24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Modeling the Incongruity between Sentence Snippets for Sarcasm Detection

Hongliang Pan[1,2] and **Zheng Lin**[1] and **Peng Fu**[1,*] and **Weiping Wang**[1]

Abstract. Sarcasm is a form of irony used to mock or convey contempt, which occurs when there is an incongruity between the literal meaning of an utterance and its intended meaning. Many studies identify sarcasm by capturing the incongruity in-between the words. However, consider the following example, “I love waking up at 4 am on Saturday”, there is no apparent incongruity in-between the words. Intuitively, the word “love” and the snippet “waking up at 4 am on Saturday” form a strong contrast. Thus, capturing the incongruity among the sentence snippets is more reasonable since a sentence snippet usually contains more semantic information than a single word. Additionally, not all snippets are equally important when human beings identify sarcasm. Thus, inspired by the above observations, we propose the Self-Attention of Weighted Snippets (SAWS) model for sarcasm detection, which overcomes the problem that the previous models are inefficient in determining the sarcasm caused by snippet incongruity. The experiment results show that our model achieves state-of-the-art performance on four benchmark datasets, including two short text Twitter datasets and two long text Internet Argument Corpus (IAC) datasets.

1 Introduction

Sarcasm is a form of figurative language which is commonly used in social media and online forum. Due to its linguistical nature, sarcasm can completely flip the polarity of opinions. Failure to detect sarcasm may result in poor performance in sentiment analysis [25]. Sarcasm can be treated as a form of implicit sentiment expression. Most studies are currently focused on explicit sentiment analysis. However, considering both explicit and implicit sentiment expression may contribute to a series of applications, such as election prediction, product review and customer support.

Sarcasm detection is a challenging task in sentiment analysis due to its figurative nature [19]. Consider some sarcastic examples in Figure 1. We can observe that all these examples involve a disparity between the author’s intention and the events. This concept is defined as “incongruity” by [9], which states that “verbal irony is recognized as a technique of using incongruity to suggest a distinction between reality and expectation”. In the given examples, without looking at the incongruous snippets, it might be difficult to conclude whether the sentence is sarcastic or not. For instance, “Going in to work for 2 hours was totally worth the 35 min drive”. It is insufficient to classify the sentence as sarcasm by only looking in-between the words.

- Going in to work for 2 hours was totally worth the 35 min drive !
- I saw a guy walking 4 dogs this morning and thought wow ! that guy must be really blind .
- A little nervous to start school 😝 5 classes in one day should be fun …
- So excited for my family hike at 9 freaking o'clock in the morning ! I love not sleeping in on my day off .
- I enjoy tweeting others and not getting a reply

Figure 1. The figure shows some examples of sarcastic tweets. The sentence snippets with incongruity are marked in color.

A contrast of the snippet “work for 2 hours” and the snippet “worth the 35 min drive” makes it a sarcastic tweet. Consequently, capturing the incongruity between sentence snippets is a good choice.

Deep learning-based methods have achieved significant improvements in many NLP tasks, such as machine translation, sentiment analysis, and question answering. Deep learning models are also used in sarcasm detection studies [6, 29, 12, 31]. [29] first proposed a self-attention based neural network to explicitly model the contrast and incongruity on word-level, which achieved the best performance at the time. After that, following the work of [29], [31] introduced the “co-attention” mechanism suggested by [20] in their model to capture the joint information between every word-to-word pair in the input sentence. Meanwhile, they used a bilinear pooling method based on Low-rank Bilinear Pooling (LRBP) [16] to reduce the dimension of the final input without losing discriminative information. Their model reported state-of-the-art results.

Both [29] and [31] only model the incongruity on word-level, which may help to recognize the sarcasm exits in incongruous words like {love, exam}. However, when it comes to the sentences with complex semantic information like the given examples, their methods would be inefficient in determining the incongruity caused by sentence snippets and lead to a low recall. Inspired by this, we propose to model the incongruity between sentence snippets as they contain more semantic information. Moreover, snippets are intuitively not equally contributed to identify sarcasm, the incongruous snippets should be given more attention as they are the indispensible parts to compose the sarcasm. Thus, we introduce a context vector to compute the weights of the snippets according to their importance, which is not used by previous works.

In this paper, our intention is to detect sarcasm by capturing the incongruity between sentence snippets. We propose a novel model based on self-attention mechanism of weighted snippets. It consists of an input module, a convolution module, an importance weighting module, and a self-attention module. As sentence snippets contain more semantic information than words, we apply the convolution

[1] Institute of Information Engineering, Chinese Academy of Sciences, Bejing, China. Emails: {panhongliang, linzheng, fupeng, wangweiping}@iie.ac.cn. * Corresponding Author.
[2] School of Cyber Security, University of Chinese Academy of Sciences, Bejing, China.

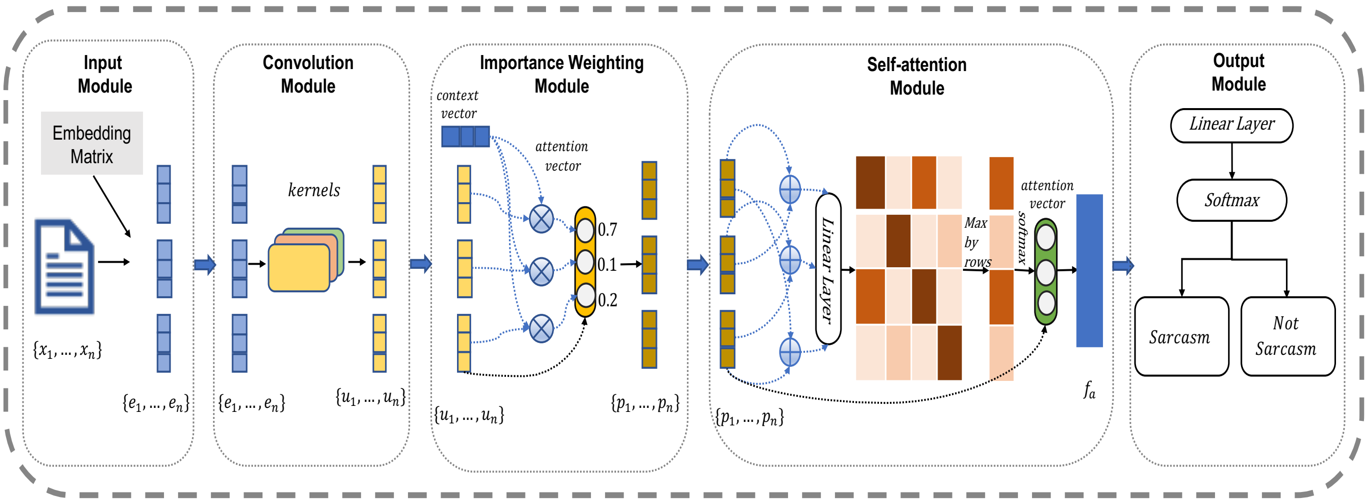

Figure 2. Overview of the model architecture. The leftmost is the input module which accepts the input words and converts them into low dimension representations. The middle part illustrates three modules, including Convolution Module to obtain sentence snippet representations, Importance Weighting Module to weight the snippets and Self-attention Module to capture the incongruity between snippets. On the right part, the Output module classifies the input into sarcasm or non-sarcasm by using a softmax layer.

module to acquire the snippet representations. Besides, we also employ an importance weighting module in our work to highlight the critical snippets for boosting the performance. Unlike [29] and [31], we did not use any RNNs to obtain the sentence compositional information and achieve promising performance.

The main contributions of this work are as follows:

- We propose a model based on capturing the incongruity of sentence snippets, which aims to address the problem that existing models are inefficient in identifying the sarcasm caused by sentence snippet incongruity.
- We introduce a context vector to calculate the importance of the snippets to improve the performance as not all snippets are equally important for sarcasm detection.
- Our model achieves state-of-the-art results on four benchmark datasets, including two short text Twitter datasets and two long text Internet Argument Corpus (IAC) datasets. Besides, the visualization part shows that our model strongly complies with human intuitions.

2 Method

2.1 Task Definition

Sarcasm detection aims to identify if there exists sarcastic meaning in a given scenario. Formally, given a text containing l words $X = \{x_1, x_2, \ldots, x_l\}$, where x_i represents a single word in the sequence. Our model is supposed to classify the given text into sarcasm or non-sarcasm categories correctly.

2.2 Framework

Figure 2 gives an overview of our model which is composed of five modules, including an input module, a convolution module, an importance weighting module, a self-attention module and an output module. The input words are firstly converted into low dimension representations via an encoding layer. Then, we use the convolution module to acquire sentence snippet representations based on the encoded words. After that, in the importance weighting module, we use a context vector to compute the weights of snippets, the critical snippets will be given high weights and the trivial snippets will be given low weights. Finally, the self-attention module is applied to the weighted snippets and the output module produces an output for classification.

2.3 Input Module

On the left of our model SAWS, the input text X is a sequence of words with $X \in \mathbb{R}^{l*1}$, where l is the number of words in the sequence. To resolve different lengths of input, we transform $X \in \mathbb{R}^{l*1}$ to $X' \in \mathbb{R}^{n*1}$ where n is a pre-defined hyper-parameter. Input sequences less than n tokens will be padded to n and input sequences more than n tokens will be truncated to n. In the input encoding layer, each word is converted into a low-dimensional vector representation (word embedding) by using a weighted matrix. We employ the publicly available GloVe vector, which contains 400,000 most frequent words [26]. Words not present in the set of pre-trained words are initialized randomly. As such, the output $E \in \mathbb{R}^{n*e}$ of this layer is a sequence of word embeddings, where e is the embedding size.

2.4 Convolution Module

Convolutional neural network (CNN) was firstly exploited by [17] in text classification tasks and it achieved outstanding perfromance. Convolution layers can be applied to grasp contextual local features by implementing convolution operation between the convolution kernels and the words in a sequence. As we want to acquire the representations of sentence snippets (usually consists of several consecutive words), a convolutional layer seems to be a good choice to encode local snippet information. A convolution filter $k \in \mathbb{R}^{m*e}$ has the same dimension e as the input matrix, which is applied to a window of m consecutive words of the input matrix $E \in \mathbb{R}^{n*e}$, performing element-wise product between selected windows of E and filter k to obtain a vector $c \in \mathbb{R}^{n-m+1}$. c is a vector consisting of $\{ c_1, c_2, c_3, \ldots, c_{n-m+1}\}$. Each c_i is calculated as the following formula:

$$c_i = \sum E_{i:i+m-1,e} \otimes k_{0:m,e} \tag{1}$$

where $\otimes$ means element-wise product. We perform the same operation using e such filters to acquire the snippet representation $U \in \mathbb{R}^{(n-m+1)*e}$. $U_i \in \mathbb{R}^e$ represents ith snippet in the original input sequence.

2.5 Importance Weighting Module

Based on the fact that sentence snippets are not equally contribute to identify sarcasm, we introduce the importance weighting module to weight the snippets during training and testing. In this module, we randomly initialize a context vector $v \in \mathbb{R}^{(n-m+1)}$ at the beginning. Then we use v to compute the attention score a_i of each snippet U_i. Firstly, we multiply snippet representation $U_i \in \mathbb{R}^e$ with $W \in \mathbb{R}^{e*(n-m+1)}$ and add to $b \in \mathbb{R}^{n-m+1}$, which is fed into $tanh$ layer to get u_i. Then, we calculate the similarity of u_i and the context vector v to measure the importance of the snippet i. After that, a softmax function is applied to normalize the weights and get a weight distribution vector a. a is calculated as following equations. To be specific,

$$u_i = \tanh(W^T U_i + b) \tag{2}$$

$$a_i = \frac{\exp(u_i v)}{\sum_i \exp(u_i v)} \tag{3}$$

$$a = (a_1,\ a_2,\ a_3 \dots a_{n-m+1}) \tag{4}$$

where $a \in \mathbb{R}^{n-m+1}$. Finally, we associate the snippets with their corresponding attention values. In detail, we multiply these two to get the weighted snippet representation $P \in \mathbb{R}^{(n-m+1)*e}$ and send it to the next module. Our importance weighting module is inspired by [32]'s work. v can be considered a high level representation of a fixed query "how informative is this snippet" over all snippets. The difference is that they use the vector v to compute attention weights on word level and sentence level for document classification, but we use v to compute attention weights across snippets. Similar to [32]'s work, v is jointly learned during the training process. This procedure is depicted in Figure 3.

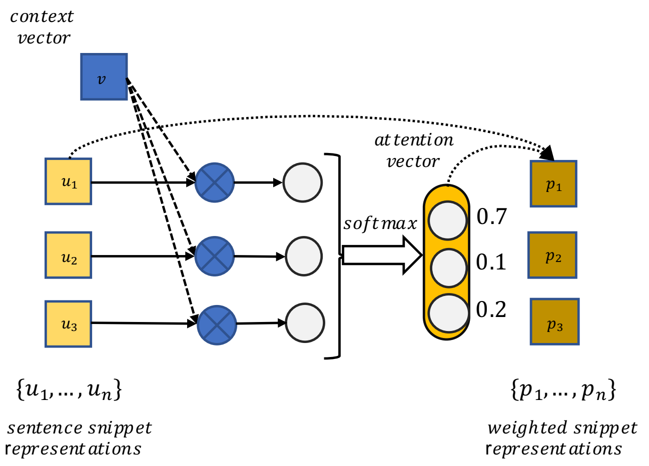

Figure 3. The illustration of how Importance Weighting module works. Critical snippets are granted with a high attention value in this module.

2.6 Self-attention Module

In this part, we use self-attention mechanism to model the contrast and incongruity between the weighted snippets P. Self-attention was firstly proposed by [3] to compute the internal representation of a single sequence relating different positions. Incongruity can be treated as an internal characteristic of sarcastic text. Accordingly, performing self-attention on the weighted phrases yields an output containing the incongruity information. Particularly, the incongruous phrases will give a high attention value to each other in order to reduce the training loss. In this module, we first compute the affinity score s between weighted snippets and obtain a self-attention matrix S. The affinity score $s_{i,j}$ between sentence snippet P_i and sentence snippet P_j is calculated as follows:

$$s_{i,j} = W([P_i; P_j]) + b \tag{5}$$

where s_{ij} is a scalar in the self-attention matrix S, representing the affinity score between snippet pair (P_i, P_j). P_i and P_j are the representations of weighted snippet i and j. $[;]$ means the concatenation operation of vectors. The self-attention matrix S consists of the affinity score of every sentence snippet pairs and shows as:

$$S = \begin{bmatrix} s_{1,1} & \cdots & s_{1,n-m+1} \\ \vdots & \ddots & \vdots \\ s_{n-m+1,1} & \cdots & s_{n-m+1,n-m+1} \end{bmatrix} \tag{6}$$

Then, we compute an attention vector $a \in \mathbb{R}^{n-m+1}$ by applying a row-wise max-pooling operation on the self-attention matrix S. [29] argues that words that might contribute to the contrastive theories of sarcasm should be highlighted (usually accompany with a high attention value). Thus, a more discriminative pooling operator like max-pooling is desirable in our case. Note that it is meaningless to calculate the interactive information between a snippet and itself marked as $s_{i,j}$, where $i = j$. Consequently, the attention value of a sentence snippet and itself has been masked to avoid influencing the final results. The attention vector a is computed as follows:

$$a_i = \max(s_{i,1},\ s_{i,2},\ s_{i,3} \dots s_{i,n-m+1}\ ) \tag{7}$$

$$a = Softmax\,(a_1,\ a_2,\ a_3 \dots a_{n-m+1}) \tag{8}$$

where $a \in \mathbb{R}^{n-m+1}$, After having the attention vector a, it is employed to the weighted snippet representations P as:

$$f_a = \sum_{i=1}^{n-m+1} P_i a_i \tag{9}$$

where $f_a \in \mathbb{R}^e$ is the self-attentive representation of the weighted snippets, which contains the incongruity information and will be used to predict.

2.7 Output Module

The input of the output module is $f_a \in \mathbb{R}^e$, which is the self-attentive representation. The prediction layer consists of a linear layer and a Softmax classification layer. The linear layer aims to reduce the dimensionality of f_a. Softmax layer is used to classify the output into two categories Sarcasm or Non-sarcasm.

$$\hat{y} = Softmax\,(W f_a + b) \tag{10}$$

where $W \in \mathbb{R}^{e*2}, b \in \mathbb{R}^2$ are the learnable parameters and training along with the model. $\hat{y} \in \mathbb{R}^2$ is the classification result of our model.

2.8 Training Objectives

Cross-entropy loss function is used in our work for optimizing the model.

$$J = -\sum_{i=1}^{N} [y_i log\hat{y_i} + (1 - y_i)\log(1 - \hat{y_i})] + \lambda R \tag{11}$$

where J is the cost function. $\hat{y_i}$ is the prediction result of our model for sample i and y_i is the true label for sample i. N is the size of training data. R is the standard L2 regularization and λ is the weight of R.

3 Experiment

In this section, we describe the datasets, experiment setups and experiment results. Besides, we also give out an ablation study and a comprehensive analysis of our model.

3.1 Datasets

We evaluate our model on four benchmark datasets, including two short text Twitter datasets[1] collected by Mishra et al. [24] and Ghosh et al. [7] as well as two long text IAC datasets[2] collected by [30] and annotated by [21]. Most previous works used the Twitter dataset collected by Riloff et al. [28]. However, we retrieved the tweets using the Twitter API by the provided tweet IDs, we found that only around 1/3 of the original dataset are available. Consequently, we test our model on a recent collected Twitter dataset provide by Mishra et al. [24]. Detailed statistics are summarized in Table 1.

Twitter is a microblogging platform on which users post and interact with messages known as “tweets”. It allows users to update their status within a limit of characters. In our work, we use two Tweets datasets for sarcasm detection. More specifically, we use the dataset collected by Mishra et al. [24] and Ghosh et al. [7]. Mishra dataset was collected by manually annotated and Ghosh dataset was automatically annotated by the hashtags in tweets, such as “Sarcasm”, “sarcastic”, “irony”. In addition, they also devised a feedback-based system that can contact the tweet authors to validate the correctness of the sarcasm labels.

Internet Argument Corpus (IAC) is mainly focus on long text. It was originally collected from an online debate forum to study political debates and annotated by [21] for sarcasm detection. We use two versions of it, namely IAC-V1 and IAC-V2. [29] and [31] also employed the same two datasets in their works.

Table 1. Datasets description

Dataset	Train	Test	Total	Avg length
Tweets(Mishra)	894	100	994	25.76
Tweets(Ghosh)	48635	3944	52579	17.90
IAC-V1	1670	186	1856	68.32
IAC-V2	4179	465	4644	55.82

[1] We acquire the same Ghosh Twitter datasets as [31]. We acquire the Mishra Twitter datasets from http://www.cfilt.iitb.ac.in/cognitive-nlp/

[2] We download IAC datasets from https://nlds.soe.ucsc.edu/sarcasm1 and https://nlds.soe.ucsc.edu/sarcasm2 respectively.

3.2 Experimental Settings

Our model is implemented using PyTorch[3] and running on a NVIDIA Tesla M40 GPU. We process the data in the same way as [29]. We use $<UNK>$ to replace the words that appear only once and remove all samples less than five tokens and duplicate instances on the datasets. We also remove the URLs on the dataset. We employ GloVe[4] [26] for our word embeddings with a fixed embedding size 100 and we fine tune the embeddings during training. As for the hyper-parameter n, which is the maximum length of the sequences. We set it to 40 for Tweets datasets because 91.1% data have a length less than 40 tokens on Tweets (Ghosh) dataset. The number is 85.5% on Tweet (Mishra) dataset. For IAC datasets, we set n to 60 as IAC datasets mainly contain long texts. The sentence snippet length m is set to 3 for Twitter datasets and 5 for IAC datasets, which is empirically motivated. We use RMSProp optimizer [13] to optimize model parameters with the learning rate equals to 10^{-4}. The L2 regularization is set to 10^{-3} for Twitter datasets and 10^{-2} for IAC datasets. We use early stopping in our experiment if the loss does not decrease on the validation set for 20 epochs, the model will stop training. Our code is publicly available[5].

3.3 Baseline Models

- NBOW: The Neural Bag-of-words model performs classification with an average of the input word embeddings followed by logistic regression. It is an effective model even it has a simple architecture.
- CNN-LSTM-DNN: The convolutional LSTM + DNN model was introduced by [6]. They used two convolutional layers and two LSTM layers to extract features from input word embeddings. Followed by a deep neural network for classification. It is a deep learning-based model but without applying any attention mechanisms.
- SIARN and MIARN: SIARN and MIARN are the models first used self-attention for sarcasm detection [29]. It overcomes the weakness of traditional sequential models such as recurrent neural networks, which cannot model the interaction between word pairs in a sentence. SIARN uses single dimension of words to calculate their interactions while MIARN used multi-dimension.
- SMSD and SMSD-BiLSTM: These two models were proposed by [31]. Compared with [29]'s work, [31] introduced a weight matrix between word pairs to improve the flexibility of capturing joint information between word pairs. For the model SMSD-BiLSTM, besides SMSD, it used an additional bi-directional LSTM encoder to cultivate sentence's compositional information instead of a common LSTM encoder in [29]'s work. SMSA and SMAD-BiLSTM are state-of-the-art models on most of the benchmark datasets.

3.4 Experimental Results

We compare our model with above baseline models on some standard evaluation metrics, including precision, recall, F1 score, and accuracy[6]. Precision describes how effective the model is in applying a label for a given category (few false positives). Recall describes

[3] https://pytorch.org/

[4] http://nlp.stanford.edu/projects/glove/

[5] https://github.com/marvel2120/SAWS

[6] We calculate the precision, recall, F1 score, and accuracy by using sklearn.metrics. https://scikit-learn.org/stable/modules/classes.html

Table 2. Experiment results in two Twitter datasets. Best results are in bold.

	Tweets(Ghosh)				Tweets(Mishra)			
Model	Precision	Recall	F1	Acc	Precision	Recall	F1	Acc
NBOW	*74.55*	*73.93*	*73.94*	*74.21*	*72.00*	*68.13*	*69.04*	*74.00*
CNN-LSTM-DNN (Ghosh and Veale, 2017)	73.20	71.70	72.50	-	-	-	-	-
SIARN (Tay et al. 2018)	*81.26*	*81.01*	*81.07*	*81.16*	*82.14*	*79.67*	*80.60*	*83.00*
MIARN (Tay et al. 2018)	*80.90*	*80.93*	*80.92*	*80.83*	*80.73*	*78.90*	*79.64*	*82.00*
SMSD (Xiong et al. 2019)	*80.25*	*80.23*	*80.24*	*80.27*	*78.21*	*80.00*	*78.78*	*80.00*
SMSD-BiLSTM (Xiong et al. 2019)	*81.02*	*81.07*	*81.02*	*81.03*	*79.08*	*79.45*	*79.26*	*81.00*
SAWS (this paper)	**83.18**	**83.12**	**83.14**	**83.21**	**86.31**	**84.73**	**85.41**	**87.00**

how effective the model is in finding all the relevant examples of a category (few false negatives). F1 score indicates a trade-off between the precision and the recall. Table 2 and 3 show the results of our model and other baseline models on Twitter dataset and IAC dataset respectively. We observe that our model SAWS achieves state-of-the-art performance in both Twitter datasets. It improves the F1 score by around 2.1% on the Ghosh dataset and around 4.8% on the Mishra dataset. It is worth noting that, some of the values that we are taking in consideration have been extracted from existing works. Others are provided by our reimplementation of their works according to their papers and they are in italic in the tables. Our model performs better on the Mishra dataset than the Ghosh dataset. Mishra dataset contains less special tokens, which makes it a clean dataset with better qualification. In contrast, the Ghosh dataset has many undefined symbols and emoticons, which are the noises on the dataset. Consequently, the high quality of Mishra dataset might contribute to the outstanding performance of our model.

As for IAC datasets, our model also achieves the best results on both IAC-V1 dataset and IAC-V2 dataset. Our model obtains the highest recall score on both datasets, which means that our model is powerful to identify the potential sarcastic texts than other models. We owe this to the involvement of modeling sentence snippet incongriuty. Our model is capable of capturing both word-level and snippet-level incongruity. Thus, some sarcastic texts which based on sentence snippets incongruity can be detected by our model and results in an improved recall. Additionally, the experimental results demonstrate that our model performs better on the Twitter datasets compared with the IAC datasets. We believe that long text datasets like IAC compries more complex semantic information, which makes it difficult to be identified without involving extra knowledge, such as “facial gesture”, “intonation” and some “facts”.

Compare with other baseline models. We notice that the NBOW is still an effective model though it has a quite simple architecture. The models proposed by [29] and [31] outperform both the NBOW model and the CNN-LSTM-DNN model benefit by introducing the self-attention mechanism and sequential models. Similarly, our model also involves self-attention mechanism and it is proved to be effective. However, we perform self-attention on the weighted sentence snippets to better capture the incongruity rather than on the words like [29] and [31]. In addition, the results illustrate that our method improves the model performance even without using separate RNNs to accquire sequential information.

3.5 Ablation Study

In order to see if the snippet-level self-attention module and the importance weigthing module improve the model performance, we conduct a series of ablative experiments. We first remove the convolution module and get the model SAWS(w\o conv). SAWS(w\o conv) accepts words rather than sentence snippets because there is no convolution operation on the words. Then, we eliminate the importance weigthing module and get the model SAWS(w\o weighting). Model SAWS(w\o weighting) uses the snippets directly obtained from the convolution module without weighting.

Table 4 gives the results of the ablative experiments. It shows that our model SAWS gets the best results when involving both snippet-level self-attention module and importance weigthing module. SAWS(w\o conv) performs the worst in the experiment, which shows the importance of the convolution module and proves that capturing the snippet incongruity is meaningful and effective. The results also comply with the intuitive feelings that a snippet contains more incongruous information than a single word. SAWS(w\o weighting) also performs worse than SAWS, which demonstrates that snippets are not equally crucial in identifying sarcasm. Giving the critical snippets with high weights contributes to the performance. Consequently, the convolution module and the importance weigthing module play an indispendable role in our model.

3.6 Model Analysis

In this section, we give out a comprehensive analysis of our model. We first measure the effect of the sentence snippet length m. Then, a model visualization is given for importance weighting module and self-attention module to prove their effectiveness.

- **The effect of sentence snippet length**
  We measure the performance of our model, along with a range of sentence snippet length m form 1 to 6. We can see in Figure 4, all the metrics including Precision, Recall, F1 scores and Accuracy keep increasing until reach a peak point when m is equal to 3. The performance begins to decrease when m continues to grow. Thus, the value of m is vital to model performance. A small m means that our model only focuses on short sentence snippets, which

Table 3. Experiment results in two IAC datasets. Best results are in bold.

	IAC-V1				IAC-V2			
Model	Precision	Recall	F1	Acc	Precision	Recall	F1	Acc
NBOW	57.17	57.03	57.00	57.51	66.01	66.03	66.02	66.09
CNN-LSTM-DNN (Ghosh and Veale, 2017)	55.50	54.60	53.31	55.96	64.31	64.33	64.31	64.38
SIARN (Tay et al. 2018)	63.94	63.45	62.52	62.69	72.17	71.81	71.85	72.10
MIARN (Tay et al. 2018)	63.88	63.71	63.18	63.21	72.92	72.93	72.75	72.75
SMSD (Xiong et al. 2019)	*63.04*	*63.06*	*62.90*	*62.90*	*72.08*	*72.12*	*72.04*	*72.04*
SMSD-BiLSTM (Xiong et al. 2019)	*62.79*	*62.53*	*62.51*	*62.90*	*71.56*	*71.49*	*71.52*	*71.61*
SAWS (this paper)	**66.22**	**65.65**	**65.60**	**66.13**	**73.52**	**73.40**	**73.43**	**73.55**

Table 4. Ablation experiment results.

Model	Precision	Recall	F1	Acc
SAWS(w\o conv)	81.50	81.46	81.47	81.51
SAWS(w\o weighting)	82.90	82.79	82.83	82.89
SAWS	**83.18**	**83.12**	**83.14**	**83.21**

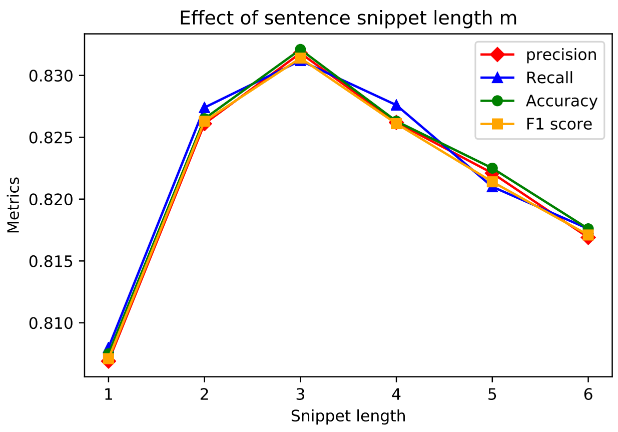

Figure 4. The performance curves with a variety of the sentence snippet length m from 1 to 6. The metrics include precision, recall, F1 score, and accuracy, which are marked with different colors.

might lose necessary information for sarcasm detection. Thus, resulting in poor performance. In contrast, a large m might involve some redundant messages. The redundant messages also would impede the model performance.

- **Model visualization**
  In this section, we visualize how importance weighting module and self-attention module act in our model. We demonstrate several sarcastic cases collected from the testing data. The first two are correctly classified by our model while the last is not.

  – "so excited for my family hike at 9 freaking o'clock in the morning ! I love not sleeping in on my day off ."

  – "yayy I have my english class from 9:00 to 12:00 today ! this is going to be fun !"

  – "love this store ! there all goodie box is awesome !"

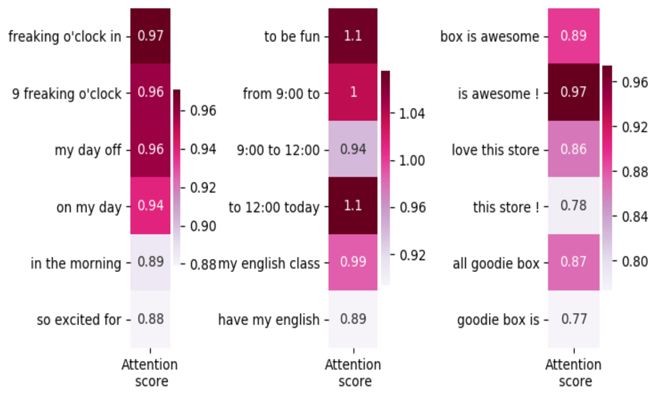

Figure 5. The demonstration of the top six critical snippets selected by our importance weigthing module under the given examples. The numbers in the graph are scaled attention score, which shows the importance of snippets.

Figure 5 gives out the top six important snippets selected by our importance weighting module. In the second example, our model takes the snippets "to be fun", "my english class", "from 9:00 to" and "to 12:00 today" as critical snippets. Intuitively, these snippets are indispensable for a human being to identify the incongruity within a sarcastic text. As a result, our importance weighting module is active in selecting the critical snippets and giving a high score. However, note that a critical snippet might mean that it appears frequently on the sarcasm dataset, but this not guarantee it must be exactly the same as a human segmented incongruous snippet for a specific instance. However, the snippet weight indicates how informative the snippet is and it can still give insights into considering the importance of snippets for sarcasm prediction. For instance, our model takes "is awesome !" as the most critical snippet, which might because "is awesome !" is very common in sarcastic expression.

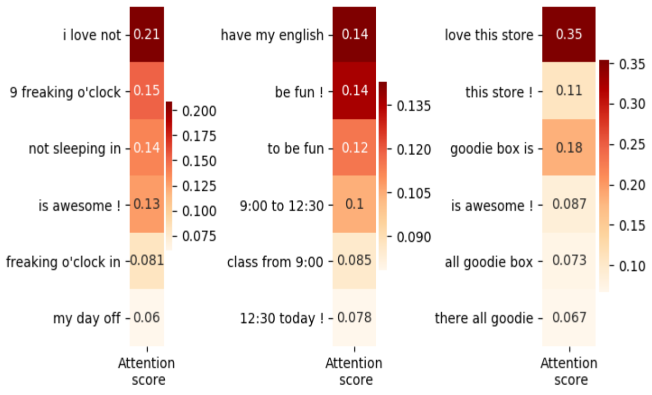

Figure 6. The visualization of snippet self-attention of the given examples. The darkness of the background varies according to the value of the corresponding attention weight.

Figure 6 visualizes how our self-attention module works. Figure 6 shows that our model is highly effective in attending the incongruous snippets, which is a strong indicator to detect sarcasm. Take the same example from last paragraph, we notice that the snippets "have my english" and "be fun !" have been given the highest attention weights. The same pattern can be found in the first example. Thus, our self-attention module is powerful in capturing the snippet incongruity within a sentence, which makes it effective to identify sarcasm. However, for the third instance, our model fails to find such incongruity patterns and leads to a false classification. This kind of sarcasm is even hard for human beings to recognize without extra information, such as the speaker's facial gesture or intonation.

4 Related Work

Sarcasm is a linguistic phenomenon which has been widely studied by linguistic scholars [1, 4, 11, 10]. Automatic sarcasm detection has gained the NLP researchers' interest partially due to the rising of social media and sentiment analysis [27]. Most existing works concentrate on text-only sarcasm detection. Nonetheless, some other valuable works also exploit user representations [18], contextual information [12], congnitive features [24, 23] and multimodal features [2] to determine sarcasm. While our work mainly focuses on text-only sarcasm detection, we skip the detail of their works for brevity. The existed text-based sarcasm detection works can be divided into three categories: rule-based approaches, feature-based machine learning approaches, and deep learning-based approaches [14].,

Sarcasm detection was originally solved by using rule-based approaches. Rule-based approaches aim to identify sarcasm with fixed patterns. Such indicative patterns are extracted by using linguistic rules. [28] observed that sarcastic Twitter message often contained a common pattern that is a positive sentiment followed by a negative situation. They proposed a bootstrapping algorithm which iteratively expands the positive and negative sentiment phrase sets. They found that identifying sarcasm using the phrases learned by their bootstrapping algorithm resulted in an improved recall. Hashtags in Twitter messages, such as *#sarcasm, #sarcastic, #not, #not true, #greatstart* were also considered in sarcasm detection. The hashtags were labeled by users to express their feelings. [22] developed a hashtag tokenizer, such that sentiment and sarcasm within hashtags can be detected. They also compiled a number of rules to improve the accuracy of sentiment classification when sarcasm is known to be present.

However, rule-based approaches only rely on fixed patterns, which makes it challenging to capture complex sarcastic texts. In order to improve the performance, the scholars began to enrich the feature set and use machine learning approaches. [5] used both structural features, such as punctuation mark frequency, tweet length, uppercase character amount and affective features to detect sarcasm. Affective features involve sentiment-related features and emotional categories. [8] treated the sarcasm detection task as a type of word sense disambiguation problem. They used an SVM classifier with a modified kernel and word embeddings, which obtained a 7-10% improvement compared with the baseline. [15] developed a system that harnesses context incongruity to detect sarcasm. Their classifier incorporated both explicit incongruity features and implicit incongruity features. Their model outperformed two past works with 10%-20% F-score improvement.

Though feature-based algorithms achieved promising performance in sarcasm detection, the construction of discrete features is a time-consuming job. Researchers have recently considered deep learning-based methods because it is capable of extracting features automatically. [27] used pre-trained Convolutional Neural Networks to extract sentiment, emotion, and personality features for sarcasm detection. A self-attention based neural model was firstly proposed by [29] for sarcasm detection. Similarly, [31] proposed a self-matching network, in which the joint information of each word-to-word pair was calculated to capture the context incongruity. Inspired by their works, we propose a novel model SAWS. The major difference between their models and SAWS is that they consider the incongruity on word-level, but we consider on snippet-level because we believe that snippets contain more discernible sentiment than words. Besides, we also introduce an importance weighting mechanism to give high weights to critical snippets and low weights to trivial snippets to enhance the performance.

5 Conclusion

In this paper, we present a novel model by introducing snippet-level self-attention to model the incongruity between sentence snippets, which addresses the issue that existing models cannot capture the snippet incongruity in sarcastic texts. Besides, human beings can recognize sarcasm by paying different level attention to different snippets. Accordingly, we introduce an importance weighting module in our model to determine critical snippets. Meanwhile, we also conduct a series of ablative experiments to verify the effectiveness of the modules proposed in this paper. Our model outperforms state-of-the-art models on four benchmark datasets and enjoys a good interpretation as it heavily complies with human intuitions.

6 Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61976207, No. 61906187). The authors would like to thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] John D Campbell and Albert N Katz, 'Are there necessary conditions for inducing a sense of sarcastic irony?', *Discourse Processes*, **49**(6), 459–480, (2012).

[2] Santiago Castro, Devamanyu Hazarika, Verónica Pérez-Rosas, Roger Zimmermann, Rada Mihalcea, and Soujanya Poria, 'Towards multimodal sarcasm detection (an _obviously_ perfect paper)', in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pp. 4619–4629, (2019).

[3] Jianpeng Cheng, Li Dong, and Mirella Lapata, 'Long short-term memory-networks for machine reading', in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pp. 551–561, (2016).

[4] Jodi Eisterhold, Salvatore Attardo, and Diana Boxer, 'Reactions to irony in discourse: evidence for the least disruption principle', *Journal of Pragmatics*, **38**(8), 1239–1256, (2006).

[5] Delia Irazú Hernández Farías, Viviana Patti, and Paolo Rosso, 'Irony detection in twitter: The role of affective content', *ACM Trans. Internet Techn.*, **16**(3), 19:1–19:24, (2016).

[6] Aniruddha Ghosh and Tony Veale, 'Fracking sarcasm using neural network', in *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity,Sentiment and Social Media Analysis, WASSA@NAACL-HLT 2016*, pp. 161–169, (2016).

[7] Aniruddha Ghosh and Tony Veale, 'Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal', in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, pp. 482–491, (2017).

[8] Debanjan Ghosh, Weiwei Guo, and Smaranda Muresan, 'Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words', in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pp. 1003–1012, (2015).

[9] Raymond W Gibbs Jr, Raymond W Gibbs, and Jr Gibbs, *The poetics of mind: Figurative thought, language, and understanding*, Cambridge University Press, 1994.

[10] Raymond W Gibbs Jr and Jennifer O'Brien, 'Psychological aspects of irony understanding', *Journal of pragmatics*, **16**(6), 523–530, (1991).

[11] Rachel Giora, 'On irony and negation', *Discourse processes*, **19**(2), 239–264, (1995).

[12] Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea, 'CASCADE: contextual sarcasm detection in online discussion forums', in *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018*, pp. 1837–1848, (2018).

[13] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky, 'Neural networks for machine learning lecture 6a overview of mini-batch gradient descent'.

[14] Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman, 'Automatic sarcasm detection: A survey', *ACM Comput. Surv.*, **50**(5), 73:1–73:22, (2017).

[15] Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya, 'Harnessing context incongruity for sarcasm detection', in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics ACL 2015*, pp. 757–762, (2015).

[16] Jin-Hwa Kim, Kyoung Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang, 'Hadamard product for low-rank bilinear pooling', in *5th International Conference on Learning Representations, ICLR 2017*, (2017).

[17] Yoon Kim, 'Convolutional neural networks for sentence classification', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1746–1751, (2014).

[18] Y. Alex Kolchinski and Christopher Potts, 'Representing social media users for sarcasm detection', in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1115–1121, (2018).

[19] Bing Liu, 'Sentiment analysis and subjectivity', in *Handbook of Natural Language Processing, Second Edition.*, (2010).

[20] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh, 'Hierarchical question-image co-attention for visual question answering', in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pp. 289–297, (2016).

[21] Stephanie Lukin and Marilyn Walker, 'Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue', in *Proceedings of the Workshop on Language Analysis in Social Media*, pp. 30–40, Atlanta, Georgia, (June 2013). Association for Computational Linguistics.

[22] Diana Maynard and Mark A. Greenwood, 'Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis', in *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pp. 4238–4243, (2014).

[23] Abhijit Mishra, Kuntal Dey, and Pushpak Bhattacharyya, 'Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network', in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 377–387, (2017).

[24] Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya, 'Predicting readers' sarcasm understandability by modeling gaze behavior', in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3747–3753, (2016).

[25] Bo Pang and Lillian Lee, 'Opinion mining and sentiment analysis', *Foundations and Trends in Information Retrieval*, **2**(1-2), 1–135, (2007).

[26] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, 'Glove: Global vectors for word representation', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 1532–1543, (2014).

[27] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij, 'A deeper look into sarcastic tweets using deep convolutional neural networks', in *COLING 2016, 26th International Conference on Computational Linguistics Proceedings of the Conference: Technical Papers, December*, pp. 1601–1612, (2016).

[28] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang, 'Sarcasm as contrast between a positive sentiment and negative situation', in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pp. 704–714, (2013).

[29] Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su, 'Reasoning with sarcasm by reading in-between', in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pp. 1010–1020, (2018).

[30] Marilyn A. Walker, Jean E. Fox Tree, Pranav Anand, Rob Abbott, and Joseph King, 'A corpus for research on deliberation and debate', in *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pp. 812–817, (2012).

[31] Tao Xiong, Peiran Zhang, Hongbo Zhu, and Yihui Yang, 'Sarcasm detection with self-matching networks and low-rank bilinear pooling', in *The World Wide Web Conference, WWW 2019*, pp. 2115–2124, (2019).

[32] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy, 'Hierarchical attention networks for document classification', in *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, (2016).