TextGT: A Double-View Graph Transformer on Text for Aspect-Based Sentiment Analysis

Shuo Yin, Guoqiang Zhong*

College of Computer Science and Technology, Ocean University of China
yinshuo@stu.ouc.edu.cn, gqzhong@ouc.edu.cn

Abstract

Aspect-based sentiment analysis (ABSA) is aimed at predicting the sentiment polarities of the aspects included in a sentence instead of the whole sentence itself, and is a fine-grained learning task compared to the conventional text classification. In recent years, on account of the ability to model the connectivity relationships between the words in one sentence, graph neural networks have been more and more popular to handle the natural language processing tasks, and meanwhile many works emerge for the ABSA task. However, most of the works utilizing graph convolution easily incur the over-smoothing problem, while graph Transformer for ABSA has not been explored yet. In addition, although some previous works are dedicated to using both GNN and Transformer to handle text, the methods of tightly combining graph view and sequence view of text is open to research. To address the above issues, we propose a double-view graph Transformer on text (TextGT) for ABSA. In TextGT, the procedure in graph view of text is handled by GNN layers, while Transformer layers deal with the sequence view, and these two processes are tightly coupled, alleviating the over-smoothing problem. Moreover, we propose an algorithm for implementing a kind of densely message passing graph convolution called TextGINConv, to employ edge features in graphs. Extensive experiments demonstrate the effectiveness of our TextGT over the state-of-the-art approaches, and validate the TextGINConv module. The source code is available at https://github.com/shuoyinn/TextGT.

Introduction

Aspect-based sentiment analysis (ABSA), as a fine-grained learning task, has been more and more popular in natural language processing (NLP) these years (Wang et al. 2016; Tang, Qin, and Liu 2016; Sun, Huang, and Qiu 2019). The goal of ABSA is to recognize the sentiment polarities of the aspects in a given sentence like a review. The task can be formalized as follows: Given an n-word sentence $S = (w_1, w_2, ..., w_n)$ where w_i denotes the i-th word, and a k-word aspect $S_a = (w_{a_1}, w_{a_2}, ..., w_{a_k}) \subset S$ with a_j standing for a word index in S, we need to predict the sentiment polarity of S_a. Note that a sentence with more than one aspects can be seen as multiple instances each with one aspect. We exemplify the aforementioned task using a sentence with a 3-word aspect, as shown in Figure 1.

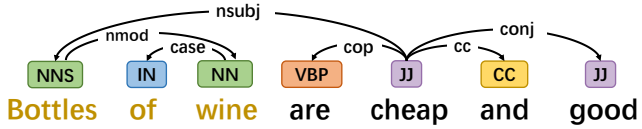

Figure 1: An example sentence with aspect words highlighted in gold. This text has already been preprocessed by a dependency parser.

Due to the great success of deep learning, there are a few approaches to dealing with the ABSA task based on recurrent neural networks (RNNs) or convolutional neural networks (CNNs) (Dong et al. 2014; Vo and Zhang 2015), but they fail to leverage the syntactical information among words. Recently, with the development of graph representation learning, research on graph neural networks (GNNs) for ABSA has been widely used. As shown in Figure 1, the sentence with an aspect is firstly parsed by an external dependency extractor to get the corresponding syntax tree which is regarded as a graph with edge features. Then kinds of GNNs designed in the previous works take the dependency graph as the input and make the subsequent word representations have the syntactic structure information.

However, most GNNs are message passing "smoothers" and have the problem of over-smoothing (Li, Han, and Wu 2018; Chen et al. 2020; Oono and Suzuki 2020). Hopefully, some previous works (Wu et al. 2021; Chen, O'Bray, and Borgwardt 2022; Rampasek et al. 2022) point out that graph Transformers (GTs) have the potential to alleviate over-smoothing. In recent years, several vision Transformers (ViTs) utilizing convolution have become the cutting-edge methods (Li et al. 2021b; Guo et al. 2021; Chen et al. 2021; Peng et al. 2021; Li et al. 2022), whilst GTs also have been a popular topic due to their good performance on graph representation learning tasks like node classification and link prediction (Dwivedi and Bresson 2020; Ying et al. 2021; Kreuzer et al. 2021).

Although there are many works dedicated to design high-performance graph Transformers, few of them are proposed for NLP tasks but graph learning ones. Our motivation is to introduce a well-performed graph Transformer into ABSA,

*Corresponding author

as a text representation learning task, and thus alleviate the problem of over-smoothing. Moreover, inspired by the way of coupling CNN and Transformer tightly to get well-performed ViTs (Guo et al. 2021; Li et al. 2022), we devise a graph Transformer constructed with graph convolutional layers and Transformer layers that appear serially and one by one alternately. In this way, each block of the resulting model is able to learn cohesively in the graph view and meanwhile the sequence view of text.

Our contributions are listed as follows:

- To tightly combine the two text learning processes in the graph view and the sequence view, we propose a novel double-view **g**raph **T**ransformer on **text** called **TextGT**.
- We propose a new algorithm to implement graph convolutional modules which densely pass messages constructed with edge features, and one of such modules called **TextGINConv** is specifically employed as the graph-view operator in our TextGT.
- We have conducted extensive experiments on the ABSA task to demonstrate the effectiveness of TextGT and TextGINConv.

Related Work

In this section, we firstly review the state-of-the-art GNN methods for ABSA, and then several representative graph Transformer works proposed in recent years.

Aspect-Based Sentiment Analysis with GNNs

Wang et al. (2020) propose R-GAT, a tailored graph attention network (GAT) (Velickovic et al. 2018) to leverage the edge features, and using both the dependency relation attention and the normal attention parallelly; besides, they reshape and prune the syntax tree before fed to the network. DualGCN (Li et al. 2021a) and DGEDT (Tang et al. 2020) also employ the parallel architecture where graph convolution and self-attention process the input graph simultaneously and then their outputs will be fused by a biaffine module. Tian et al. (2021) propose a type-aware GCN (T-GCN) where they use the relation type matrix and the adjacency matrix to get a weight matrix, by which node representations are aggregated. SSEGCN (Zhang, Zhou, and Wang 2022) makes use of aspect-aware attention and designs mask matrices based on different distances between nodes in the syntax tree, and thus both the syntactic information and the semantic information are taken into account. AG-VSR (Feng et al. 2022) applies a VAE-like encoder-decoder structure as the pooling method to get the variational sentence representation, which is then used to help predict the sentiment polarity of the aspect. Sentic GCN (Liang et al. 2021) utilizes SenticNet (Cambria et al. 2010), an additional opinion mining tool, to assist GCN in text modeling, and therefore it actually introduces external knowledge. Chen et al. (2022) induce aspect-specific discrete opinion trees by taking attention scores as syntactic distances, and they use reinforcement learning to train the tree inducer. Niu et al. (2022) propose CHGMAN to encode heterogeneous graphs they constructed, and thus the inter-aspect relationships and aspect-context relationships can be considered simultaneously. However, the aforementioned works are actually GNN methods which have the problem of over-smoothing. Although DGEDT uses Transformer to learn long range dependencies, the biaffine fusion method to combine parallel graph convolution and self-attention may limit the model performance.

Graph Transformer

Dwivedi and Bresson (2020) introduce Transformer into graph representation learning, and their graph Transformer performs attention computation only in the neighborhood of a node instead of across the whole graph. Wu et al. (2021) simply stack a multi-layer Transformer encoder on a multi-layer GNN and thus the resulting model called GraphTrans can have the advantages of both. Graphormer (Ying et al. 2021) and SAN (Kreuzer et al. 2021) both utilize specific graph-related positional/structural encodings (PE/SE) and tailored attention mechanisms adapted for the graph data. Chen et al. (2022) interpret a graph Transformer from the perspective of kernel function and make their model (called SAT) aware of the structural information of graphs. Rampasek et al. (2022) summarize many kinds of PE/SE of graph Transformer and propose GraphGPS, a general framework of GTs widely applied to graph prediction tasks. Nevertheless, the models mentioned above are designed mainly for graph learning tasks, like node classification and graph classification for molecules, while graph Transformers for ABSA in NLP has not been specially explored yet.

Analysis on the Over-Smoothing Problem

Thereinafter, we use graph convolution to denote any locally message passing operation on graphs, like one layer of GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), GIN (Xu et al. 2019), GAT (Velickovic et al. 2018), APPNP (Klicpera, Bojchevski, and Günnemann 2019) or PNA (Corso et al. 2020); and for the sake of simplicity, we use "Transformer" to specially refer to Transformer encoder. Here we analyze GNN, Transformer and our double-view graph Transformer in terms of model architecture.

GNN handles text in the syntax tree (graph) view, having the risk of over-smoothing. As shown in Figure 2 (a), several graph convolutional modules constitute a multi-layer GNN. Since graph convolution can be regarded as a smoothing procedure, the constructed GNN is actually a node feature smoother. With the GNN getting deep, the node representations will be over-smoothed, i.e., they will tend to be similar values and hard to distinguish (Li, Han, and Wu 2018; Chen et al. 2020; Oono and Suzuki 2020). Therefore, for GNNs used in NLP, over-smoothing limits the model depth and thus limits the performance. Note that for the ABSA task at hand, the previous GNNs such as R-GAT (Wang et al. 2020), DualGCN (Li et al. 2021a), SSEGCN (Zhang, Zhou, and Wang 2022) and AG-VSR (Feng et al. 2022) all employ no more than 2 layers in most cases.

Transformer deals with text in the sequence view, also having the problem of over-smoothing. See Figure 2 (b),

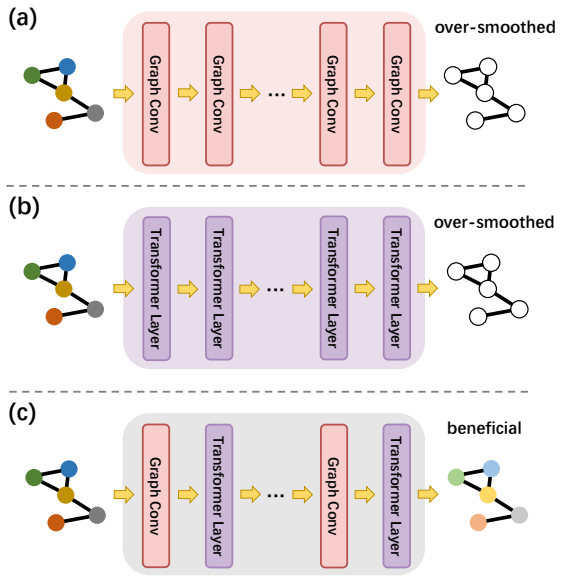

Figure 2: (a) GNN; (b) Transformer; (c) our double-view graph Transformer.

Transformer's full-attention mechanism takes the sequence as a complete graph and thus it can be regarded as a special GNN. Still, over-smoothing also occurs in this scenario if the model gets deep enough, as theoretically proved by Shi et al. (2022).

Our double-view graph Transformer can alleviate over-smoothing. Different from an entire GNN or Transformer, the proposed double-view graph Transformer on text (TextGT) tightly combines the graph operator and the sequence operator (i.e., GNN and Transformer), as Figure 2 (c) shows. Since Transformer can help GNN alleviate over-smoothing (Wu et al. 2021; Chen, O'Bray, and Borgwardt 2022; Rampasek et al. 2022), we also take advantage of it. In TextGT, we alternate the graph convolution layers and the Transformer layers one by one in series, corresponding to syntactic information in the graph view and semantic information in the sequence view, respectively. Intuitively, since these 2 aforementioned smoothing procedures are for different views, they can just somehow "hinder" each other proceeding easily, and thus help each other prevent over-smoothing in time. Hence, the resulting model has the potential to alleviate the problem of over-smoothing, which makes the intermediate representations beneficial to the learning task.

Methodology

In the following, we first elaborate on the whole pipeline of our model for ABSA, including the architecture of the proposed TextGT. Then TextGINConv as the graph convolutional module of TextGT is introduced in detail.

Pipeline

Our model includes 4 parts: dependency parser, text encoder, TextGT and pooling method. Figure 3 shows the whole pipeline.

Dependency parser and preprocessing. We employ Stanford CoreNLP (https://stanfordnlp.github.io/CoreNLP) to preprocess each sentence into a syntax tree, where dependency relationships of some word pairs are parsed out. In such a way we can get the corresponding graph with an adjacency matrix and the element values represent the types of edges. Except for the syntax tree, the dependency parser also outputs the parts of speech (POS) of the words in a same sentence, which can serve as the PE of the graph Transformer.

Text encoder. There are two options of text encoders for TextGT: one is BiLSTM (Graves and Schmidhuber 2005) trained from scratch, and the other is BERT (Devlin et al. 2019) already pre-trained. For BiLSTM we use GloVe (https://github.com/stanfordnlp/GloVe) word vectors as the frozen embeddings, while word embeddings in BERT is trainable. For either of them, we firstly have

$$\boldsymbol{H}_{emb} = embedding_w(S), \tag{1}$$

where $\boldsymbol{H}_{emb} \in \mathbb{R}^{n \times d_w}$ represents d_w-dimensional word embedding matrix of the sentence S. Then if BiLSTM is used, we have

$$\boldsymbol{H}^0 = BiLSTM(\boldsymbol{H}_{emb}), \tag{2}$$

where $\boldsymbol{H}^0 \in \mathbb{R}^{n \times d}$ is the d-dimensional word hidden features; while if BERT then

$$\boldsymbol{H}^0, \boldsymbol{CLS} = BERT(\boldsymbol{H}_{emb}), \tag{3}$$

where $\boldsymbol{CLS} \in \mathbb{R}^d$ is the representation of a special token used for global pooling, i.e., the output from the BERT pooler.

PE. We utilize POS embeddings and relative distance embeddings as the node PE for our graph Transformer. As stated earlier, the POS of each word in a sentence is parsed in the preprocessing step. As for the relative distances from the aspect words, we set a maximum and a minimum, and a specific learnable embedding vector is assigned to each value. The embeddings of POS and relative distance are concatenated to the corresponding word embeddings before fed into the text encoder.

TextGT architecture. As shown in Figure 3, one block of TextGT basically consists of 3 components: TextGINConv, middle layer and Transformer layer. We regard texts (sentences) as double-view data, being graphs in one view and temporal sequences in the other. On the one hand, our TextGINConv processes texts in the graph view, as a dense implementation of GINE (Hu et al. 2020). Roughly speaking, TextGINConv is composed of message constructor, aggregator and updater (i.e., an MLP), which will be detailed in the following subsection. On the other hand, Transformer layer processes texts as sequential data and its attention mechanism is performed on all the words globally in a same instance. In addition, there is a middle layer between the graph operator and the sequence operator, making the two modules compatible with and adapted to each other and thus they can be combined well. The formulas of the aforementioned procedures are as follows:

$$\boldsymbol{O}^l = TextGINConv^{(l)}(\boldsymbol{H}^{l-1}, \boldsymbol{A}), \tag{4}$$

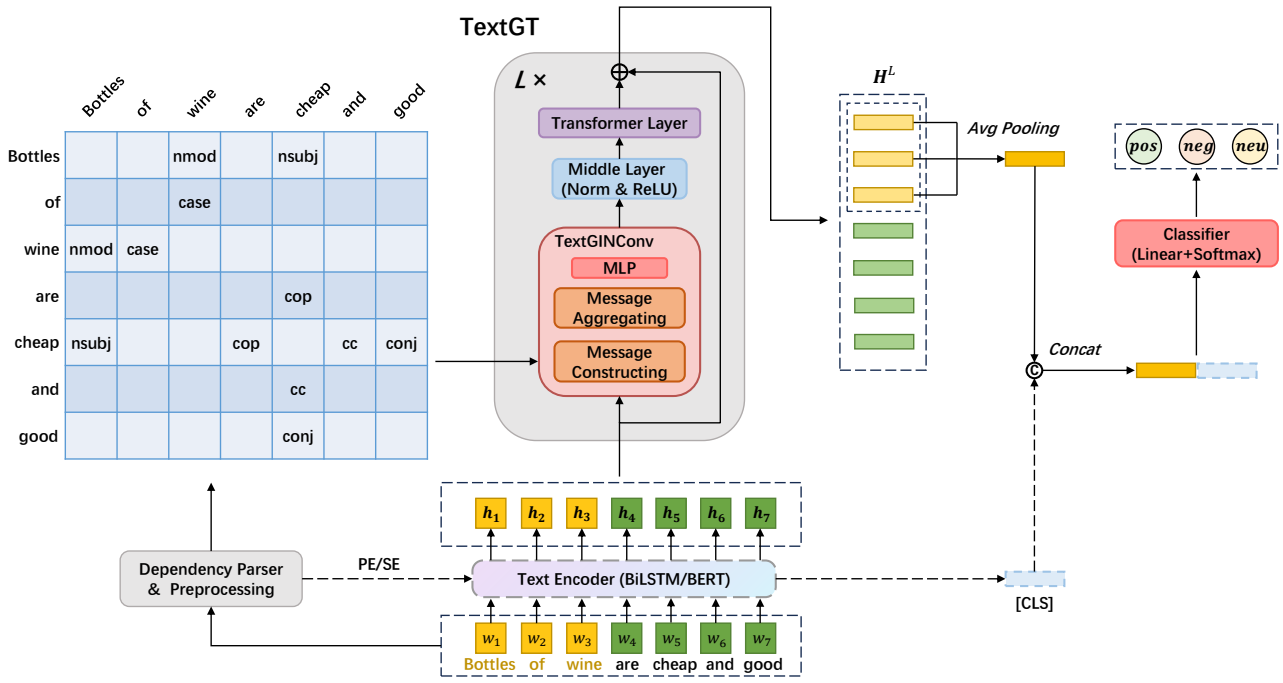

Figure 3: The pipeline of our model for ABSA, including the overview of the proposed TextGT which alternates graph convolutional layers and Transformer layers.

$$\hat{\boldsymbol{H}}^l = ReLU(LayerNorm^{(l)}(\boldsymbol{O}^l)), \tag{5}$$

$$\hat{\boldsymbol{O}}^l = TransformerLayer^{(l)}(\hat{\boldsymbol{H}}^l), \tag{6}$$

$$\boldsymbol{H}^l = \hat{\boldsymbol{O}}^l + \boldsymbol{H}^{l-1}, \tag{7}$$

where $\boldsymbol{H}^{l-1}, \boldsymbol{H}^l \in \mathbb{R}^{n\times d}$ are the word intermediate representations after the $(l-1)$-th and l-th TextGT blocks respectively (totally L blocks), $\boldsymbol{O}^l, \hat{\boldsymbol{O}}^l \in \mathbb{R}^{n\times d}$ are the outputs of the graph convolution layer and the Transformer layer in the l-th block, and $\hat{\boldsymbol{H}}^l \in \mathbb{R}^{n\times d}$ is the output of the middle layer, i.e., LayerNorm followed by ReLU. There is a skip connection from the beginning to the end in each TextGT block, which can also be considered as a jumping knowledge connection (Xu et al. 2018).

Pooling method. We take the output of the L-th TextGT block as the final word representations, and then an average pooling is performed on all the word representations from the same aspect to extract the aspect representation, formulated as:

$$\bar{\boldsymbol{h}} = \frac{\sum_{j=1}^{k} \boldsymbol{H}^L_{a_j}}{k}, \tag{8}$$

where $\bar{\boldsymbol{h}} \in \mathbb{R}^d$ denotes the aspect representation, and the meanings of a_j as well as k is described in the introduction. Afterwards, if BERT serves as the text encoder, then the input of the classifier is $\boldsymbol{p} = [\bar{\boldsymbol{h}} \| \boldsymbol{CLS}]$, where $\boldsymbol{p} \in \mathbb{R}^{2d}$ represents the concatenating result of $\bar{\boldsymbol{h}}$ and $\boldsymbol{CLS}$; otherwise, if BiLSTM is used then $\boldsymbol{p} = \bar{\boldsymbol{h}} \in \mathbb{R}^d$.

TextGINConv

To accommodate graph learning for text, with node and edge features both in dense batch instead of sparse batch implementation in Pytorch Geometric (https://pytorch-geometric.readthedocs.io/en/latest/index.html), we design a dense version of GINE Conv specifically designed for text (thus called **TextGINConv**). Figure 4 shows how TextGINConv deals with an example graph, whose adjacency matrix with edge types is denoted as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, and d-dimensional node features as $\boldsymbol{X} \in \mathbb{R}^{n\times d}$. Before message passing, the mask matrix is generated based on $\boldsymbol{A}$, meanwhile, $\boldsymbol{X}$ is expanded along the row and column to $\boldsymbol{X}_{row} \in \mathbb{R}^{n\times n\times d}$ and $\boldsymbol{X}_{col} \in \mathbb{R}^{n\times n\times d}$, respectively.

In terms of message passing mechanism, the message constructing procedure is:

$$\boldsymbol{E} = embedding_e(\boldsymbol{A}), \tag{9}$$

$$\boldsymbol{M} = mask(\boldsymbol{X}_{row} + \boldsymbol{E}), \tag{10}$$

where $\boldsymbol{E} \in \mathbb{R}^{n\times n\times d}$ denotes the embeddings of the edges, and $\boldsymbol{M} \in \mathbb{R}^{n\times n\times d}$ is the constructed messages corresponding to edges. After masking via the 0-1 mask matrix, we can guarantee any non-edge feature vector is filled with zero in $\boldsymbol{M}$. Then the aggregation is as follows:

$$\hat{\boldsymbol{X}} = f_x(\sum_{j=1}^{n} \boldsymbol{M}_{\cdot j}), \tag{11}$$

$$\hat{\boldsymbol{E}} = f_e(\boldsymbol{E} + \boldsymbol{X}_{row} + \boldsymbol{X}_{col}), \tag{12}$$

where $\hat{\boldsymbol{X}} \in \mathbb{R}^{n\times d}$ and $\hat{\boldsymbol{E}} \in \mathbb{R}^{n\times n\times d}$ represent the aggregating results of nodes and edges respectively, while f_x and f_e are the updating functions like MLP.

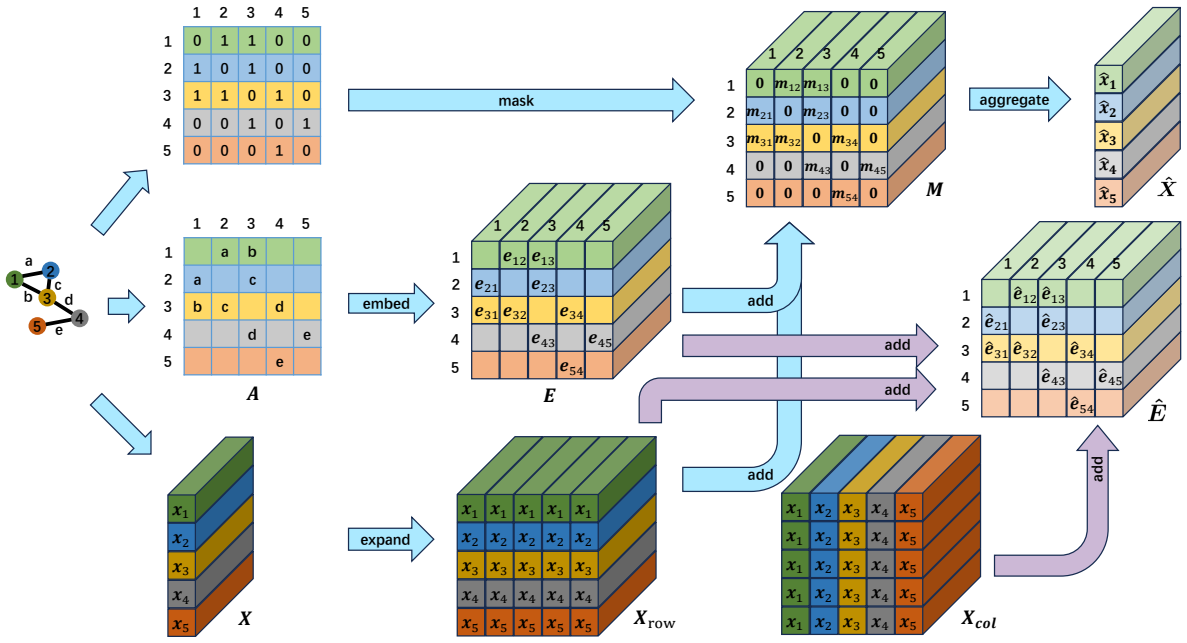

Figure 4: The illustration of our proposed TextGINConv.

One significant step of TextGINConv is to use the mask to filter out the redundant edge features, i.e., set $\boldsymbol{m}_{ij} = 0$ if there is no edge between node i and j, which guarantees noise messages will not influence node/edge representations updating in the next layer. Note that TextGINConv can be generalized to a GNN framework to handle graphs with edge features and dense adjacency matrices, e.g., we can just normalize messages with node degrees / attention scores before aggregation, to realize TextGCNCon/TextGATConv.

In our method, for simplicity the edge updating procedures are not used (purple arrows in Figure 4) and the edge features remain constant while forward propagating.

Experiments

This section presents the experiments we conducted for comparison, ablation study, depth study, case study and visualization.

Datasets

We conduct our experiments on 4 public benchmark datasets for ABSA. Restaurant and Laptop are from SemEval 2014 (Pontiki et al. 2014), consisting of reviews related with restaurants and laptops, respectively; Rest16 is also a collection of reviews about restaurants, provided from SemEval 2016 (Hercig et al. 2016); and Twitter, built by Dong et al. (2014), is composed of twitter posts.

Comparison to the State-of-the-Art Methods

We compare TextGT to the state-of-the-art baselines, which are divided into 4 categories based on whether they use a pre-trained model (i.e., BERT) and whether they use a GNN. (Methods without released source code are out of the comparison.)

Baselines:

a. Methods without GNN or BERT including IAN (Ma et al. 2017), RAM (Chen et al. 2017), MGAN (Fan, Feng, and Zhao 2018) and TNet (Li et al. 2018).

b. Methods with GNN but without BERT including AS-GCN (Zhang, Li, and Song 2019), CDT (Sun et al. 2019), BiGCN (Zhang and Qian 2020), kumaGCN (Chen, Teng, and Zhang 2020), InterGCN (Liang et al. 2020), R-GAT (Wang et al. 2020), DualGCN (Li et al. 2021a), DGEDT (Tang et al. 2020), SSEGCN (Zhang, Zhou, and Wang 2022) and AG-VSR (Feng et al. 2022).

c. BERT (Devlin et al. 2019).

d. Methods with both GNN and BERT including BERT4GCN (Xiao et al. 2021), T-GCN (Tian, Chen, and Song 2021), Sentic GCN (Liang et al. 2021), R-GAT, DualGCN, DGEDT, SSEGCN and AG-VSR.

Setup. According to different situations of the baselines, we conduct 2 comparison experiments: as shown in Table 1, the first is on Restaurant, Laptop and Twitter to demonstrate the effectiveness of our TextGT with BiLSTM (top) and that with BERT (bottom); for the lack of statistics from original papers of GNN baselines on Rest16 and in case of unfair comparison (some use external knowledge or additional supplement datasets), Table 2 only validates TextGT+BERT by comparing to other BERT-based models.

Results. From Table 1 and 2 we can see that whether the metric is accuracy or F1 score, our TextGT (BiLSTM) is consistently the best among all the methods without BERT, and TextGT+BERT achieves the best results in most cases. Although on Laptop and Twitter our TextGT+BERT does not take the first place, the results are comparable and close to the best ones. Note that in Table 2 Sentic GCN is a method

Models	Restaurant		Laptop		Twitter	
	Accuracy ↑	Micro-F1 ↑	Accuracy ↑	Micro-F1 ↑	Accuracy ↑	Micro-F1 ↑
IAN (IJCAI, 2017)	78.60	–	72.10	–	–	–
RAM (EMNLP, 2017)	80.23	70.80	74.49	71.35	69.36	67.30
MGAN (EMNLP, 2018)	81.25	71.94	75.39	72.47	72.54	70.81
TNet (ACL, 2018)	80.69	71.27	76.54	71.75	74.90	73.60
ASGCN (EMNLP, 2019)	80.77	72.02	75.55	71.05	72.15	70.40
CDT (EMNLP, 2019)	82.30	74.02	77.19	72.99	74.66	73.66
BiGCN (EMNLP, 2020)	81.97	73.48	74.59	71.84	74.16	73.35
kumaGCN (EMNLP, 2020)	81.43	73.64	76.12	72.42	72.45	70.77
InterGCN (COLING, 2020)	82.23	74.01	77.86	74.32	–	–
R-GAT (ACL, 2020)	83.30	76.08	77.42	73.76	75.57	73.82
DGEDT (ACL, 2020)	83.90	75.10	76.80	72.30	74.80	73.40
DualGCN (ACL, 2021)	<u>84.27</u>	<u>78.08</u>	<u>78.48</u>	74.74	<u>75.92</u>	74.29
SSEGCN† (NAACL, 2022)	83.29	76.31	77.22	73.53	74.74	73.32
AG-VSR (KBS, 2022)	83.45	76.05	78.16	<u>74.77</u>	75.78	<u>74.33</u>
Our TextGT	**85.17**	**79.70**	**78.64**	**74.91**	**76.51**	**75.26**
BERT (NAACL, 2019)	85.97	80.09	79.91	76.00	75.92	75.18
R-GAT+BERT (ACL, 2020)	86.60	81.35	78.21	74.07	76.15	74.88
DGEDT+BERT (ACL, 2020)	86.30	80.00	79.80	75.60	**77.90**	75.40
BERT4GCN (EMNLP, 2021)	84.75	77.11	77.49	73.01	74.73	73.76
T-GCN+BERT (NAACL, 2021)	86.16	79.95	80.88	77.03	76.45	75.25
DualGCN+BERT (ACL, 2021)	<u>87.13</u>	<u>81.16</u>	**81.80**	<u>78.10</u>	77.40	76.02
SSEGCN+BERT† (NAACL, 2022)	86.15	79.96	79.75	76.38	<u>77.70</u>	<u>76.36</u>
AG-VSR+BERT (KBS, 2022)	86.34	80.88	79.92	75.85	76.45	75.04
Our TextGT+BERT	**87.31**	**82.27**	<u>81.33</u>	**78.71**	<u>77.70</u>	**76.45**

Table 1: Comparison on the benchmark datasets. † indicates the baselines are rerun based on the open source code, and the other results are from the related original papers or (Li et al. 2021a). The best results are highlighted in boldface while the second best ones are underlined, and lacking results are marked as "–".

Models	Rest16	
	Accuracy ↑	Micro-F1 ↑
BERT (NAACL, 2019)	90.10	74.16
R-GAT+BERT (ACL, 2020)	89.71	76.62
DGEDT+BERT (ACL, 2020)	91.90	79.00
DualGCN+BERT† (ACL, 2021)	90.99	78.37
SSEGCN+BERT† (NAACL, 2022)	91.60	<u>79.66</u>
Sentic GCN+BERT (KBS, 2022)	<u>91.97</u>	79.56
Our TextGT+BERT	**92.21**	**81.48**

Table 2: Comparison on Rest16.

using external knowledge, which is unfair to us, but our TextGT still outperforms it by an obvious margin.

Ablation Study

We conduct ablation studies to demonstrate the effectiveness of each components making up TextGT. We replace Transformer layer with TextGINConv and the resulting model is called TextGIN. Moreover, we replace TextGINConv with TextGCNConv/TextGATConv, then get TextGT (GCN) / TextGT (GAT). As shown in Table 3, neither TextGIN nor Transformer alone works well, and GIN is most suitable for TextGT among 3 commonly used graph convolutional modules. Additionally, LayerNorm + ReLU can narrow the gap between TextGINConv and the Transformer layer thus making the overall model perform better.

Depth Study

To further show the superiority of TextGT, we vary model layers and observe test accuracy with respect to model depth. From Figure 5, we can see the performance of R-GAT, DualGCN, TextGIN and Transformer all drop dramatically with the depth increasing, while that of our TextGT keeps at a high level steadily. Consequently, TextGT is able to alleviate over-smoothing and is robust to the increase in model depth.

Case Study

We display some test cases in Table 4, to show the ability of TextGT to discriminate aspects. These reviews are randomly selected across different datasets. The results indicate TextGT can decide the polarity of an aspect better than the baselines, benefiting from the tightly combined 2 operations of the graph view and the sequence view, which shows the advantage of alternating graph convolution and global attention to model the context syntactically and semantically.

Visualization

We further validate TextGT from the perspective of model interpretation. Figure 6 is the visualization result of a word

Models	Restaurant		Laptop		Twitter	
	Accuracy ↑	Micro-F1 ↑	Accuracy ↑	Micro-F1 ↑	Accuracy ↑	Micro-F1 ↑
TextGIN	80.07	72.96	74.68	70.45	74.89	74.15
Transformer	81.77	74.33	75.16	70.97	75.48	74.44
w/o middle layer	83.82	77.54	75.63	71.97	76.22	74.79
TextGT (GCN)	83.56	76.89	77.37	73.77	75.33	74.09
TextGT (GAT)	82.13	75.35	75.63	71.51	75.78	74.81
TextGT (GIN)	**85.17**	**79.70**	**78.64**	**74.91**	**76.51**	**75.26**

Table 3: Ablation results on the benchmark datasets.

#	Reviews	DGEDT	DualGCN	SSEGCN	TextGT
1	**[Pizzas]** were excellent in addition to **[appetizers]** and **[main courses]**.	(P✔, P✔, O✘)	(P✔, O✘, O✘)	(P✔, O✘, P✔)	(P✔, P✔, P✔)
2	It's good to go there for **[drinks]** if you don't want to get drunk because you 'll be lucky if you can get one **[drink]** an hour, the **[service]** is so bad.	(O✔, P✘, N✔)	(O✔, N✘, N✔)	(N✘, P✘, N✔)	(O✔, O✔, N✔)
3	I use it mostly for **[content creation]** ([**audio**], [**video**], [**photo editing**]) and its reliable.	(O✘, O✘, O✘, O✘)	(O✘, O✘, O✘, O✘)	(P✔, O✘, O✘, O✘)	(P✔, P✔, O✘, P✔)
4	Unfortunately, it runs **[XP]** and Microsoft is dropping **[support]** next April.	(N✘, N✔)	(N✘, N✔)	(N✘, N✔)	(O✔, N✔)

Table 4: Case study. We compare our TextGT with the other 3 state-of-the-art methods.

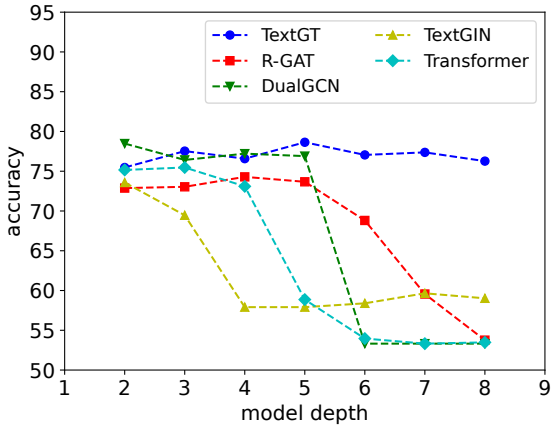

Figure 5: Depth study on Laptop.

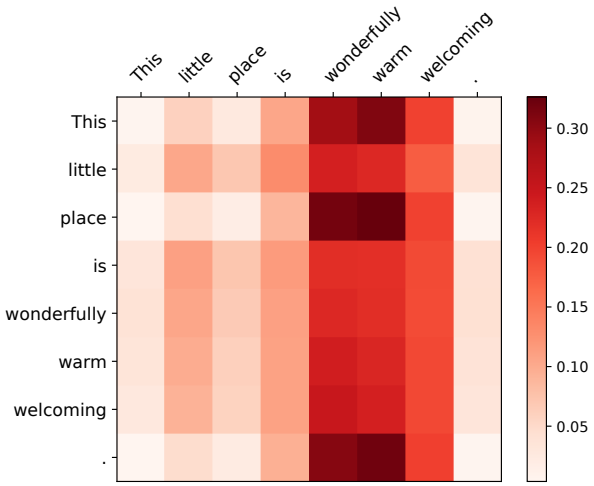

Figure 6: A visualization result.

attention matrix extracted from an intermediate layer of TextGT. The corresponding sentence is "This little place is wonderfully warm welcoming" with a one-word aspect "place" and obviously its polarity is positive. Note that the aspect word "place" attaches most attention on the sentimental and informative words "wonderfully warm welcoming", while almost ignores the interfering sentiment word "little".

Conclusion

In this paper, we analyze the over-smoothing problem in GNN and Transformer from the perspective of model structure. Then we propose a double-view graph Transformer on text called TextGT for ABSA. TextGT alternates the 2 learning processes in the graph view and the sequence view. Adapting GNN for text representation learning syntactically, we propose an algorithm for implementing a kind of densely message passing graph convolution called TextGIN-Conv. Extensive experiments on public benchmark datasets demonstrate the effectiveness of our method. In the future work, we plan to apply our TextGT to other NLP tasks, and we will further explore other schemes to combine graph convolution and self-attention to get high-performance graph Transformers for NLP.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China under Grant No. 2018AAA0100400, HY Project under Grant No. LZY2022033004, the Natural Science Foundation of Shandong Province under Grants No. ZR2020MF131 and No. ZR2021ZD19, the Science and Technology Program of Qingdao under Grant No. 21-1-4-ny-19-nsh, and Project of Associative Training of Ocean University of China under Grant No. 202265007.

References

Cambria, E.; Speer, R.; Havasi, C.; and Hussain, A. 2010. SenticNet: A Publicly Available Semantic Resource for Opinion Mining. 14–18.

Chen, C.; Teng, Z.; Wang, Z.; and Zhang, Y. 2022. Discrete Opinion Tree Induction for Aspect-based Sentiment Analysis. 2051–2064.

Chen, C.; Teng, Z.; and Zhang, Y. 2020. Inducing Target-specific Latent Structures for Aspect Sentiment Classification. 5596–5607.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*.

Chen, D.; O'Bray, L.; and Borgwardt, K. 2022. Structure-aware transformer for graph representation learning. In *ICML*, 3469–3489. PMLR.

Chen, P.; Sun, Z.; Bing, L.; and Yang, W. 2017. Recurrent Attention Network on Memory for Aspect Sentiment Analysis. 452–461.

Chen, Y.; Dai, X.; Chen, D.; Liu, M.; Dong, X.; Yuan, L.; and Liu, Z. 2021. Mobile-Former: Bridging MobileNet and Transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5260–5269.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Veličković, P. 2020. Principal neighbourhood aggregation for graph nets. *NeurIPS*, 13260–13271.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics.

Dong, L.; Wei, F.; Tan, C.; Tang, D.; Zhou, M.; and Xu, K. 2014. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. 49–54.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs. *CoRR*, abs/2012.09699.

Fan, F.; Feng, Y.; and Zhao, D. 2018. Multi-grained Attention Network for Aspect-Level Sentiment Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3433–3442. Association for Computational Linguistics.

Feng, S.; Wang, B.; Yang, Z.; and Ouyang, J. 2022. Aspect-based sentiment analysis with attention-assisted graph and variational sentence representation. *Knowl. Based Syst.*, 258: 109975.

Graves, A.; and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, 18 5-6: 602–10.

Guo, J.; Han, K.; Wu, H.; Xu, C.; Tang, Y.; Xu, C.; and Wang, Y. 2021. CMT: Convolutional Neural Networks Meet Vision Transformers. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12165–12175.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 1024–1034.

Hercig, T.; Brychcin, T.; Svoboda, L.; and Konkol, M. 2016. UWB at SemEval-2016 Task 5: Aspect Based Sentiment Analysis. 342–349.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; and Leskovec, J. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Kreuzer, D.; Beaini, D.; Hamilton, W. L.; Létourneau, V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. In *NeurIPS*, 21618–21629.

Li, J.; Xia, X.; Li, W.; Li, H.; Wang, X.; Xiao, X.; Wang, R.; Zheng, M.; and Pan, X. 2022. Next-ViT: Next Generation Vision Transformer for Efficient Deployment in Realistic Industrial Scenarios. *ArXiv*, abs/2207.05501.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

Li, R.; Chen, H.; Feng, F.; Ma, Z.; Wang, X.; and Hovy, E. 2021a. Dual Graph Convolutional Networks for Aspect-based Sentiment Analysis. 6319–6329.

Li, X.; Bing, L.; Lam, W.; and Shi, B. 2018. Transformation Networks for Target-Oriented Sentiment Classification. *ArXiv*, abs/1805.01086.

Li, Y.; Yao, T.; Pan, Y.; and Mei, T. 2021b. Contextual Transformer Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP: 1–1.

Liang, B.; Su, H.; Gui, L.; Cambria, E.; and Xua, R. 2021. Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks. *Knowl. Based Syst.*, 235: 107643.

Liang, B.; Yin, R.; Gui, L.; Du, J.; and Xu, R. 2020. Jointly Learning Aspect-Focused and Inter-Aspect Relations with Graph Convolutional Networks for Aspect Sentiment Analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, 150–161. International Committee on Computational Linguistics.

Ma, D.; Li, S.; Zhang, X.; and Wang, H. 2017. Interactive Attention Networks for Aspect-Level Sentiment Classification. *ArXiv*, abs/1709.00893.

Niu, H.; Xiong, Y.; Gao, J.; Miao, Z.; Wang, X.; Ren, H.; Zhang, Y.; and Zhu, Y. 2022. Composition-based Heterogeneous Graph Multi-channel Attention Network for Multi-aspect Multi-sentiment Classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, 6827–6836. International Committee on Computational Linguistics.

Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*.

Peng, Z.; Huang, W.; Gu, S.; Xie, L.; Wang, Y.; Jiao, J.; and Ye, Q. 2021. Conformer: Local Features Coupling Global Representations for Visual Recognition. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 357–366.

Pontiki, M.; Galanis, D.; Pavlopoulos, J.; Papageorgiou, H.; Androutsopoulos, I.; and Manandhar, S. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *International Workshop on Semantic Evaluation*, 27–35.

Rampasek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. In *NeurIPS*.

Shi, H.; Gao, J.; Xu, H.; Liang, X.; Li, Z.; Kong, L.; Lee, S. M. S.; and Kwok, J. 2022. Revisiting Over-smoothing in BERT from the Perspective of Graph. *ArXiv*, abs/2202.08625.

Sun, C.; Huang, L.; and Qiu, X. 2019. Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 380–385. Association for Computational Linguistics.

Sun, K.; Zhang, R.; Mensah, S.; Mao, Y.; and Liu, X. 2019. Aspect-Level Sentiment Analysis Via Convolution over Dependency Tree. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5679–5688. Association for Computational Linguistics.

Tang, D.; Qin, B.; and Liu, T. 2016. Aspect Level Sentiment Classification with Deep Memory Network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 214–224. Association for Computational Linguistics.

Tang, H.; Ji, D.; Li, C.; and Zhou, Q. 2020. Dependency Graph Enhanced Dual-transformer Structure for Aspect-based Sentiment Classification. 6578–6588.

Tian, Y.; Chen, G.; and Song, Y. 2021. Aspect-based Sentiment Analysis with Type-aware Graph Convolutional Networks and Layer Ensemble. 2910–2922.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Vo, D.; and Zhang, Y. 2015. Target-Dependent Twitter Sentiment Classification with Rich Automatic Features. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 1347–1353. AAAI Press.

Wang, K.; Shen, W.; Yang, Y.; Quan, X.; and Wang, R. 2020. Relational Graph Attention Network for Aspect-based Sentiment Analysis. 3229–3238.

Wang, Y.; Huang, M.; Zhu, X.; and Zhao, L. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, 606–615.

Wu, Z.; Jain, P.; Wright, M.; Mirhoseini, A.; Gonzalez, J. E.; and Stoica, I. 2021. Representing long-range context for graph neural networks with global attention. *NeurIPS*, 13266–13279.

Xiao, Z.; Wu, J.; Chen, Q.; and Deng, C. 2021. BERT4GCN: Using BERT Intermediate Layers to Augment GCN for Aspect-based Sentiment Classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 9193–9200. Association for Computational Linguistics.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 5453–5462. PMLR.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T. 2021. Do Transformers Really Perform Badly for Graph Representation? In *NeurIPS*, 28877–28888.

Zhang, C.; Li, Q.; and Song, D. 2019. Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4568–4578. Association for Computational Linguistics.

Zhang, M.; and Qian, T. 2020. Convolution over Hierarchical Syntactic and Lexical Graphs for Aspect Level Sentiment Analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3540–3549. Association for Computational Linguistics.

Zhang, Z.; Zhou, Z.; and Wang, Y. 2022. SSEGCN: Syntactic and Semantic Enhanced Graph Convolutional Network for Aspect-based Sentiment Analysis. 4916–4925.