# YNU-HPCC at SemEval-2018 Task 3: Ensemble Neural Network Models for Irony Detection on Twitter

**Bo Peng, Jin Wang** and **Xuejie Zhang**
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
Contact:xjzhang@ynu.edu.cn

## Abstract

This paper describes our proposed to participate in the first year of the irony detection in English tweets competition. Previous works have demonstrated that long short-term memory models have achieved remarkable performance in natural language processing; moreover, combining multiple classifications from various individual classifiers is generally more powerful than a single classification. In order to obtain more precise irony detection classification, our system trains several individual neural network classifiers and combines their results according to the ensemble-learning algorithm.

## 1 Introduction

In most sentiment analysis tasks, recognition of the precise emotional polarity of a sentence forms the basis for further work. However, much of the corpus we used for analysis and training contains numerous sarcasm and irony features that will have a negative impact on the results of our analysis and training. For example, although the tweets provided by Twitter constitute a valuable and widely applicable corpus for many natural language processing tasks, Twitter users express their feelings and opinions on social networks with frequent irony (Amir et al., 2016). Therefore, such tweets may contain converse sentiments information compared their literal meaning. For example, *@someuser Yeah keeping cricket clean, that's what he wants #Sarcasm:* ignoring the hash tag, this tweet would be positive, which would miss lead an analysis system that uses these types of tweets as input.

Thus, it makes sense to discriminate whether a text is ironic, particularly for social network texts such as tweets. Further applications including tweet sentiment analysis, will benefit from automatic irony detection. The SemEval-2018 Twitter competition promotes research in this area, and is divided into two subtasks that involve binary and four-class classification.

Subtask A is a two-class (or binary) classification task whereby the system must predict whether or not a tweet is ironic. The subtask B is a multi-class classification task where the system has to predict one out of four labels describing i) verbal irony realized through a polarity contrast, ii) verbal irony without such a polarity contrast (i.e., other verbal irony), iii) descriptions of situational irony, and iv) non-irony (Cynthia Van Hee and Hoste, 2018). For a more detailed description, please see Carman et al. (2017).

In recent years, deep learning techniques have significantly outperformed traditional methods in several natural language processing (NLP) tasks (Cliche, 2017). In such task, several deep learning architecture-based methods have achieved outstanding performance in irony and sarcasm detection in social media. Silvio (Amir et al., 2016) presented a novel convolutional network-based method for learning user embeddings from their previous posts and used the user embeddings with lexical signals to recognize sarcasm. Ghosh and Veale (2016) proposed a combined convolutional neural network (CNN) model and long short-term memory (LSTM) method followed by a deep neural network (DNN), which also achieved an improvement compared to traditional machine learning approaches such as support vector machines (SVM).In this paper, we propose an ensemble of multiple deep learning models with a voting classifier in order to enhance the performance of individual neural network models for to detecting the ironic tweets. We trained six individual classifiers, including LSTMs, bi-directional LSTMs, gated recurrent units (GRUs), bi-directional GRUs, attention-based BiLSTMs and attention-based Bi-GRU. Thereafter, we use a voting mechanism to



*Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 622–627
New Orleans, Louisiana, June 5–6, 2018. 

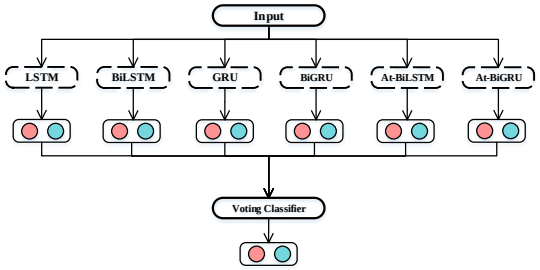


Figure 1: Ensemble voting classifier.

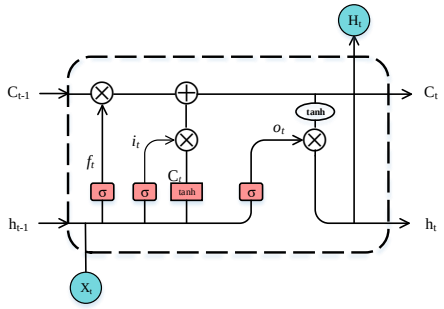


Figure 2: The LSTM memory cell

combine the results from the six classifiers in order to produce the final prediction label.

The remainder of this paper is organized as follows. In section 2, we describe the overall structure of our system and the LSTM-based models, as well as the selected individual classifiers. In section 3, we present the experimental results of our system, and conclusions are drawn in section 4.

## 2 System Description

### 2.1 Overview

Numerous previous research studies have demonstrated that the resulting classifier is generally more accurate than any of the individual classifiers making up the ensemble (Maclin and Opitz, 1999). For this reason, we decided to build our system following this strategy. Our system is based on ensemble learning and combined with various popular LSTM models. As illustrated in Figure 1, each classifier is a LSTM-based model, such as bidirectional LSTM (BiLSTM) and attention LSTM (AtLSTM). Each classifier is trained using the complete training set for that network. Following this, for each classifier, the predicted outputs of all classifiers are combined to produce the ensemble system output. As the ironic and non-ironic samples in the training set are evenly distributed (1911 irony samples; 3834 in total), and each classifier in our system is trained by the entire training set. Therefore, we selected the voting classifier as the combining scheme for our system. The principle of the voting classifier is the selection of the prediction supported by most of classifiers according certain rules. For example, if the predictions for a given sample are:

- classifier 1 - class 1
- classifier 2 - class 2
- classifier 3 - class 1

According to majority voting rules, the voting classifier would classify the sample as class 1.

Combining the output of several classifiers is useful only if disagreement exists among them. Thus, the selection of classifiers is rather important for our system.

Neural networks, particularly for recurrent neural networks (RNNs) (Mikolov et al., 2010), have achieved effective results in NLP. Owing to their circular network structure, which allows them to save previous information in a text sentence. Furthermore, conventional RNNs contain cyclic connections, making them powerful for modeling sequences. However, RNNs will face vanishing and exploding gradient problems when dealing with lengthy sequences. The LSTM, which is also a special type of RNN, was designed to address these problems. Therefore, we selected LSTM-based models as our individual classifiers.

### 2.2 LSTMs

The difference between the RNN and LSTM is that an LSTM (Sak et al., 2014) includes a different and more complex repeating module, as illustrated in Figure 2. This repeating module, also known as cell, provides the LSTM with the ability to discriminate whether input information is useful. A cell contains three gates, namely the input, forgotten and output gates. These gates determine the selection of information by means of the following formulae:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \end{aligned} \quad (1)$$

where $f_t$ and $i_t$ are the forgotten and retained features; $\sigma$ denotes the sigmoid function; $x_t$ and $h_t$ are the $t$-th input and output; and $W$ and $b$ are cell parameters.

Following this, the cell decides which new information will be stored in the cell state according

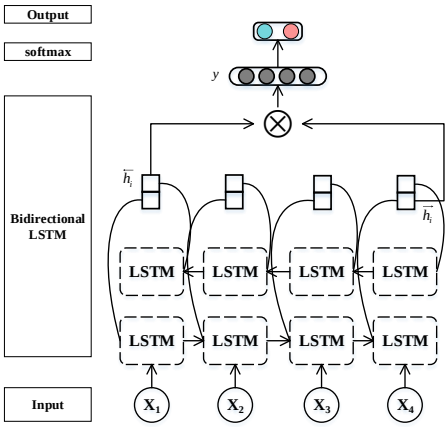


Figure 3: BiLSTM

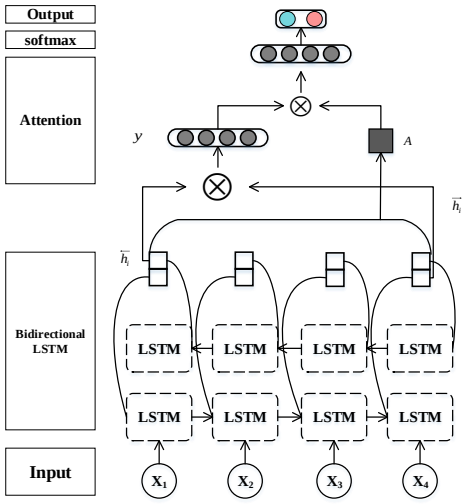


Figure 4: The LSTM memory cell

to the next equation. Here $\tilde{C}_t$ represents the candidate values, created by a tanh gate.

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \quad (2)$$

Finally, updating of the cell state and calculating the output of the cell are carried out according to the equations,

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ h_t &= o_t * \tanh(C_t) \end{aligned} \quad (3)$$

For an input tweet with length $t$, we firstly place it into an LSTM layer and generate vector $h_t$; then, use this vector to calculate the possibility of whether it is ironic by means of a softmax layer.

### 2.3 BiLSTMs

Standard RNNs use only the previous context and ignore the future context information when dealing with sequence texts. Bidirectional RNNs process the data in both directions with two separate hidden layers which then feed forward to the same output layer (Schuster and Paliwal, 1997). BiLSTMs replace the RNN cell with an LSTM cell based on BiRNNs, as illustrated in Figure 3. BiLSTMs compute the forward hidden state $\overrightarrow{h_t}$ and back forward hidden state $\overleftarrow{h_t}$, and then output the sequence $y$ by calculating equation (4), following which the output layer is updated.

$$y = \overrightarrow{h_t} \otimes \overleftarrow{h_t} \quad (4)$$

### 2.4 Attention BiLSTMs

LSTMs have promoted RNNs to a great extent in NLP, and a further significant step is the attention mechanism. We combined the attention mechanism (Wang et al., 2016) with the BiLSTM as illustrated in Figure 4. The attention captures the context information of an entire tweet, defined as follows:

$$\begin{aligned} e_t &= \tanh(W_e h_t + b_e) \\ \alpha_t &= softmax(e_t) \end{aligned} \quad (5)$$

where $W_e$ and $b_e$ denote the weight and bias, respectively, and $_t$ represents the attention vector that will be combined with the internal representation generated by BiLSTM layers. The remaining steps are consistent with the BiLSTMs.

### 2.5 GRUs

The GRU is a variant of LSTM. GRUs reduce the gating signals to two from LSTMs to two, namely reset and update gates. Although GRUs are simpler in terms of structure and calculation compared to LSTMs, their performance and efficiency in specific tasks are not reduced (Cho et al., 2014).

Therefore, we also trained three similar models using GRU cells. In total, we trained six individual classifiers: they are LSTM, BiLSTM, GRU, BiGRU, Attention BiLSTM and attention BiGRU.

## 3 Experiment

### 3.1 Datasets

The training dataset is constructed from 3834 English tweets collected by the organizers by means of searching Twitter for the hashtags #irony, #sarcasm, and #not. The training dataset for task A consists of tweets with a binary value score (0 or 1)

| Subtask A | Accuracy | F1 score (macro) | Recall (macro) | Precision (macro) |
|---|---|---|---|---|
| LSTM | 0.64163 | 0.64161 | 0.64174 | 0.65364 |
| BiLSTM | **0.64163** | 0.64161 | **0.64174** | **0.65364** |
| GRU | 0.63109 | 0.63108 | 0.63111 | 0.64031 |
| BiGRU | 0.64295 | 0.64099 | 0.64232 | 0.65110 |
| Attention BiLSTM | 0.64295 | 0.64085 | 0.64230 | 0.64519 |
| Attention BiGRU | 0.65744 | 0.65506 | 0.65673 | 0.66053 |
| Ensemble | 0.66007 | **0.66871** | 0.61894 | 0.62095 |

Table 1: Cross-validation results for subtask A.

| Subtask B | Accuracy | F1 score (macro) | Recall (macro) | Precision (macro) |
|---|---|---|---|---|
| LSTM | 0.7148 | 0.4939 | 0.5071 | 0.5341 |
| BiLSTM | 0.7330 | 0.4885 | 0.4964 | 0.5103 |
| GRU | 0.7369 | 0.5150 | 0.5017 | 0.5438 |
| BiGRU | 0.7031 | 0.4800 | 0.4946 | 0.5319 |
| Attention BiLSTM | 0.7200 | 0.5129 | 0.5315 | 0.5324 |
| Attention BiGRU | 0.7278 | 0.5081 | 0.5099 | 0.5216 |
| Ensemble | **0.7539** | **0.5172** | **0.5198** | **0.5385** |

Table 2: Cross-validation results for subtask B.

indicating whether the tweet is ironic. The training data for subtask B includes tweets with a numeric value corresponding to one of the subcategories, namely ironic by clash, other irony, situational irony and non-ironic. For subtasks A and B, the content of the tweet is exactly the same apart from the labels. The organizers also provided a version with no emoticons or hashtags and one with emoticons or hashtags. According to people's tweeting habits, emoticons and hashtags are important tools for expressing emotions, thus, we used tweets with these features for training.

### 3.2 Preprocessing

Before feeding the tweets to any classifier, they are pre-processed by following procedure:

- All uppercase letters are converted to lowercase.
- URLs are replaced by *<url>*; instance of @someone are replaced by *<user>*.
- Certain emoticons and emojis expressing positive sentiments are transformed into words such as *smile, like*, and *happy*. Others that express negative emotions are all replaced by *<irony>*.
- For subtask A, all hashtags are replaced by *<hashtag>*; for subtask B: except for *#irony, #sarcasm* and *#not*, all other hashtags are replaced by *<hashtag>*, and the remainder are all converted to the word *irony*.

We did not replace #irony, #sarcasm and #not with word irony for subtask A because it is easy for overfitting to occur while training. We consider that the reason for this is that the searching and labeling of these tweets mostly dependent on their hashtags. In the four-category subtask B, this does not lead to over-fitting, but aids in improving accuracy.

### 3.3 Word embedding

We obtain word embeddings by training with the corpus of English articles in Wikipedia pages using Global Vector (GloVe) (Pennington et al., 2014). Compared to Word2vec (Mikolov et al., 2013), GloVe achieves superior performance in this task under the same conditions. Moreover, we set the dimension of a single word as 300. Following the above steps, we create a look-up table that allows for most of the words in the training dataset to correspond to word vectors trained in advance, with the dataset containing 9056 unique words. However, 1266 words remain that cannot be matched, with most of these be-ing numbers and certain user-created words.

| | Accuracy | F1 score (macro) | Recall (macro) | Precision (macro) |
|---|---|---|---|---|
| Subtask A | 0.5089 | 0.4086 | 0.4277 | 0.3912 |
| Subtask B | 0.466 | 0.3127 | 0.3229 | 0.3384 |

Table 3: Evaluation for Subtask A and B.

### 3.4 Parameters

We used earlystopping to observe the accuracy value convergence of each epoch, with patience set to 3 and min_delta set to 0.05; we found that each model stopped training with no more than 35 epochs or even less. Consequently, we set the number of epochs to 30 for the training of every classifier. Furthermore, we set the batch size to 100 and drop-out rate to 0.25 for training of each model. We selected the categorical cross-entropy as the loss function and Adagrad as the optimizer (Duchi et al., 2011).

For subtasks A and B, the individual classifiers are trained with the training dataset. Ow-ing to the lack of development dataset, we only evaluated the performance of the classifiers and prevented overfitting by cross-validation. The models were implemented in Keras using TensorFlow backend.

### 3.5 Results and analysis

The experimental results of the individual classifiers and ensemble are displayed in Table 1 for subtask A and Table 2 for subtask B.

As indicated in Table 1, BiLSTM and attention GRU achieved a superior performance. However, there is no significant difference among the results of each model. This may be the reason why the ensemble does not operate effectively, because a good ensemble is one in which the individual classifiers are both ac-curate and create their errors in different parts of the input space (Maclin and Opitz, 1999). Our input space is not sufficiently large and the classifiers are similar, creating their errors in the same place.

For subtask B, our preprocessing strategy aids in improving accuracy. However, the samples in the training set are not as evenly distributed as subtask A, reflecting the ensemble effect. The precision achieved by our system achieved in subtask B ranks 10th out of 32 participants. However, numerous aspects of our system require further improvement.

The evaluation results from the committee are illustrated in Table 3. Due to our negligence, we submitted a wrong result of Subtask B. After the organizing committee reminded us that we have corrected the error and re-evaluated our result for Subtask b. Table 3 shows the corrected results for Subtask B. We apologize for the inconvenience caused by our own negligence and we thanked the organizers for prompt reminders so that we could correct the results in a timely manner.

## 4 Conclusion and future work

In this paper, we have presented the system we used to compete in SemEval-2018 task 3 - Irony detection in English tweets. The purpose of our participation in this competition is to deepen our understanding of irony detection as a novel NLP application. Moreover, we hope to determine an effective combination approach to ensemble learning and neural networks by means of practical application.

For future work, it would be meaningful to improve the neural network by combining the characteristic that ironic sentences are often inconsistent. Moreover, the goal is to identify superior practical ensemble methods to achieve improved performance in increased NLP applications.

## References


Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mrio J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. pages 167–177.

Mark J. Carman, Mark J. Carman, and Mark J. Carman. 2017. *Automatic Sarcasm Detection: A Survey*. ACM.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Mathieu Cliche. 2017. $BB_twtr$ at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms.

Els Lefever Cynthia Van Hee and Vronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):257–269.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *The Workshop on Computational Approaches To Subjectivity*.

R. Maclin and D. Opitz. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September*, pages 1045–1048.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. 26:3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Ha?im Sak, Andrew Senior, and Fran?oise Beaufays. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *Computer Science*, pages 338–342.

Mike Schuster and Kuldip K Paliwal. 1997. *Bidirectional recurrent neural networks*. IEEE Press.