THU_NGN at SemEval-2018 Task 3: Tweet Irony Detection with Densely Connected LSTM and Multi-task Learning

Chuhan Wu[1], **Fangzhao Wu**[2], **Sixing Wu**[1], **Junxin Liu**[1],
Zhigang Yuan[1] **and Yongfeng Huang**[1]
[1]Tsinghua National Laboratory for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University Beijing 100084, China
[2]Microsoft Research Asia
{wuch15,wu-sx15,ljx16,yuanzg14,yfhuang}@mails.tsinghua.edu.cn
wufangzhao@gmail.com

Abstract

Detecting irony is an important task to mine fine-grained information from social web messages. Therefore, the Semeval-2018 task 3 is aimed to detect the ironic tweets (subtask A) and their irony types (subtask B). In order to address this task, we propose a system based on a densely connected LSTM network with multi-task learning strategy. In our dense LSTM model, each layer will take all outputs from previous layers as input. The last LSTM layer will output the hidden representations of texts, and they will be used in three classification task. In addition, we incorporate several types of features to improve the model performance. Our model achieved an F-score of 70.54 (ranked 2/43) in the subtask A and 49.47 (ranked 3/29) in the subtask B. The experimental results validate the effectiveness of our system.

1 Introduction

Figurative languages such as irony are widely used in web messages such as tweets to convey different sentiment. Identifying the ironic texts can help to understand the social web better and has many applications such as sentiment analysis (Ghosh and Veale, 2016). Irony detecting techniques are important to improve the performance of sentiment analysis. For example, the tweet *"Monday mornings are my fave:)# not"* is an irony with negative sentiment, but it will be probably classified as a positive one by a standard sentiment analysis model (Van Hee et al., 2016b). Thus, capturing the ironic information in texts is useful to predict sentiment more accurately (Van Hee et al., 2016a).

However, determining whether a text is ironic is a challenging task since the the differences between ironic and non-ironic texts are usually subtle. For example, the tweet *"Love this weather #not"* is ironic, but a similar tweet *"Hate this weather #not happy"* is non-ironic. Different approaches are proposed to recognize the complex irony in texts. Existing methods to detect irony are mainly based on rules or machine learning techniques (Joshi et al., 2017). Rules based methods usually depend on lexicons to identify irony (Khattri et al., 2015; Maynard and Greenwood, 2014). However, these methods cannot utilize the contextual information from texts. Traditional machine learning based methods such as SVM (Desai and Dave, 2016) are also effective in this task, but they usually need manually feature engineering (Barbieri et al., 2014). Recently, deep learning techniques are successfully applied to this task. For example, Ghosh et al. (2016) propose to use a CNN-LSTM model to classify the ironic and non-ironic tweets. Their method can significantly improve the classification performance without heavy feature engineering. However, existing methods are aimed to detect irony in tweets with explicit irony related hashtags. For example, tweets with #irony or #sarcasm hashtags are very likely to be ironic. Therefore, models may focus on these hashtags rather than the contextual information.

To fill this gap, the SemEval-2018 task 3[1] aims to detect irony of tweets without explicit irony hashtags (Van Hee et al., 2018). The subtask A is aimed to determine whether a tweet is ironic. the subtask B is aimed to identify the irony types of tweets: Verbal irony by means of a polarity contrast, other verbal irony and situational irony. Several examples are as follows:

- **verbal irony by means of a polarity contrast**: I love waking up with migraines #not
- **other verbal irony**: @user Yeah keeping cricket clean, that's what he wants #Sarcasm

[1]https://competitions.codalab.org/competitions/17468

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 51–56
New Orleans, Louisiana, June 5–6, 2018.

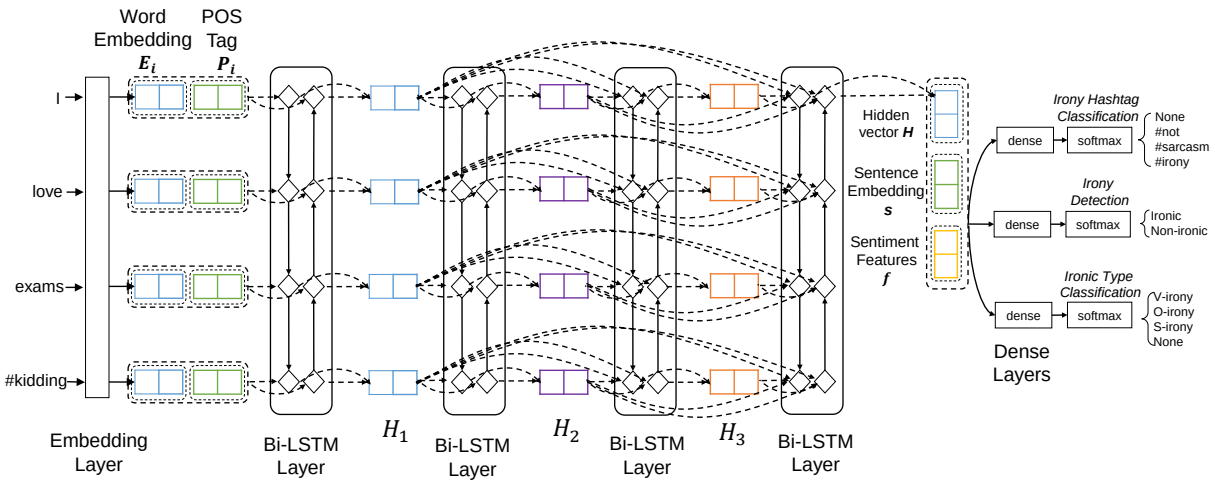

Figure 1: Architecture of our Dense-LSTM model. The V-irony, O-irony and S-irony denote the three different irony types respectively (Van Hee et al., 2018).

- **situational irony**: most of us didn't focus in the #ADHD lecture. #irony

In order to address this problem, we propose a system[2] based on a densely connected LSTM model (Wu et al., 2017) with multitask learning techniques. In our model, each LSTM layer will take all outputs of previous LSTM layers as input. Then different levels of contextual information can be learned at the same time. Our model is required to predict in three tasks simultaneously: 1) identifying the missing irony related hashtags; 2) classify ironic or non-ironic; 3) irony type classification. By using multitask learning strategy, the model can combine the information in the different tasks to improve the performance. The experimental results in both subtasks validate the effectiveness of our method.

2 Densely Connected LSTM with Multi-task Learning

The architecture of our densely connected LSTM model is shown in Figure 1. We denote this model as *Dense-LSTM*. The detailed information will be introduced in the following paragraphs.

In our model, the embedding layer is used to convert the input tweets into a sequence of dense vectors. The POS tag features P_i are one-hot encoded and concatenated with the word embedding vectors E_i. Usually the affective words and creative languages in tweets are important irony clues. Since these words usually have specific POS tags, adding these features can help our model to capture the ironic information better. We use tweetokenize[3] tool to tokenize and the Ark-Tweet-NLP[4] tool to obtain the POS tags of tweets (Owoputi et al., 2013).

The first Bi-LSTM layer takes the sequential vectors as input. For the j_{th} Bi-LSTM layer, its output H_j will input all LSTM layers after it. As shown in Figure 1, the blue dashed lines represent such over-layer connections. All inputs of an LSTM layer will be concatenated together. Thus, the input of the j_{th} ($j > 1$) layer is $[H_1; ...; H_{j-1}]$. It indicates that each layer can learn different levels of information at the same time. Since the irony information is complex, jointly using all levels of information is beneficial to predict irony more accurately. The last LSTM layer will output the hidden representation H of texts. It will be concatenated with the sentiment features and the sentence embedding features. The sentiment features can provide additional sentiment information to detect irony, such as the sentiment polarity assigned by lexicons. The sentiment features are generated via the AffectiveTweets[5] package in weka provided by Mohammad et al. (Mohammad and Bravo-Marquez, 2017). We use the TweetToLexiconFeatureVector (Bravo-Marquez et al., 2014) and TweetToSen-

[2]https://github.com/wuch15/SemEval-2018-task3-THU_NGN.git

[3]https://github.com/jaredks/tweetokenize

[4]http://www.cs.cmu.edu/ ark/TweetNLP

[5]https://github.com/felipebravom/AffectiveTweets

tiStrengthFeatureVector (Thelwall et al., 2012) filters in this package. The embedding of a sentence is obtained by taking the average of all words in this sentence using the 100-dim pre-trained embedding weights provided by Bravo et al. (Bravo-Marquez et al., 2016). By incorporating the vector representation of tweet sentence, the irony information can be easier to be captured.

Three dense layers with ReLU activation are used to predict for three different tasks including: determining the missing ironic hashtags (i.e. #not, #sarcasm, #irony or none of them) *(task1)*; identifying ironic or non-ironic *(task2)* ; identifying the irony types *(task3)*. Thus, the objective function of our model can be formulated as:

$$\mathcal{L} = \alpha_1 \mathcal{L}_1 + \alpha_2 \mathcal{L}_2 + \alpha_3 \mathcal{L}_3, \quad (1)$$

where $\mathcal{L}_i$ and α_i denote the loss function and its weight of task i. $\mathcal{L}_1$ and $\mathcal{L}_2$ are categorical and binary cross-entropy respectively. In addition, the numbers of tweets with different irony types are very unbalanced. Motivated by the cost-sensitive entropy used by Santos et al. (2009), we formulate $\mathcal{L}_3$ as follows:

$$\mathcal{L}_\ni = -\sum_{i=1}^{N} w_{y_i} y_i \log(\hat{y_i}), \quad (2)$$

where N is the number of tweets, y_i is the irony type of the i_{th} tweet, $\hat{y_i}$ is the prediction score, and w_{y_i} is the loss weight of irony type label y_i. w_{y_i} is defined as $\frac{\sum_{k=1}^{C} N_k}{N_{y_i}}$, where C is the number of irony types and N_j is the number of tweets with irony type label j. Thus, the infrequent irony types will gain relatively larger loss weights. By using this multi-task learning method, our model can incorporate different information such as the irony hashtags. In addition, classifying ironic/non-ironic and the irony types are similar tasks. Therefore, the performance of both tasks can be improved by combining the information of both tasks.

In order to improve the performance of our system, we use an ensemble strategy by averaging the classification results predicted by 10 models. Each model will be trained using a random dropout rate. Therefore in this way, the classification results will be voted by different models, which can improve the model performance.

3 Experiment

3.1 Dataset and Experimental Settings

The detailed statistics of the dataset[6] in this task are shown in Table 1. V-irony, O-irony and S-irony represent the three types respectively: verbal irony by means of a polarity contrast, other types of verbal irony and situational irony (Van Hee et al., 2018). In subtask A, the performance of systems is evaluated by F-score for the positive class. In subtask B, the macro-averaged F-score over all classes is used as the metric.

Task	A		B			
Label	Ironic	Non-ironic	V-irony	O-irony	S-irony	Non-ironic
#train	1911	1923	1390	316	205	1923
#test	311	473	164	85	62	473

Table 1: The detailed statistics of the dataset.

We combine two pre-trained word embeddings: 1) the embeddings provided by Godin et al. (2015), which are trained on a corpus with 400 million tweets; 2) the embeddings provided by Barbieri et al. (2016), which are trained on 20 million tweets. The dimensions of them are 400 and 300 respectively. They are concatenated together as the embeddings of words.

In our network, the Dense-LSTM model has 4 LSTM layers with 200-dim hidden states. The hidden dimensions of dense layers are set to 300. The dropout rate of each layer is set to a random number between 0.2 to 0.4, and it will be set to a fixed value 0.3 in the comparative experiments without ensemble strategy. In subtask A, the loss weights α of the three task are set to 0.5, 1 and 0.5 respectively. In subtask B, they are 0.5, 0.5 and 1. We use RMSProp as the optimizer, and the batch size is set to 64. In addition, we use 10% training data for validation to select the hyperparameters above.

3.2 Performance Evaluation

We compare the performance of different methods including: 1) SVM, the benchmark system using SVM and BOW model; 2) CNN, using CNN with a global average pooling layer to obtain the hidden vector h, which is used to predict in the three tasks; 3) LSTM, using one Bi-LSTM layer in the network to get h; 4) 2-layer LSTM, using 2 Bi-LSTM layers; 5) Dense-LSTM, using our

[6]https://github.com/Cyvhee/SemEval2018-Task3/tree/master/datasets

Dense-LSTM model; 6) Dense-LSTM+ens, using our Dense-LSTM model and ensemble strategy. In addition, we apply multi-task learning technique to all models except the benchmark system based on SVM. The results are shown in Table 1. The experimental results show that our Dense-LSTM model significantly outperforms the baselines. Since the layers in our Dense-LSTM can learn from all previous outputs, our model can combine different levels of contextual information to capture the high-level irony clues. In addition, our model can predict more accurately via ensemble. Since models with random dropout can extract different information, we can take advantage of all models by voting. The ensemble strategy can reduce the noise in the dataset and make our system more stable (Xia et al., 2011).

Model	Subtask A			Subtask B
	P	R	F	Macro-F
Baseline	54.78	62.70	58.47	32.69
CNN	59.32	61.41	60.35	45.30
LSTM	57.73	67.20	62.11	45.76
2-layer LSTM	60.34	68.49	64.16	47.16
Dense-LSTM	62.78	72.69	67.36	48.28
Dense-LSTM+ens	**63.04**	**80.06**	**70.54**	**49.47**

Table 2: The performance of different methods. P, R, F represent precision, recall and F-score respectively.

3.3 Effectiveness of Multi-task Learning

The performance of our Dense-LSTM model using different combinations of training tasks is shown in Table 3. Note that we don't apply model ensemble here. Compared with the models trained in task2 or task3 only, the combination of both tasks can improve the performance. It may be because the two tasks have inherent relatedness and can share rich mutual information. Learning to predict the missing ironic hashtags (task1) can also improve the model performance. Since the ironic hashtags are often important ironic clues, identifying such clues can help our model to mine ironic information better.

3.4 Influence of Pre-trained Word Embedding

We compare the performance using different combinations of pre-trained embeddings in our model. The results are illustrated in Table 4. The results show that the pre-trained embeddings are important to capture irony information, and using the combination of two different word embeddings can improve the model performance. It proves that this method can reduce the out-of-vocabulary words in the single embedding file and provide richer semantic information.

Task Combination	Subtask A			Subtask B
	P	R	F	Macro-F
task2	60.05	71.06	65.10	-
task3	-	-	-	44.65
task2+task3	61.81	72.34	66.67	46.94
task1+task2	61.33	71.38	65.97	-
task1+task3	-	-	-	45.57
task1+task2+task3	**62.78**	**72.69**	**67.36**	**48.28**

Table 3: The performance in two subtasks using different combinations of training tasks.

Feature	Subtask A			Subtask B
	P	R	F	Macro-F
w/o pre-trained	56.25	67.14	61.21	42.28
+emb1	60.96	69.95	65.14	47.69
+emb2	61.77	70.59	65.89	47.24
+emb1 +emb2	**62.78**	**72.69**	**67.36**	**48.28**

Table 4: Influence of pre-trained word embedding. The emb1 and emb2 denote the embeddings provided by Godin et al. (2015) and Barbieri et al. (2016) respectively.

3.5 Influence of Additional Features

The influence of different features on our model is shown in Table 5. According to this table, all features can improve the classification performance in both subtasks, and the combination of the three features can achieve better performance. The improvement brought by POS tags is most significant. Affective words are important irony clues and they are usually verbs, adjectives or hashtags. Thus, incorporating the POS tag features can help to identify these words and capture the ironic information better. The sentiment features also improve our model, which can be inferred from the results. The sentiment polarities of ironic tweets are usually negative, but these texts often contain positive sentiment words. Since our sentiment features are obtained by several different sentiment or emotion lexicons, they can be used to assign the sentiment scores of texts, which can provide rich information to detect irony. The sentence embedding can also slightly improve the performance. The sentence embedding contains information of each word in the sentence. Thus, it can help to capture the word information better, which is ben-

eficial to identify the overall sentiment of texts. The combination of all three types of features can take advantage of them and gain significant performance improvement. It validates the effectiveness of each type of features.

Feature	Subtask A			Subtask B
	P	R	F	Macro-F
None	59.84	70.42	64.70	45.56
+*POS tags*	61.04	72.03	66.08	46.61
+*Sentiment Features*	61.16	71.38	65.88	46.37
+*Sentence Embedding*	61.39	71.06	65.87	46.24
+*All Features*	**62.78**	**72.69**	**67.36**	**48.28**

Table 5: Influence of different features on our model.

4 Conclusion

Detecting irony in web texts is an important task to mine fine-grained sentiment information. In order to address this problem, we develop a system based on a densely connected LSTM model to participate in the SemEval-2018 Task 3. In our model, every LSTM layer will take all outputs of previous layers as inputs. Thus, the different levels of information can be learned at the same time. In addition, we propose to combine three different tasks to train our model jointly, which includes identifying the missing irony hashtags, determining ironic or non-ironic and classifying the irony types. These tasks have inherent relatedness thus the performance can be improved by sharing the mutual information. Our system achieved an F-score of 70.54 and 49.47 which ranked the 2nd and 3rd place in the two subtasks. The experimental results validates the effectiveness of our method.

Acknowledgments

The authors thank the reviewers for their insightful comments and constructive suggestions on improving this work. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800402 and in part by the National Natural Science Foundation of China under Grant U1705261, Grant U1536207, Grant U1536201 and U1636113.

References

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–58.

Felipe Bravo-Marquez, Eibe Frank, Saif M Mohammad, and Bernhard Pfahringer. 2016. Determining word-emotion associations from tweets by multi-label classification. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*, pages 536–539. IEEE.

Felipe Bravo-Marquez, Marcelo Mendoza, and Barbara Poblete. 2014. Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, 69:86–99.

Nikita Desai and Anandkumar D Dave. 2016. Sarcasm detection in hindi sentences using support vector machine. *International Journal*, 4(7):8–15.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 161–169.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):73.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark Carman. 2015. Your sentiment precedes you: Using an authors historical tweets to predict sarcasm. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 25–30.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Lrec*, pages 4238–4243.

Saif M Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. *arXiv preprint arXiv:1708.03700*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Raúl Santos-Rodríguez, Darío García-García, and Jesús Cid-Sueiro. 2009. Cost-sensitive classification based on bregman divergences for medical diagnosis. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 551–556. IEEE.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163–173.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016a. Exploring the realization of irony in twitter data. In *LREC*.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016b. Monday mornings are my fave:)# not exploring the automatic recognition of irony in english tweets. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2730–2739.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 Task 3: Irony Detection in English Tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, SemEval-2018, New Orleans, LA, USA. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, Sixing Wu, and Zhigang Yuan. 2017. Thu_ngn at ijcnlp-2017 task 2: Dimensional sentiment analysis for chinese phrases with deep lstm. *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 47–52.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138–1152.