Augmenting Affective Dependency Graph via Iterative Incongruity Graph Learning for Sarcasm Detection

Xiaobao Wang[1], Yiqi Dong[2], Di Jin[1], Yawen Li[3]*, Longbiao Wang[1,4], Jianwu Dang[5,1]

[1]Tianjin Key Laboratory of Cognitive Computing and Application, College of Intelligence and Computing, Tianjin University, Tianjin, China,
[2]School of New Media and Communication, Tianjin University, Tianjin, China,
[3] School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing, China
[4] Huiyan Technology (Tianjin) Co., Ltd, Tianjin, China,
[5] Peng Cheng Laboratory, Shenzhen, China,
{wangxiaobao, dongyiqi, jindi}@tju.edu.cn, warmly0716@bupt.edu.cn, longbiao_wang@tju.edu.cn, jdang@jaist.ac.jp

Abstract

Recently, progress has been made towards improving automatic sarcasm detection in computer science. Among existing models, manually constructing static graphs for texts and then using graph neural networks (GNNs) is one of the most effective approaches for drawing long-range incongruity patterns. However, the manually constructed graph structure might be prone to errors (e.g., noisy or incomplete) and not optimal for the sarcasm detection task. Errors produced during the graph construction step cannot be remedied and may accrue to the following stages, resulting in poor performance. To surmount the above limitations, we explore a novel Iterative Augmenting Affective Graph and Dependency Graph (IAAD) framework to jointly and iteratively learn the incongruity graph structure. IAAD can alternatively update the incongruity graph structure and node representation until the learning graph structure is optimal for the metrics of sarcasm detection. More concretely, we begin with deriving an affective and a dependency graph for each instance, then an iterative incongruity graph learning module is employed to augment affective and dependency graphs for obtaining the optimal inconsistent semantic graph with the goal of optimizing the graph for the sarcasm detection task. Extensive experiments on three datasets demonstrate that the proposed model outperforms state-of-the-art baselines for sarcasm detection with significant margins.

Introduction

With the rapid growth of social media, Internet users employ a variety of rhetorical strategies to convey their thoughts and emotions, such as sarcastic expressions. Sarcasm is the form of language to convey implicit information/intention that has the opposite connotation of what is said or written (Joshi, Bhattacharyya, and Carman 2017). As illustrated in the upper examples of Figure 1, both sentences contain the decisive sentiment word "love". In contrast, the word "ignore" leads to a contradiction expression in the sarcastic example. That is, there are some incongruity expressions in a sarcastic context. Such intentional ambiguity has a significant impact on the performance of sentiment analysis (Bakshi et al. 2016). Hence sarcasm detection has gained increasing attention over the past decade, it provides a more realistic picture of social media users' intentions.

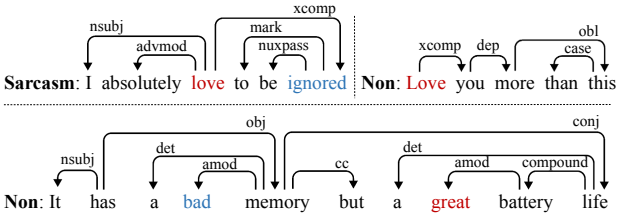

Figure 1: Three examples of sarcasm and non-sarcasm expression. The words with incongruity are colored. Dependency relationships derived from the parsing tree are represented as directed edges.

Early works attempt to extract incongruity expressions in sarcasm detection by exploring a collection of positive verbs and negative contexts or employing lexical features (Bamman and Smith 2015). Modern methods employing deep neural networks have significantly improved sarcasm detection by capturing incongruity (Babanejad et al. 2020; Hazarika et al. 2018). Deep neural networks augmented by attention mechanisms are widely utilized in recent methods for explicitly exploring the contrast and incongruity on word-level or snippet-level (Xiong et al. 2019; Pan et al. 2020), which yielded promising results at the time.

Although these attention-based models are adept at capturing semantic incongruity patterns, they inevitably result in misjudgment since the attention module may highlight extraneous words due to the absence of syntactic context. Considering the blow example in Figure 1, the attention module may still assign a large weight to the terms *"bad"* and *"great"*, which are similar to the upper sarcastic example and have a strong contrast. If simply taking such semantic inconsistent information into account, we might wrongly categorize it as a sarcastic expression.

Inspired by existing graph-based models (Tang et al. 2020), (Lou et al. 2021) manually build an affective graph

*Corresponding author

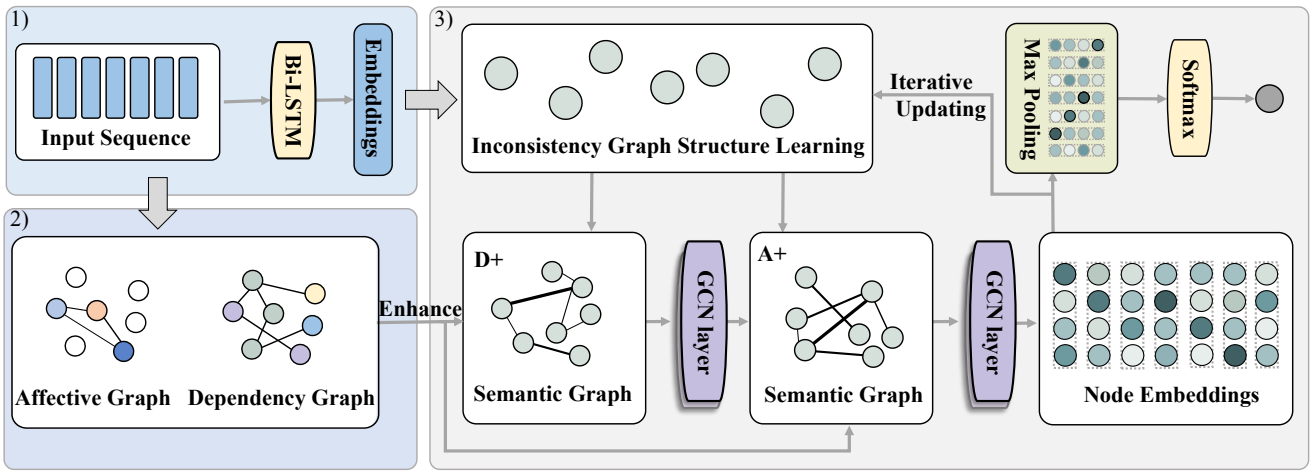

Figure 2: Architecture of our proposed model, which mainly contains three modules: 1) the sequence encoding module, 2) the affective and dependency graphs construction module, and 3) the iterative incongruity graph learning module.

and a dependency graph for each instance to capture incongruity sentiment expression and syntactic structure, respectively. Then, they encode the information conveyed by the two graphs using GCN (Welling and Kipf 2017; Jin et al. 2021b; Wu et al. 2022) for improved modeling of distant or unconnected words in the affective and dependency graphs, which achieved excellent performance at the time.

There is an apparent disadvantage to the direct application of static affective and dependency graphs. Due to imperfect parsing performance and the arbitrariness of the input sentences, as well as the fact that the same word has different meanings and sentiments varying with times, contexts, or domains, the affective and dependency graphs inevitably encompass noisy information.

In this paper, we propose a novel Iterative Augmenting Affective Graph and Dependency Graph (IAAD) framework for jointly and iteratively learning the optimal incongruity graph structure toward sarcasm detection. The proposed model (IAAD) first extracts contextual information based on Bi-LSTM (Ghosh and Veale 2016) and then constructs an affective graph and a dependency graph for each instance based on external affective knowledge and the dependency tree of the sentence, so as to exploit the affective and syntactic information of incongruity expressions. Based on it, the iterative incongruity graph learning module is employed to augment affective and dependency graphs for obtaining the optimal inconsistent semantic graph with the goal of optimizing the graph for the sarcasm detection task.

The main contributions of our work can be summarized as follows:

- To the best of our knowledge, the proposed IAAD is the first to introduce iterative incongruity graph structure learning to augment affective dependency graphs for sarcasm detection. When the learned incongruity graph structure approaches the optimal graph (for prediction), IAAD dynamically terminates.
- The optimal incongruity graph is able to best filter out the noise information introduced through imperfect parsing and sentiment mining performance. Furthermore, it can explicitly show the inconsistency between words in a sentence in the form of a graph.
- Experimental results on several benchmark datasets show that our suggested strategy achieves state-of-the-art sarcasm detection performance.

Method

This section begins by defining the sarcasm detection task. Then, we describe our proposed IAAD framework and present our training goals. Figure 2 gives an overview architecture of our model.

Task Definition

Sarcasm detection is intended to identify whether an utterance has a sarcastic meaning. This work focuses on sarcasm recognition in social texts. Formally, given a text containing n words $L = \{x_1, x_2, ..., x_n\}$, our model is supposed to correctly predict whether the text is satirical or non-satirical.

Model Framework

In general, our model can be divided into three main components: 1) *the sequence encoding module*, which utilizes a bidirectional LSTM (Ghosh and Veale 2016) to obtain text embeddings with context information. 2) *the affective and dependency graph construction module*, which derives an affective graph and a dependency graph for each input sequence to capture the sentimental inharmony and syntactic dependencies respectively. 3) *the iterative incongruity graph learning module*, which augments affective and dependency graphs to obtain the inconsistent semantic graph by iteratively applying GCNs.

1) Sequence Encoding: We apply a Bi-LSTM to obtain vector representations of the input text, which has a long-term memory and could capture the context information

well. We begin by converting each word x_i in a sequence into a fixed k-dimensional vector $\boldsymbol{e}_i$ by using a lookup table. Specifically, we employ the pre-trained GloVe vector (Pennington, Socher, and Manning 2014), which is publicly available and contains the most frequent words. Then we feed the word embeddings $\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_n]$ into a Bi-LSTM to encode the input word sequence into vectors incorporating positional information:

$$\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n\}, \quad (1)$$

$$\boldsymbol{h}_t = \left[\overrightarrow{LSTM}\left(\boldsymbol{e}_t\right), \overleftarrow{LSTM}\left(\boldsymbol{e}_t\right)\right], t \in (1, 2, ..., n), \quad (2)$$

where $\boldsymbol{e}_t \in R^k$ is the initial vector of each word and $\boldsymbol{h}_t \in R^{d_h}$ denotes the hidden output at time step t. d_h is the output dimension of the Bi-LSTM encoder.

2) Affective and Dependency Graphs Construction: As sarcasm is always coupled with sentiment inconformity features, we build an affective graph to acquire affective inconsistent information in the given text. Inspired by (Lou et al. 2021), for each input sequence $L = \{x_1, x_2, ..., x_n\}$, we construct an affective graph and obtain an adjacency matrix $\boldsymbol{A} \in R^{n \times n}$, based on the affective scores of words retrieved from an external sentiment dictionary SenticNet (Cambria et al. 2020). Each element $a_{i,j}$ in $\boldsymbol{A}$ is calculated as follows:

$$a_{i,j} = \left|u\left(x_i\right) - u\left(x_j\right)\right|, \quad (3)$$

where $u\left(x_i\right) \in [-1, 1]$ is the affective score of word x_i extracted from SenticNet. If x_i or x_j is not in the dictionary, we set $a_{i,j} = 0$. $|\cdot|$ indicates absolute value calculation. In this way, the greater the sentiment reversal between two words, the larger the weight of the corresponding edge. Words with opposite sentiments can be given high attention.

In addition, the information in the affective graph is insufficient since it can only extract the sentimental inconsistency between discrete word pairs in sentences, while ignoring the syntactic structure information of the entire sequence, which is closely associated with the expressed sentiment and meaning. Therefore, a dependency graph based on the parsing tree is built to serve as supplementary information for the affective graph. Each element in the dependency adjacency matrix $\boldsymbol{D} \in R^{n \times n}$ is derived as follows:

$$d_{i,j} = \begin{cases} 1 & if \ \ \psi\left(x_i, x_j\right) \\ 0 & \text{else} \end{cases}, \quad (4)$$

where $\psi\left(x_i, x_j\right)$ reveals there is an edge between x_i and x_j in the dependency tree of the input sequence. In addition, a self-loop $d_{i,i} = 1$ is applied for each word, and we do not consider the directionality of edges in the graph to extend the dependency information so we set $d_{i,j} = d_{j,i}$.

3) Iterative Incongruity Graph Learning: In contrast to building static graphs from a given text according to certain rules as in previous studies, we explore a novel way to learn the inconsistent semantic graph structure for sarcasm detection dynamically and adaptively to augment affective and dependency graphs in order to filter out noisy information. The Iterative Graph Learning Module contains three parts: a) Inconsistency Graph Structure Learning, b) Augmenting Affective and Dependency Graphs, c) Iterative Learning:

a) Inconsistency Graph Structure Learning: As previous studies have found that there is obvious semantic disaccord between word pairs in most of the sarcastic utterances (Joshi, Sharma, and Bhattacharyya 2015), we consider the dynamic graph construction as a semantic inconsistency metric learning problem. Therefore, the most critical part of this module is how to design the semantic inconsistency metric. As stated in (Xiong et al. 2019), simply using the inner product operation to calculate joint information absorbs and dilutes too much information across words. For instance, in a pre-trained word embedding space, the inner product between the vectors for *"Loud"* and *"Silent"* is typically larger than the inner product between *"Loud"* and *"Noisy"*; however, the inner product between *"Loud"* and *"New"* is also likely to be significant, as these two words are less likely to have similar occurrence patterns. Furthermore, the inner product operation causes semantically similar words to have large values, while we desire words with potential semantic inconsistency to produce larger values, which are fed into the subsequent GNNs for iterative learning. In this work, we introduce a parameter matrix-guided semantic inconsistency metric to address this limitation.

Specifically, we take the output of the Sequence Encoding Module $\boldsymbol{H} \in R^{n \times d_h}$ as our current input, and introduce a full connection weight matrix $\boldsymbol{S}$ for each input sequence by computing attention scores to the word-level embeddings. For any word pair $(\boldsymbol{h}_i, \boldsymbol{h}_j)$, for $i, j \in (1, 2, ..., n)$, our attention score $s_{i,j}$ is calculated by introducing a parameter matrix which can be trained together. Two different types of activation functions are also added to make the measurement of scores stronger expressive ability:

$$s_{i,j} = \sigma(\mathrm{ReLU}\left(\boldsymbol{h}_i \boldsymbol{W}_{i,j}\right) \mathrm{ReLU}(\boldsymbol{h}_j \boldsymbol{W}_{i,j})^T), \quad (5)$$

where $\boldsymbol{W}_{i,j} \in R^{d_h \times d_h}$ is a trainable parameter matrix, $s_{i,j}$ measures the inconsistency between the two words x_i and x_j, σ represents the sigmoid function, and $()^T$ denotes matrix transposition. Then we use the obtained $s_{i,j}$ to construct an undirected and weighted semantic inconsistency graph of the input sequences:

$$\boldsymbol{S} = \begin{bmatrix} s_{1,1} & s_{1,2} & ... & s_{1,n} \\ s_{2,1} & s_{2,2} & ... & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & ... & s_{n,n} \end{bmatrix}. \quad (6)$$

Then, considering that most of the graph structures in the real scene are sparse but our graph is fully connected, we employ a graph sparse mechanism by introducing a non-negative threshold ε. For those elements $s_{i,j}$ that are lower than ε, we set them to zero.

b) Augmenting Affective and Dependency Graphs: To explore the best structure information of affective and syntactical features for each input text, we utilize the learned semantic inconsistent graph $\boldsymbol{S}$ to enhance the affective graph $\boldsymbol{A}$ and the dependency graph $\boldsymbol{D}$ alternatively and dynamically. More concretely, two hyper-parameters are applied to control the integration proportion of features from the two graphs respectively.

$$\boldsymbol{S}_k^A = \alpha \boldsymbol{A} + (1-\alpha)\,\boldsymbol{S}_k, \tag{7}$$

$$\boldsymbol{S}_k^D = \beta \boldsymbol{D} + (1-\beta)\,\boldsymbol{S}_k, \tag{8}$$

where $\boldsymbol{S}_k^A$ and $\boldsymbol{S}_k^D$ represent the graph structures after augmenting affective and dependency graphs respectively, $\alpha \in R$ and $\beta \in R$ are hyper-parameters, $\boldsymbol{A}$ is the affective graph and $\boldsymbol{D}$ is the dependency graph, $\boldsymbol{S}_k$ is the learned semantic inconsistency graph at iteration *k*.

c) Iterative Learning: We feed the augmented graph structures $\boldsymbol{S}_k^A$ and $\boldsymbol{S}_k^D$ in each iteration into the GNNs layer by turn, where we use a multi-layer GCNs network, to get node embeddings $\boldsymbol{Z}_k$ at current iteration rounds. Then we yield a better graph structure in the next iteration based on those node embeddings. Equation (9) indicates how to update the node representation in each iteration.

$$\boldsymbol{Z}_k^l = \varphi_a^l(\varphi_d^l(\boldsymbol{Z}_k^{l-1}, \boldsymbol{S}_k^D), \boldsymbol{S}_k^A), \tag{9}$$

where $\boldsymbol{Z}_k^l \in R^{n \times d_m}$ represents the hidden representations in *l*-th GCN layer at *k*-th iteration, d_m is the dimensionality of hidden representation. φ_d^l and φ_a^l mean the convolution applied to affective graph and dependency graph in the *l*-th layer of GCNs. After going through all the GCN layers, we eventually obtain the node representations in this iteration, which are deployed to learn the inconsistent semantic graph structure in the next iteration. When we get a better structure, we send it to GCNs again to further learn better node representations, and so on. Until the graph structure learned in the two adjacent iterations has been less than a preset threshold θ, we believe that the learning graph structure reaches a stable state and approaches optimal for the sarcasm identification, then the iteration stops.

After that, we perform a max-pooling on the node representations to get the final graph representation $\boldsymbol{G} \in R^{d_m}$. Then we put $\boldsymbol{G}$ into a linear layer with softmax function to calculate the probability distribution of sarcasm decision :

$$\boldsymbol{G} = \max\left(z_{1,j}, z_{2,j}, \cdots, z_{n,j}\right), \forall j \in (1, 2, \cdots, d_m), \tag{10}$$

$$\hat{\boldsymbol{y}} = soft\max\left(\boldsymbol{G}\boldsymbol{K} + \boldsymbol{b}\right), \tag{11}$$

where $z_{i,j}$ represents elements in node representations at position i, j, $\hat{\boldsymbol{y}} \in R^{d_o}$ denotes the probability that the input sequence is a sarcastic utterance. $\boldsymbol{K} \in R^{d_m \times d_o}, \boldsymbol{b} \in R^{d_o}$ are trainable parameters, d_o is the dimensionality of sarcasm labels. Here, we set $d_o = 2$, i.e., sarcasm and non-sarcasm.

Training Objectives

Our model tries to dynamically learn the optimal graph structure, and the quality of the learned graph $\boldsymbol{S}_k$ plays a significant role in improving the quality of the final graph representation for sarcasm detection. Except for the previous definition of loss function for sarcasm detection, we add the graph regularization loss based on the standard cross entropy which is commonly used to improve the quality of learning graph structure. Following (Kalofolias 2016), our graph regularization loss contains two items to control the sparsity and connectivity of the learned graph. We minimize the integrated loss function via the gradient descent algorithm to train the model:

$$\mathcal{L}_{\mathcal{G}} = \sum_{i=1}^{N} \left[\frac{\omega_i}{n_i^2} \left\| \boldsymbol{S}_i \right\|_F^2 + \frac{-\lambda_i}{n_i} 1^T \log\left(\boldsymbol{S}\boldsymbol{1}_i\right) \right], \tag{12}$$

$$\mathcal{L}p = -\sum_{i=1}^{N} \sum_{j=1}^{d_o} y_i^j \log \hat{y}_i^j + \eta R, \tag{13}$$

$$\mathcal{L}_{all} = \mathcal{L}_{\mathcal{G}} + \mathcal{L}_p, \tag{14}$$

where N is the size of the training set, $\mathcal{L}_{\mathcal{G}}$ denotes the graph regularization loss, $\boldsymbol{S}_i$ is the learned inconsistency graph matrix and $\boldsymbol{S}\boldsymbol{1}_i$ is the node degree vector of instance i, $\|\cdot\|_F$ is the Frobenius norm of a matrix. The two elements in $\mathcal{L}_{\mathcal{G}}$ control sparsity and connection separately. $\mathcal{L}_p$ denotes the cross-entropy loss, y_i and $\hat{y}_i$ denote the ground-truth and predicted label distribution of the input text i separately. R is the standard L_2 regularization and ω, λ, η are hyper-parameters.

The integral training process of the IAAD is described in Algorithm 1.

Algorithm 1: The Overall Training Process of IAAD

Input: training set $T = \{L_1, L_2, ..., L_N\}$, the ground-truth $Y = \{y_1, y_2, ..., y_N\}$, and hyper-parameters $\varepsilon, \alpha, \beta, \theta, \omega, \lambda, \eta$

Output: Ψ (Ψ absorbs all parameters during training process)

```
Let L_all = 0
for each text L_q in set T do
    Let k = 1. (k is the iteration number.)
    H ← L_q using Eq. (1) and (2)
    Let Z_0 = H
    A, D ← L_q using Eq. (3) and (4)
    Stop Condition ← |S_k − S_{k−1}|_F^2 > θ |S_1|_F^2
    while (k==1 or Stop Condition) do
        S_k ← Z_{k−1} using Eq. (5) and (6)
        S_k^D, S_k^A ← (A, D, S_k) using Eq. (7) and (8)
        Z_k ← (S_k^D, S_k^A, Z_{k−1}) using Eq. (9)
        k ← k + 1
    end while
    ŷ ← Z_k using Eq. (10),(11)
end for
L_all ← (L_G, L_p) using Eq. (12), (13), and (14)
Back propagate L_all to update Ψ.
```

Experiments

In this section, we first describe the datasets, baselines, and experimental settings, and then compare the experimental performance of our model to that of other baselines. Finally, we demonstrate the effectiveness of the proposed approach by visualizing the semantic inconsistency graph structures that have been learned.

Datasets	Train	Dev	Test	Total
Tweets(Riloff)	888	296	297	1481
Tweets(Ghosh)	32958	10986	10987	54571
Reddit	15626	5208	5210	26044

Table 1: Statistics of datasets used in experiments

Datasets

We evaluate our model on three labeled sarcasm detection datasets from two commonly used sources: Twitter and Reddit. Specifically, we use two Twitter datasets proposed by (Riloff et al. 2013) and (Ghosh and Veale 2016) separately, and a political subset of the Reddit dataset collected by (Khodak, Saunshi, and Vodrahalli 2017). Detailed statistics of those data are described in Table 1.

Baseline Models

We adopt the following models as comparisons to measure the performance of our proposed model:

- **NBOW** takes the average of the word embedding vectors as sentence expressions and subsequently sends it to a standard logistic regression. It is an effective baseline model despite its simple structure.
- **ATT-LSTM** is proposed by (Yang et al. 2016), which is a hierarchical attention network for text classification and has two levels of attention mechanisms applied at the word and sentence levels respectively.
- **CNN-LSTM-DNN** (Ghosh and Veale 2016) combines two Convolutional Neural Network (CNN) layers, two Long Short-Term Memory (LSTM) layers for feature engineering and a Deep Neural Network (DNN) layer for sarcasm classification.
- **TextCNN** (Kim 2014) shows a simple convolutional neural network (CNN) trained with pre-trained word embeddings.
- **BERT** is a pre-trained language representation model proposed by (Devlin et al. 2018), which shows significant superiority in many NLP tasks.
- **roBERTa** (Liu et al. 2019b) is a variant of the BERT model pre-trained on a large corpus, has a larger batch size, and takes a longer time to train.
- **SAWS** (Pan et al. 2020) captures the incongruity among the sentence snippets, which contains an importance weighting module and a self-attention module and achieves excellent performance in capturing inconsistent features in the text.
- **SarDeCK** (Li et al. 2021) is a deep learning architecture for sarcasm detection by introducing commonsense knowledge with a pre-trained COMET model.
- **ADGCN** (Lou et al. 2021) derives a static affective graph and a static dependency graph for each sentence to capture incongruity sentiment expression and syntactic structure respectively. Then, a multi-layer graph convolutional network is leveraged to encode the information conveyed by the two graphs. This model is very similar to our static affective and dependency graphs construction module.

Experimental Settings

In our experiments, we employ GloVe (Pennington, Socher, and Manning 2014) as our initial word embeddings with a fixed vector size of 300. We adopt Adam (Kingma and Ba 2014) optimizer to optimize parameters and set the learning rate equal to 10^{-3}. The L_2 regularization is set to 10^{-5} and the batch size is fixed to 16 for all the datasets. We set ε and θ as 0.3 and 0.04 respectively. We apply the early stop strategy with 10 patiences in our experiments. The training set, validation set, and test set are partitioned by a 6:2:2 spilt.

Experiment Results

In this part, we compare the performance of our model with the above baselines on two standard metrics, including Accuracy[1] and F1-score[2]. Table 2 shows the experimental results of all compared models on three datasets. Our IAAD model has the best performance across all three datasets. Specifically, compared to the state-of-the-art ADGCN model, our approach improves sarcasm detection performance by 8.17%, 1.06%, and 1.75% on the Tweets(Riloff), Tweets(Ghosh), and Reddit(pol) datasets, respectively, in terms of F1-score.

The aforementioned results imply that our model is more effective than other models at detecting sarcastic utterances. We believe that this improvement benefits from the enhancement of the adaptive inconsistent graph learning framework. Our model is capable of dynamically learning the optimal graph structure for sarcasm detection. Thus, the substantial structural information hidden behind text could be fully exploited.

Ablation Study

In this section, we conduct a series of ablation experiments to test the effectiveness of different modules and report the F1-score results in Table 3. Firstly, we remove the affective graph and get the IAAD (w\A). IAAD (w\A) just leverages the dependency graph structure feed to GCNs and then conducts iterative learning. Similarly, we omit the dependency graph and get the IAAD (w\D), so the knowledge about the sentiment incongruities of the sentence is missing in this way.

Then, we eliminate the iterative incongruity graph learning module and get the model IAAD (w\iter). Without this module, node embeddings and incongruity graph structure could no longer alternate update, thus the inevitable noisy information hidden in affective and dependency graph will not be filtered and impact the training of the whole network.

Table 3 indicates that both IAAD (w\A) and IAAD (w\D) result in lower performance. The results suggest that the

[1]Accuracy describes the ratio of correctly classified samples to the total samples.

[2]F1-score is a trade-off between the precision and the recall, which gives a more comprehensive measurement of model performance.

Models	Tweets(Riloff)		Tweets(Ghosh)		Reddit(pol)	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
NBOW	0.7580	0.6632	0.7710	0.7706	0.6347	0.6311
ATT-LSTM	0.7770	0.5964	0.7252	0.7014	0.6969	0.7009
TextCNN	0.7512	0.6269	0.8484	0.8380	0.6977	0.6902
CNN-LSTM-DNN	0.7433	0.6272	0.7418	0.7453	0.6783	0.6766
BERT	0.8216	0.6929	0.7923	0.7883	0.7281	0.7192
roBERTa	0.7791	0.7727	0.7237	0.7223	0.6673	0.6671
SAWS	0.7770	0.5510	0.8228	0.8136	0.7082	0.7187
SarDeCK	0.8513	0.7352	0.8401	0.8335	0.7351	0.7316
ADGCN	0.8514	0.7701	0.8503	0.8427	0.7320	0.7248
OURS	**0.8620**	**0.8518**	**0.8602**	**0.8533**	**0.7408**	**0.7423**

Table 2: Main experimental results on three datasets. The best results are in bold.

Models(%)	Tweets(R)		Tweets(G)		Reddit	
	Acc	F1	Acc	F1	Acc	F1
IAAD	**86.20**	**85.18**	**86.02**	**85.33**	**74.08**	**74.23**
w\A	84.84	82.86	83.16	82.97	71.70	71.74
w\D	84.51	83.95	83.96	83.64	72.28	72.09
w\iter	83.50	81.71	82.78	82.19	71.21	70.60

Table 3: Main experimental results of ablation study. The best results are in bold. The three datasets are abbreviated as Tweets(R),Tweets(G) and Reddit.

preservation of emotionally inconsistent features is crucial for the learning of sarcastic expressions. It also highlights the importance of using global syntactic structure to extract salient incongruous expressions more effectively.

IAAD (w\iter) also results in poorer performance, proving the effectiveness of utilizing the dynamically learned inconsistency graph to strengthen the affective dependency graph, as well as the iterative updating mechanism could efficiently capture the node representations and graph structures which are most instrumental for sarcasm detection.

Visualization

To highlight the role of the iterative incongruity graph learning module in sarcasm detection, Figure 3 depicts how the module improves and corrects the initial affective and dependency graph. We use the non-sarcastic example *"It has a bad memory but a great battery life"* as mentioned above. Each node in Figure 3 (a)-(d) represents a single word in the text. The words in our example are numbered sequentially from 1 to 10. Figure 3 (e) illustrates the correspondence between words and numbers.

Figure 3 (a) represents the initial affective graph structure derived from Equation (3), and Figure 3 (b) is the final fused affective graph after iterative learning. The edges between nodes indicate the degree of sentimental inconsistency. We adopt the thickness of the edge to represent the sentiment inconsistency score. Only four nodes (words) having edges are shown in the graph, the rest isolated nodes are omitted. In the initial affective graph, there is a significant discordance between nodes 4 ("*bad*") and 8 ("*great*"), which would have made it extremely likely to misclassify the sentence. However, after the fusion of iterative learning, the thickness of the edges between nodes 4 and 8 is significantly reduced, indicating that the role played by the inconsistency between the two words has been greatly diminished. Thus our model could pay less attention to the word pair *"bad"* and *"great"* and then predicts its label correctly. We owe this to the iterative incongruity graph learning module, where the affective graph is dynamically enhanced and modified with syntactic information and semantic features.

Similarly, Figure 3 (c) and (d) represent the comparison between the initial dependency graph and the augmented dependency graph. We plot the initial dependency graph in Figure 3 via Equation (4), which is an unweighted graph. Consequently, with the original graph, all syntax dependencies would receive equal attention, which puts a great deal of redundant information into the training process and may cause the model to become confused. Nevertheless, after filtering and integrating by our method, the final dependency graph is depicted in Figure 3 (d), where the thickness of the edges also denotes their weight. We can see that the attention varies with different syntactic relations, and that the dependencies between nodes 4 and 5 have been greatly strengthened, i.e., the relationship between the adjective *"bad"* and the noun *"memory"* is in the spotlight. Similarly, the relationship between the word pairs *"great"* and *"battery"* also garners a good amount of attention. Thus, our model learns that the nouns modified by the inconsistent sentiment pairs *"bad"* and *"great"* are not identical, as indicated by the arrow in Figure 3 (e), and that they do not constitute an affective conflict, allowing it to accurately identify the non-sarcastic label of the example text.

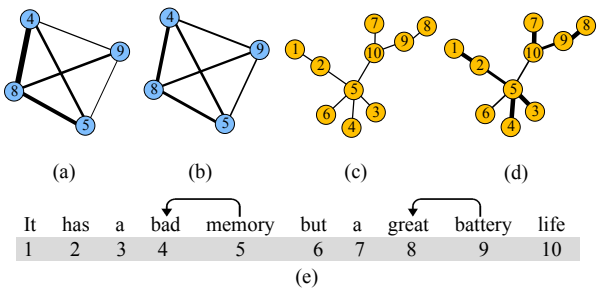

Figure 3: Visualization of affective and dependency graphs. (a) and (b) illustrate the initial and iterative learned affective graph respectively; (c) and (d) depict the initial and iterative learned dependency graph; (e) reveals the words in a sentence which are represented by nodes marked with a specific number. The arrows indicate the syntactic relationships between words.

Related Work

In this section, we provide an overview of the relevant work, including research trends in sarcasm detection and related work with Graph Neural Networks utilized for Natural Language Processing.

Sarcasm Detection

As a universal linguistic phenomenon, sarcasm detection has aroused the attention of numerous linguistic scholars and NLP researchers in recent years (Moores and Mago 2022). The existing approaches involve three main streams: rule-based approaches, machine-learning approaches and deep-learning approaches.

Rule-based approaches aim to detect sarcastic utterances through several typical linguistic evidence (Bamman and Smith 2015). As hashtags may imply the author's emotional attitude or some clues and patterns about irony, the hashtags-based approach was extensively studied by (Kunneman et al. 2015) in the early years. However, rule-based methods are largely dependent on hand-made linguistic patterns, which are time-consuming and with poor generalization. To eliminate these defects, machine learning methods are proposed. For example, support vector machines (SVM) is applied by (Joshi et al. 2016) for sarcasm identification. In more recent approaches, deep learning models have been widely explored to extract features from text automatically. (Yang et al. 2016) proposes an ATT-LSTM framework for irony recognition, which has two levels of attention mechanisms applied at the word and sentence levels respectively. As the success of pre-trained BERT model in many NLP tasks, (Babanejad et al. 2020) trains a new affective BERT model from scratch to jointly encode sentiment and context information. (Lou et al. 2021) firstly explores a new way to detect sarcasm by feeding an affective and a dependency graph for each instance into a multi-layer Graph Convolution Network.

Graph Neural Networks for Natural Language Processing

With the development of Graph Neural Networks(GNNs) (Jin et al. 2021b; Wu et al. 2022) and plenty of information (i.e. dependency parsing tree and semantic parsing graph) in the text could be presented in the structured form, GNNs variants (Jin et al. 2021a; Yu et al. 2021) applied in in many NLP tasks achieve significant success. Depending on how the graph is constructed from text, we categorize GNNs used in NLP tasks into two distinct categories: static and dynamic graph-based approaches. Static graph-based approaches are based on external knowledge and specific rules to mine the structural information implied in the text during pre-processing including Co-occurrence Graph (Christopoulou, Miwa, and Ananiadou 2019), and Dependency Graph (Vashishth et al. 2018) etc. Due to the inevitable noise and incompleteness of manual static graph construction, dynamic graph-based methods have received more attention recently, which aim to learn the graph structure dynamically during training. In particular, (Liu et al. 2019a) proposes a contextualized non-local neural network, in which structures of sentences can be learned on the fly.

Regarding the use of graphs in sarcasm detection, existing work like (Lou et al. 2021) employs a static graph-building approach, which is subject to rules and inherently noisy. Moreover, dynamic graph-based methods have not been fully implemented in this task. Our suggested model addresses in the gap by integrating dynamic graphs as a supplement and augmentation to static graphs, allowing it to dynamically filter noisy information depending on various scenarios.

Conclusion

Due to unsatisfactory parsing performance and the arbitrariness of the input sentences, the straight application of static affective and dependency graphs can invariably result in the inclusion of noisy data. In this paper, we propose a novel Iterative Augmenting Affective Graph and Dependency Graph (IAAD) framework, which is capable of augmenting the affective dependency graph through dynamic incongruity graph learning. In particular, we first use a typical Bi-LSTM to extract contextual information and then construct an affective graph and a dependency graph for each input sequence to exploit the sentiment and syntactic aspects of incongruity expressions. In addition, to complement the affective and dependency graphs, we provide an iterative incongruity graph learning module for getting the optimal inconsistent semantic network for the sarcasm detection task. Experiments give solid evidence for the effectiveness of our model. We also find empirical evidence supporting the effectiveness of Graph Neural Networks applied to the sarcasm detection task, demonstrating that maximizing the utilization of structured information in the text improves performance. In the future, we plan to consider the edge information of the dependency trees for better-constructing dependency graphs dynamically with little noise.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (No. 62272340, 62276187, 62172056) and the Tianjin Municipal Science and Technology Project (Grant No. 19ZXZNGX00030).

References

Babanejad, N.; Davoudi, H.; An, A.; and Papagelis, M. 2020. Affective and contextual embedding for sarcasm detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, 225–243.

Bakshi, R. K.; Kaur, N.; Kaur, R.; and Kaur, G. 2016. Opinion mining and sentiment analysis. In *Proceedings 3rd International Conference on Computing for Sustainable Global Development*, 452–455. IEEE.

Bamman, D.; and Smith, N. 2015. Contextualized sarcasm detection on twitter. In *Proceedings of the 29th International AAAI Conference on Web and Social Media*, 574–577.

Cambria, E.; Li, Y.; Xing, F. Z.; Poria, S.; and Kwok, K. 2020. SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 105–114.

Christopoulou, F.; Miwa, M.; and Ananiadou, S. 2019. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. *arXiv preprint arXiv:1909.00228*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Hazarika, D.; Poria, S.; Gorantla, S.; Cambria, E.; Zimmermann, R.; and Mihalcea, R. 2018. CASCADE: contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics*, 1837–1848.

Jin, D.; Huo, C.; Liang, C.; and Yang, L. 2021a. Heterogeneous graph neural network via attribute completion. In *Proceedings of the 30th Web Conference*, 391–400.

Jin, D.; Yu, Z.; Jiao, P.; Pan, S.; He, D.; Wu, J.; Yu, P.; and Zhang, W. 2021b. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*.

Joshi, A.; Bhattacharyya, P.; and Carman, M. J. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5): 1–22.

Joshi, A.; Sharma, V.; and Bhattacharyya, P. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 757–762.

Joshi, A.; Tripathi, V.; Bhattacharyya, P.; and Carman, M. J. 2016. Harnessing Sequence Labeling for Sarcasm Detection in Dialogue from TV Series' Friends'. In *Proceedings of the 20th Conference on Computational Natural Language Learning*, 146–155.

Kalofolias, V. 2016. How to learn a graph from smooth signals. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistic*, 920–929.

Khodak, M.; Saunshi, N.; and Vodrahalli, K. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kunneman, F.; Liebrecht, C.; Van Mulken, M.; and Van den Bosch, A. 2015. Signaling sarcasm: From hyperbole to hashtag. *Information Processing & Management*, 51(4): 500–509.

Li, J.; Pan, H.; Lin, Z.; Fu, P.; and Wang, W. 2021. Sarcasm Detection with Commonsense Knowledge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29: 3192–3201.

Liu, P.; Chang, S.; Huang, X.; Tang, J.; and Cheung, J. C. K. 2019a. Contextualized non-local neural networks for sequence learning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 6762–6769.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lou, C.; Liang, B.; Gui, L.; He, Y.; Dang, Y.; and Xu, R. 2021. Affective dependency graph for sarcasm detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1844–1849.

Moores, B.; and Mago, V. 2022. A Survey on Automated Sarcasm Detection on Twitter. *arXiv preprint arXiv:2202.02516*.

Pan, H.; Lin, Z.; Fu, P.; and Wang, W. 2020. Modeling the incongruity between sentence snippets for sarcasm detection. In *Proceedings of the 21st European Conference on Artificial Intelligence*, 2132–2139. IOS Press.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing*, 1532–1543.

Riloff, E.; Qadir, A.; Surve, P.; De Silva, L.; Gilbert, N.; and Huang, R. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 18th Conference on Empirical Methods in Natural Language Processing*, 704–714.

Tang, H.; Ji, D.; Li, C.; and Zhou, Q. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6578–6588.

Vashishth, S.; Joshi, R.; Prayaga, S. S.; Bhattacharyya, C.; and Talukdar, P. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. *arXiv preprint arXiv:1812.04361*.

Welling, M.; and Kipf, T. N. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Wu, L.; Cui, P.; Pei, J.; and Zhao, L. 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore.

Xiong, T.; Zhang, P.; Zhu, H.; and Yang, Y. 2019. Sarcasm detection with self-matching networks and low-rank bilinear pooling. In *Proceedings of the 28th World Wide Web Conference*, 2115–2124.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 14th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1480–1489.

Yu, Z.; Jin, D.; Liu, Z.; He, D.; Wang, X.; Tong, H.; and Han, J. 2021. AS-GCN: Adaptive semantic architecture of graph convolutional networks for text-rich networks. In *Proceedings of the 21st IEEE International Conference on Data Mining*, 837–846. IEEE.