the proposed Attentive RNN achieves the best performance with no further feature engineering. It increases the average F1 score to 89.8% in Ghosh et al. dataset. We further show that the attention vectors generated by our Attentive RNN captures specific words, which are useful to decide whether a tweet is ironic or not. With calculating the total reciprocal rank for each word, we find several linguistic styles popular in ironic utterances.

Acknowledgements. This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-104-2221-E-002-061-MY3 and MOST-105-2221-E-002-154-MY3.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of International Conference on Learning Representations (2015)
2. Barbieri, F., Saggion, H.: Modelling irony in Twitter. In: Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics, pp. 56–64 (2014)
3. Ghosh, D., Guo, W., Muresan, S.: Sarcastic or not: word embeddings to predict the literal or sarcastic meaning of words. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1003–1012 (2015)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
5. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751 (2014)
6. Kreuz, R., Glucksberg, S.: How to be sarcastic: the echoic reminder theory of verbal irony. J. Exp. Psychol. Gen. **118**(4), 374–386 (1989)

7. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Mller, U., Sckinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. In: Proceedings of International Conference on Artificial Neural Networks, pp. 53–60 (1995)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
9. Reyes, A., Rosso, P., Veale, T.: A multidimensional approach for detecting irony in Twitter. Lang. Resour. Eval. **47**(1), 239–268 (2013)
10. Tang, Y.-j., Chen, H.-H.: Chinese irony corpus construction and ironic structure analysis. In: Proceedings of the 25th International Conference on Computational Linguistics, pp. 1269–1278 (2014)
11. Tieleman, T., Hinton, G.: Lecture 6.5 - rmsprop. COURSERA: Neural Networks for Machine Learning, pp. 26–31 (2012)
12. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)