A Novel Recursive Network for Irony Detection in Tweets

Arthur Cruz de Araujo, Vandad Davoodnia, Guangyi Zhang, Ali Etemad, and Xiaodan Zhu [*]

Abstract

In this paper, we present a recursive-neural-network-based architecture as the solution for irony detection. We design, train, and evaluate an ensemble of several models based on our proposed model, namely recursive attention based Bi-LSTM, to attain the predictions for each task. We perform a comprehensive comparison of different publicly available word embeddings, position tagging, and recursive elements. Our results outperform the existing state of the art in the SemEval-2018 irony detection task, by achieving an F1 score of 0.796 and 0.579 on Task A and B, respectively.

1 Introduction

Sentiment analysis, also known as opinion mining, is the field of study that analyzes peoples opinions, attitudes, and sentiments toward entities and their attributes expressed in text (Zhao et al., 2016). It essentially encompasses a wide range of tasks, like sentiment classification, subjectivity analysis, abusive language detection, and review mining.

Opinions are central to most human activities and heavily influence our behavior. Our perception of reality and choices are highly conditioned on how others evaluate the world, reason why it is so natural for entities to seek out the opinions of others before making decisions (Zhang et al., 2018). Such motivation for mining public's critic, and therefore acting on it, has driven the development of automated sentiment analysis systems.

Sentiment analysis research dates back to 2001 when Kamps et al. explored the structure of the WordNet lexical database to assess the affective or emotive meaning of words. Later, Wilson et al. (2005) used an ensemble of classifiers to recognize context polarities in sentences. Interest on the field has grown over the last few years, evidenced, for example, by the number of participants in SemEval campaigns on tasks from the classification of positive/negative opinions (2015) to offense identification and categorization (2019) in Twitter.

Among the solutions submitted for the irony task, Wu et al. (2018) achieved the state-of-the-art in Task A and the 3rd place in Task B, using four densely connected Bi-LSTM layers with a multi-task learning strategy. They utilized two pre-trained embeddings and concatenated them with Pos tagging to enhance semantic representation. Baziotis et al. ranked 2nd on both tasks, but after some post-task-completion enhancements, outperformed the state-of-the-art on both. They used two models, operating independently at word and character level, each using two Bi-LSTM layers for semantic representation. Unlike Wu et al., this work trained its own embeddings on 550 million tweets for word-level. On another work, Ghosh and Veale achieved the best result for Task B using a Siamese Neural Network (SNN) (Bromley et al., 1994). They used two identical networks, composed of an LSTM layer and a word embedding layer, which are fed by two fragments of each input sentence for training; then, a subtract layer captured the weight difference in the sub-networks, so a tweet would be taken for non-ironic if the two parts were semantically consistent, and ironic otherwise.

In the last decade, a wide range of recursive neural networks has been proposed for sentence modeling. For instance, Socher et al. applied the same neural network in a recursive manner to each node of a directed acyclic graph, defined in word level to extract sentence and phrase based embedding on Wall Street Journal dataset. Similarly, Kalchbrenner et al. used a convolutional neural network with Dynamic k-Max Pooling to do sentiment classification. They applied the same network recursively in order to group words from a sentence to extract a general, sentence level sentimental vector. There are two common principles in the mentioned works and our paper. First, the decoder part of the neural networks, which holds the trainable weights, is shared in each recursive

[*] The first three authors have equally contributed and the names are given in alphabetical order.

step, and secondly, at each stage, some information from the previous stage is passed on to the next stage. However, our paper proposes to use the same embedding vector at each time step, given the attention-modulated response from the previous stage. We also investigate the effect of shared weights and the recursive steps to further explore the effectiveness of our recursive architecture.

In this paper, we propose a novel deep learning based solution to identify irony in Twitter. Our model is built on a recursive Bi-LSTM based architecture with attention mechanisms. Furthermore, the effects of various word embeddings and hyper-parameters on the performance are also investigated.

2 Problem Formulation

Our proposed model is tested on Irony detection, which is a recently published SemEval task, subdivided into different tasks. The third task proposed for the SemEval-2018 competition relates to the automated detection of irony in Twitter (Van Hee et al., 2018). Within Task 3, two different tasks are defined: Task A is a binary classification problem, for determining if a tweet is ironic or not, while Task B is a multi-class tweet classification among four classes. The dataset provided for this task comprises a total of 4792 samples, 3834 of them reserved for training. Regarding the binary classification of Task A, the data is balanced between ironic and non-ironic tweets. On the other hand, it is highly unbalanced for Task B, where 36.25%, 8.24%, and 5.35% of the data are respectively for verbal ironies by polarity contrast, verbal ironies without explicit polarity, and instances of situational irony.

3 Materials and Methods

This section describes the general pipeline of our models, introducing several methodologies and models that were explored throughout this paper.

3.1 Preprocessing

We utilized *ekphrasis* (Baziotis et al., 2017) as a tool for pre-processing the tweets. For this part, we tokenized Twitter-specific words, applied spell correction and word normalization (e.g. URL, @user), and finally, we did word segmentation for splitting hashtags and annotating the words (e.g. all caps). Word tokenization has proven to be an effective way of dealing with the challenge of capturing and normalizing the information of specific social media words, including emojis and dates which are not available in other corpora.

3.2 Word Embeddings

Word embeddings are generally used to map word tokens to vectors, representing them in a space that preserves semantics and/or syntactic similarities. In this paper, we perform extensive experimentation across various word embeddings. We consider using pre-trained ones, trained on tweets corpus from (Baziotis et al., 2018), using word2vec (Mikolov et al., 2013), and Gigaword5 twitter from using GloVe (Pennington et al., 2014). Finally, we make use of position tagging in the embeddings and analyze its performance gain for showing its effectiveness.

3.3 Classification Model

In this paper, we explore a recursive model based on Bi-LSTM plus attention, all explained in the following.

Bi-LSTM: an LSTM block has three gates: the *forget gate* $\mathbf{f}_t$ controls the amount of information from the previous state $\tilde{\mathbf{c}}_{t-1}$ to be discarded, the input gate $\mathbf{i}_t$ determines how much new data flows into the LSTM cell, and the output gate $\mathbf{o}_t$ how much of the cell current state is exposed in $\mathbf{h}_t$. All operate based on the previous hidden state and the current input, similarly defined as follows:[1]

$$\begin{aligned}\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_t), &(1)\\ \mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), &(2)\\ \mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o). &(3)\end{aligned}$$

The *candidate* cell state $\tilde{\mathbf{c}}_t$ is then similarly computed, but with a different activation function:

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c). \quad (4)$$

The actual cell state $\mathbf{c}_t$ uses $\mathbf{f}_t$ and $\mathbf{i}_t$ to respectively balance $\tilde{\mathbf{c}}_{t-1}$ (previous) and $\tilde{\mathbf{c}}_t$ (*candidate*):

$$\mathbf{c}_t = \mathbf{f}_t \circ \tilde{\mathbf{c}}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t \quad (5)$$

Finally, the current hidden state $\mathbf{h}_t$ is defined by the cell state $\mathbf{c}_t$ under the effect of the output gate:

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t). \quad (6)$$

The same gating mechanisms apply to the so-called bidirectional LSTM (Bi-LSTM), but this

[1] $\mathbf{W}_k$ and $\mathbf{b}_k$, with $k \in \{f, i, o, c, s\}$, are weights and biases that can be obtained through training by BPTT.

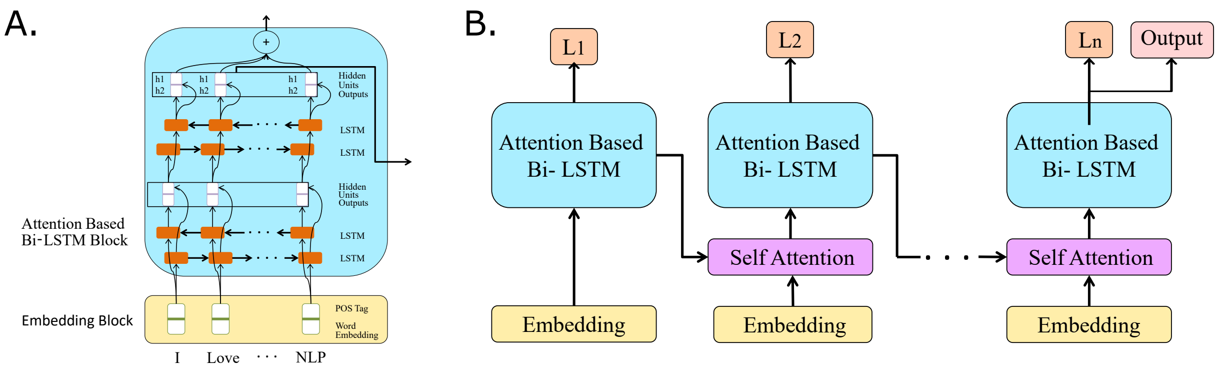

Figure 1: Our proposed pipeline is illustrated. A shows the main block of our model, and B shows the general overview of the model architecture, by which the loss at the end of each block is averaged and the output is only taken from the last block.

concatenates, at each time step, the hidden states of two LSTMs of opposite directions. Bi-LSTMs have outperformed LSTMs in speech recognition (Graves and Schmidhuber, 2005), by learning information from the sequence in both start-to-end and end-to-start directions. Accordingly, after the embedding layers, we use Bi-LSTM layers to extract bidirectional information from sequences.

Attention Mechanism: to rely only on the last time step of the final Bi-LSTM layer, despite the bidirectional advantage, might still lead to missing important dependencies due to gradient vanishing on long sequences. To address that, we used a soft attention layer, described by the equations

$$\mathbf{u}_t = \tanh(\mathbf{W}_s \mathbf{h}_t + \mathbf{b}_s)\ , \tag{7}$$

$$\alpha_t = \frac{exp(u_t)}{\sum_t exp(u_t)}\ , \tag{8}$$

$$\mathbf{v} = \sum_t \alpha_t \mathbf{h}_t\ , \tag{9}$$

where $\mathbf{v}$, the attention's output, summarizes the information at the last Bi-LSTM layer, assigning different weights to the learned representations.

Recursive Mechanism with Attention: our final model, illustrated in Figure 1, comprises several blocks of Bi-LSTM+Attention in a recursive manner of several stages, and shared weights across the blocks, where the concatenated output of the last hidden layer of each block modulates the attention on the embeddings of the next stage. We use self-attention and multi-head attention as described in Vaswani et al., formulated as:

$$\textsc{Att}(Q,K,V) = softmax(QK^T/\sqrt{d_k})V, \tag{10}$$

$$head_i = \textsc{Att}(QW_i^Q, KW_i^K, VW_i^V), \tag{11}$$

$$\text{M-}\textsc{Head}(Q,K,V) = [head_1, ..., head_h]W^O, \tag{12}$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_v}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are trainable weights and h is the number of parallel attention heads.

Finally, a dense layer with SoftMax activation follows to perform both binary and multi-class classifications. The general overview of our model is illustrated in Figure 1.

4 Experimental Setup

We perform several experiments using PyTorch (Paszke et al., 2017) with different embedding layers and deep classifiers in order to find the best solution for each task. The details of our planned experiments are described in the following sections.

4.1 Evaluation Metrics

We adopt the same performance metrics used on the SemEval competition (Van Hee et al., 2018). Given the True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN), the usual metrics for classification are Accuracy, Precision, Recall and F1-Score, defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = 2 \times \frac{Recall \cdot Precision}{Recall + Precision}. \tag{16}$$

4.2 Hyper-Parameters

We train our models with different sets of hyper-parameters for each task. In Task A, we used 4 layers of Bi-LSTM with 5 recursive stages and 10 attention heads. For regularization, we used an embedding Gaussian noise of 0.3, a dropout rate

Table 1: Effect of word embeddings for Task A

Feature	Pos Tag	Acc	Prec	Rec	F1
emb1	Yes	0.744	0.744	0.744	0.740
	No	0.731	0.744	0.755	0.720
emb2	Yes	**0.802**	**0.794**	**0.800**	**0.796**
	No	0.718	0.715	0.723	0.713
emb3	Yes	0.750	0.767	0.770	0.748
	No	0.714	0.701	0.701	0.700
Char-level	N/A	0.622	0.622	0.617	0.616
ENS-MV	N/A	0.765	0.765	0.757	0.759
ENS-UA	N/A	0.765	0.765	0.757	0.759

Table 2: Effect of word embeddings for Task B

Feature	Pos Tag	Acc	Prec	Rec	F1
emb1	Yes	0.660	0.524	0.576	0.519
	No	0.591	0.529	0.594	0.511
emb2	Yes	**0.708**	0.570	0.472	**0.579**
	No	0.678	0.558	0.570	0.525
emb3	Yes	0.662	0.541	0.581	0.532
	No	0.634	0.533	**0.616**	0.550
Char-level	N/A	0.562	0.352	0.376	0.358
ENS-MV	N/A	0.687	**0.616**	0.556	0.548
ENS-UA	N/A	0.687	0.616	0.589	0.548

of 0.2 on the final output of Bi-LSTM+Attention block, and another dropout right after the embedding layer. Moreover, the rest of the layers used a dropout rate of 0.1 and a batch normalization layer was utilized between the embedding, self-attention, and Bi-LSTM blocks. We trained our model with an Adam optimizer (Kingma and Ba, 2014) at a constant learning rate of 0.002.

Similarly to Task A, we trained our model for Task B using 2 Bi-LSTM layers and 7 recursive stages. The embedding noise, embedding dropout, and the encoder dropout rate were chosen 0.2, 0.1, and 0.1 respectively. The rest of the hyper-parameters and training procedure were the same as in Task A.

4.3 Effect of Different Pre-trained Word Embeddings

This paper performs evaluation using several pre-trained word embeddings: **emb1:** 300 dimensions embeddings provided by (Pennington et al., 2014) trained on common crawls; **emb2:** 200 dimensions embeddings provided by (Pennington et al., 2014) trained on 2 billion tweets; **emb3:** 300 dimensions embeddings provided by (Baziotis et al., 2018) trained on 550 million archived tweets.

As seen from Table 1 and 2, our model achieves the best results for both tasks using the embeddings provided by Pennington et al.. Both Majority Voting Ensemble approach (ENS-MV) and Unweighted Average Ensemble approach (ENS-UA) achieve very similar results on all four metrics. In both two tasks, the ensemble approaches outperform the word-level solution with following two embeddings: emb1 and emb3. However, the word-level solutions with the emb2 have the best results in general. Effect of different word embeddings combinations with or without position tagging is also investigated, which boosts the performance of almost all of the embeddings.

5 Results and Discussion

We experimented with shared and non-shared weights for the encoder part of our model. We observed a relatively lower testing accuracy when performing the same experiment with non-shared weights in the encoder, inferring existence case of over-fitting.

In Task A, our model outperforms the state-of-the-art solutions in the following metrics: accuracy, precision, and F1 scores, as shown from Table 3; in Task B, our model is achieving a better F1 score than the state-of-the-art, as shown from Table 4. It must be noted that as the data for Task B was highly unbalanced, In the SemEval campaign, the competition teams were ranked by their F1 scores.

Table 3: Comparison of results for Task A

Team	Acc	Prec	Rec	F1
NTUA-SLP	0.788	0.786	0.799	0.786
THU_NGN	0.735	0.630	0.801	0.705
WLV	0.643	0.532	**0.836**	0.650
Ours	**0.802**	**0.794**	0.800	**0.796**

Table 4: Comparison of results for Task B

Team	Acc	Prec	Rec	F1
Unknown	**0.732**	**0.577**	0.504	0.507
NTUA-SLP	0.699	0.543	**0.579**	0.536
THU_NGN	0.605	0.486	0.541	0.495
Ours	0.708	0.570	0.472	**0.579**

6 Conclusions

In this paper, we proposed a model based on a recursive attention Bi-LSTM. We show that by choosing the best embedding combined with positional embedding, it achieves superior performance to other models. The best evaluation metrics obtained were as follows: an accuracy of 0.802, precision of 0.794, and F1 score of 0.796 are achieved in Task A; an F1 score of 0.579 is achieved in Task B, showing the superiority of our model compared to the existing state-of-the-art in Twitter irony detection. In the future, we plan to test the novel recursive model on other semantic tasks to further develop and study its capability.

References

Christos Baziotis, Athanasiou Nikolaos, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 613–621, New Orleans, Louisiana. Association for Computational Linguistics.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. 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)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

Aniruddha Ghosh and Tony Veale. 2018. Ironymagnet at semeval-2018 task 3: A siamese network for irony detection in social media. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 570–575.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Jaap Kamps, Maarten Marx, Robert J Mokken, and Marten de Rijke. 2001. *Words with attitude*. Citeseer.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

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

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, volume 2010, pages 1–9.

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*, pages 39–50.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. Thu_ngn at semeval-2018 task 3: Tweet irony detection with densely connected lstm and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Jun Zhao, Kang Liu, and Liheng Xu. 2016. Sentiment analysis: mining opinions, sentiments, and emotions.