Graph Ensemble Learning over Multiple Dependency Trees for Aspect-level Sentiment Classification

Xiaochen Hou[1*], **Peng Qi**[1], **Guangtao Wang**[1], **Rex Ying**[2], **Jing Huang**[1], **Xiaodong He**[1], **Bowen Zhou**[1]
[1]JD AI Research, Mountain View, CA
[2]Department of Computer Science, Stanford University, Stanford, CA
[*]xclmm1994@gmail.com

Abstract

Recent work on aspect-level sentiment classification has demonstrated the efficacy of incorporating syntactic structures such as dependency trees with graph neural networks (GNN), but these approaches are usually vulnerable to parsing errors. To better leverage syntactic information in the face of unavoidable errors, we propose a simple yet effective graph ensemble technique, GraphMerge, to make use of the predictions from different parsers. Instead of assigning one set of model parameters to each dependency tree, we first combine the dependency relations from different parses before applying GNNs over the resulting graph. This allows GNN models to be robust to parse errors at no additional computational cost, and helps avoid overparameterization and overfitting from GNN layer stacking by introducing more connectivity into the ensemble graph. Our experiments on the SemEval 2014 Task 4 and ACL 14 Twitter datasets show that our GraphMerge model not only outperforms models with single dependency tree, but also beats other ensemble models without adding model parameters.

1 Introduction

Aspect-level sentiment classification is a fine-grained sentiment analysis task, which aims to identify the sentiment polarity (e.g., positive, negative or neutral) of a specific aspect term in a sentence. For example, in "*The exterior, unlike the food, is unwelcoming.*", the polarities of aspect terms "exterior" and "food" are negative and positive, respectively. This task has many applications, such as assisting customers to filter online reviews or make purchase decisions on e-commerce websites.

Recent studies have shown that syntactic information such as dependency trees is very effective in capturing long-range syntactic relations that are obscure from the surface form (Zhang et al., 2018). Several successful approaches employed graph neural network (GNN) (Kipf and Welling, 2016) model over dependency trees to aspect-level sentiment classification (Huang and Carley, 2019; Zhang et al., 2019; Sun et al., 2019; Wang et al., 2020b), which demonstrate that syntactic information is helpful for associating the aspect term with relevant opinion words more directly for increased robustness in sentiment classification.

Figure 1: An example where an incorrect parse (above the sentence) can mislead aspect-level sentiment classification for the term "food" by connecting it to the negative sentiment word "unwelcoming" by mistake. Although having its own issues, the parse below correctly captures the main syntactic structure between the aspect terms "exterior", "food" and the sentiment word, and is more likely to lead to a correct prediction.

However, existing approaches are vulnerable to parsing errors (Wang et al., 2020b). For example, in Figure 1, the blue parse above the sentence can mislead models to predict negative sentiment for the aspect term "food" with its direct association to "unwelcoming". Despite their high edge-wise parsing performance on standard benchmarks, state-of-the-art dependency parsers usually struggle to predict flawless parse trees especially in out-of-domain settings. This poses great challenge to dependency-based methods that rely on these parse trees—the added benefit from syntactic structure does not always prevail the noise introduced by model-predicted parses (He et al., 2017; Sachan et al., 2021).

In this paper, we propose GraphMerge, a *graph ensemble* technique to help dependency-based models mitigate the effect of parsing errors. Our technique is based on the observation that different parsers, especially ones with different inductive biases, often err in different ways. For instance,

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2884–2894
June 6–11, 2021.

in Figure 1, the green parse under the sentence is incorrect around "unlike the food", but it nevertheless correctly associates "unwelcoming" with the other aspect term "exterior", and therefore is less likely to mislead model predictions. Given dependency trees from multiple parses, instead of assigning each dependency tree a separate set of model parameters and ensembling model predictions or dependency-based representations of the same input, we propose to combine the different dependency trees before applying representation learners such as GNNs.

Specifically, we take the union of the edges in all dependency trees from different parsers to construct an ensemble graph, before applying GNNs over it. This exposes the GNN model to various graph hypotheses at once, and allows the model to learn to favor edges that contribute more to the task. To retain the syntactic dependency information between words in the original dependency trees, we also define two different edge types—parent-to-children and children-to-parent—which are encoded by applying relational graph attention networks (RGAT) (Busbridge et al., 2019) on the ensemble graph.

Our approach has several advantages. Firstly, since GraphMerge combines dependency trees from different parsers, the GNN models can be exposed to multiple parsing hypotheses and learn to choose edges that are more suitable for the task from data. As a result, the model is less reliant on any specific parser and more robust to parsing errors. Secondly, this improved robustness to parsing errors does not require any additional computational cost, since we are still applying GNNs to a single graph with the same number of nodes. Last but not least, GraphMerge helps prevent GNNs from overfitting by limiting over-parameterization. Aside from keeping the GNN computation over a single graph to avoid separate parameterization for each parse tree, GraphMerge also introduces more edges in the graph when parses differ, which reduces the diameter of graphs. As a result, fewer layers of GNNs are needed to learn good representations from the graph, alleviating the over-smoothing problem (Li et al., 2018b).

To summarize, the main contribution of our work are the following:

- We propose a GraphMerge technique to combine dependency parsing trees from different parsers to improve model robustness to parsing errors. The ensemble graph enables the model to learn from noisy graph and select correct edges among nodes at no additional computational cost.
- We retain the syntactic dependency information in the original trees by parameterizing parent-to-children and children-to-parent edges separately, which improves the performance of the RGAT model on the ensemble graph.
- Our GraphMerge RGAT model outperforms recent state-of-the-art work on three benchmark datasets (Laptop and Restaurant reviews from SemEval 2014 and the ACL 14 Twitter dataset). It also outperforms its single-parse counterparts as well as other ensemble techniques.

2 Related Work

Much recent work on aspect-level sentiment classification has focused on applying attention mechanisms (e.g., co-attention, self attention, and hierarchical attention) to sequence models such recurrent neural networks (RNNs) (Tang et al., 2015, 2016; Liu and Zhang, 2017; Wang et al., 2018; Fan et al., 2018; Chen et al., 2017; Zheng and Xia, 2018; Wang and Lu, 2018; Li et al., 2018a,c). In a similar vein, pretrained transformer language models such as BERT (Devlin et al., 2018) have also been applied to this task, which operates directly on word sequences (Song et al., 2019; Xu et al., 2019; Rietzler et al., 2019).

In parallel, researchers have also found syntactic information to be helpful for this task, and incorporated it into aspect-level sentiment classification models in the form of dependency trees (Dong et al., 2014; He et al., 2018) as well as constituency trees (Nguyen and Shirai, 2015). More recently, researchers have developed robust dependency-based models with the help of GNNs that operate either directly on dependency trees (Huang and Carley, 2019; Zhang et al., 2019; Sun et al., 2019), as well as reshaped dependency trees that center around aspect terms (Wang et al., 2020b). While most recent work stack GNNs on top of BERT models, Tang et al. (2020) have also reported gains by jointly learning the two with a mutual biaffine attention mechanism.

Despite the success of these dependency-based models, they are usually vulnerable to parse errors since they rely on a single parser. Tu et al. (2012) used a dependency forest to combine multiple dependency trees, however they tackled the sentence-level sentiment analysis task instead, and

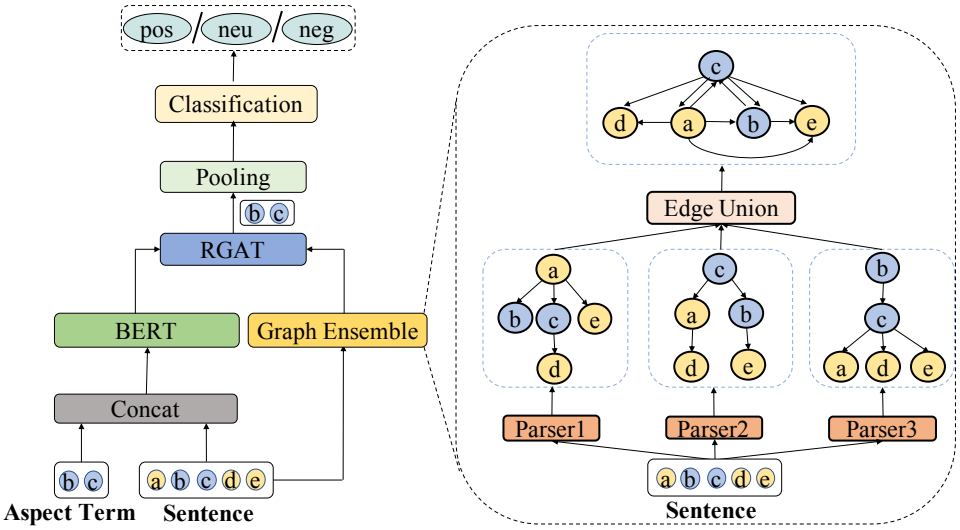


Figure 2: The framework of the GraphMerge model for aspect-level sentiment classification over multiple dependency trees. The left side shows the overall architecture for sentiment classification; and the right side shows the details of how to perform graph ensemble with GraphMerge.

their proposed ensemble technique is also significantly different from ours. Furthermore, most prior work that leverage GNNs to encode dependency information treats the dependency tree as an undirected graph, therefore ignores the syntactic relation between words in the sentence.

3 Proposed Model

We are interested in the problem of predicting the sentiment polarity of an aspect term in a given sentence. Specifically, given a sentence of n words $\{w_1, w_2, \ldots, w_\tau, \ldots, w_{\tau+t}, \ldots, w_n\}$ where $\{w_\tau, w_{\tau+1}, \ldots, w_{\tau+t-1}\}$ is the aspect term, the goal is to classify the sentiment polarity toward the term as positive, negative, or neutral. Applying GNNs over dependency trees is shown effective to solve this problem, however it is vulnerable to parsing errors. Therefore, we propose a GraphMerge technique to utilize multiple dependency trees to improve robustness to parsing errors. In this section, we will first introduce GraphMerge, our proposed graph ensemble technique, then introduce the GNN model over GraphMerge graph for aspect-level sentiment analysis.

3.1 GraphMerge over Multiple Dependency Trees

To allow graph neural networks to learn dependency-based representations of words while being robust to parse errors that might occur, we introduce GraphMerge, which combines different parses into a single ensemble graph. Specifically, given a sentence $\{w_1, w_2, \ldots, w_n\}$ and M different dependency parses $G_1, \ldots, G_M$, GraphMerge takes the union of the edges from all parses, and constructs a single graph G as follows

$$G = (V, \{e|e = (w_i, w_j) \in \bigcup_{m=1}^{M} E_m\}) \quad (1)$$

where V is the shared set of nodes among all graphs[1] and $E_m (1 \leq m \leq M)$ is the set of edges in G_m (see the right side of Figure 2 for an example).

As a result, G contains all of the (directed) edges from all dependency trees, on top of which we can apply the same GNN models when a single dependency tree is used. Therefore, GraphMerge introduces virtually no computational overhead to existing GNN approaches, compared to traditional ensemble approaches where computational time and/or parameter count scale linearly in M. Note that the parsing time is not accounted for computational cost, because the dependency tree from three parsers could be obtained in parallel thus the running time is the same as the single parser.

What is more, the resulting graph G likely contains more edges from the gold parse which correctly captures the syntactic relation between words in the sentence, allowing the GNN to be robust to parse errors from any specific parser. Finally, since G contains more edges between words when parses

[1]This is true for dependency trees as long as parsers share the same tokenization as input.

differ than any single parse and reduces the diameter of the graph, it is also more likely that a shallower GNN model is enough to learn good representations, therefore avoiding over-parameterization and thus overfitting from stacking more GNN layers.

3.2 RGAT over Ensemble Graph

To learn node representations from ensemble graphs, we apply graph attention networks (GAT; Veličković et al., 2017). In one layer of GAT, the hidden representation of each node in the graph is computed by attending over its neighbors, with a multi-head self-attention mechanism. The representation for word i at the l-th layer of GAT can be obtained as follows

$$\boldsymbol{h}_i^{(l)} = \|_{k=1}^{K} \sigma\Big(\sum_{j \in N_i} \alpha_{ij}^k \boldsymbol{W}^k \boldsymbol{h}_i^{(l-1,k)}\Big) \quad (2)$$

Where K is the number of attention heads, N_i is the neighborhood of node i in the graph, and $\|$ the concatenation operation. $\boldsymbol{W}^k \in \mathbb{R}^{d_B \times d_h}$ represents the learnable weights in GAT and σ denotes ReLU activation function. α_{ij}^k is the attention score between node i and node j with head k.

Edge Types. To apply GAT to ensemble graphs, we first add reciprocal edges for each edge in the dependency tree, and label them with parent-to-children and children-to-parent types, respectively. This allows our model to retain the original syntactic relation between words in the sentence. We also follow previous work to add self loop to each node in the graph, which we differentiate from dependency edges by introducing a third edge type.

We adapt Relational GAT (RGAT) to capture this edge type information. Specifically, we encode the edge type information when computing the attention score between two nodes. We assign each edge type an embedding $\boldsymbol{e} \in \mathbb{R}^{d_h}$, incorporate it into attention score computation as follows

$$\alpha_{ij} = \frac{\exp(\sigma(\boldsymbol{a}\boldsymbol{W}(\boldsymbol{h}_i \| \boldsymbol{h}_j) + \boldsymbol{a}_e \boldsymbol{e}_{ij}))}{\sum_{v \in N_i} \exp(\sigma(\boldsymbol{a}\boldsymbol{W}(\boldsymbol{h}_i \| \boldsymbol{h}_v) + \boldsymbol{a}_e \boldsymbol{e}_{iv}))} \quad (3)$$

where $\boldsymbol{e}_{ij}$ is the representation of the type of the edge connecting nodes i and j. $\boldsymbol{a} \in \mathbb{R}^{d_h}$, $\boldsymbol{W} \in \mathbb{R}^{d_h \times 2d_h}$ and $\boldsymbol{a}_e \in \mathbb{R}^{d_h}$ are learnable matrices.

3.3 Sentiment Classification

We extract hidden representations from nodes that correspond to aspect terms in the last RGAT layer, and conduct average pooling to obtain $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$. Then we feed it into a two-layer MLP to calculate the final classification scores $\hat{\boldsymbol{y}}_s$:

$$\hat{\boldsymbol{y}}_s = \mathrm{softmax}(\boldsymbol{W}_2 \mathrm{ReLU}(\boldsymbol{W}_1 \boldsymbol{h}_t)) \quad (4)$$

where $\boldsymbol{W}_2 \in \mathbb{R}^{C \times d_{out}}$ and $\boldsymbol{W}_1 \in \mathbb{R}^{d_{out} \times d_h}$ denote learnable weight matrices, and C is the number of sentiment classes. We optimize the model to minimize the standard cross entropy loss function, and apply weight decay to model parameters.

Dataset	Positive		Neutral		Negative	
	Train	Test	Train	Test	Train	Test
Laptop	987	341	460	169	866	128
Restaurant	2164	728	633	196	805	196
Twitter	1561	173	3127	346	1560	173

Table 1: Statistics of the three benchmark datasets used in our experiments.

3.4 RGAT Input

The initial word node features for RGAT are obtained from a BERT encoder, with positional information from positional embeddings.

BERT Encoder. We use the pre-trained BERT base model as the encoder to obtain word representations. Specifically, we construct the input as "[CLS] + sentence + [SEP] + term + [SEP]" and feed it into BERT. This allows BERT to learn term-centric representations from the sentence during fine-tuning. To feed the resulting wordpiece-based representations into the word-based RGAT model, we average pool representations of subwords for each word to obtain $\boldsymbol{X}$, the raw input to RGAT.

Positional Encoding. Position information is beneficial for this task, especially when there are multiple aspect terms in one sentence, where it helps to locate opinion words relevant to an aspect term. Although the BERT encoder already takes the word position into consideration, it is dampened after layers of Transformers. Therefore, we explicitly encode the absolute position for each word and add it to the BERT output. Specifically, we add a trainable position embedding matrix to $\boldsymbol{X}$ before feeding the resulting representation into RGAT.

4 Experiments

4.1 Setup

Data & Processing. We evaluate our model on three datasets: Restaurant and Laptop reviews from

SemEval 2014 Task 4 (14Rest and 14Lap)[2] and ACL 14 Twitter dataset (Twitter) (Dong et al., 2014). We remove several examples with "conflict" sentiment polarity labels in the reviews. The statistics of these datasets are listed in Table 1. Following previous work, we report the accuracy and macro F1 scores for sentiment classification.

For dependency-based approaches, we tokenize sentences with Stanford CoreNLP (Manning et al., 2014), and then parse them with CoreNLP, Stanza (Qi et al., 2020), and the Berkeley neural parser (Kitaev and Klein, 2018). Since the Berkeley parser returns constituency parses, we further convert it into dependency parses using CoreNLP.

Baselines. We compare our GraphMerge model against published work on these benchmarks, including: **BERT-SPC** (Song et al., 2019) feeds the sentence and term pair into the BERT model and uses the BERT outputs for predictions; **AEN-BERT** (Song et al., 2019) uses BERT as the encoder and employs several attention layers. BERT + Dependency tree based models: **DGEDT-BERT** (Tang et al., 2020) proposes a mutual biaffine module to jointly consider the representations learnt from Transformer and the GNN model over the dependency tree; **R-GAT+BERT** (Wang et al., 2020b) reshapes and prunes the dependency tree to an aspect-oriented tree rooted at the aspect term, and then employs RGAT to encode the new tree for predictions. For fair comparison, we report the results of our GraphMerge model using the same data split (without a development set).

To understand the behavior of different models, we also implement several baseline models. In our experiments, we randomly sample 5% training data as held-out development set for hyper-parameter tuning, use the remaining 95% for training and present results of the average and standard deviation numbers from five runs of random initialization on the test set. We consider these baselines:

1. *BERT-baseline* which feeds the sentence-term pair into the BERT-base encoder and then applies a classifier with the representation of the aspect term token.
2. *GAT-baseline with Stanza* which employs a vanilla GAT model over single dependency tree obtained from Stanza without differentiating edge types. And the initial node features are the raw output of the BERT encoder.
3. *RGAT over single dependency trees*, where we apply RGAT models with parent-to-children and child-to-parent edge types over different dependency trees from the CoreNLP, Stanza, and Berkeley parsers. For a fair comparison to our GraphMerge model, the RGAT input comes from BERT encoder plus position embeddings.
4. Two ensemble models to take advantage of multiple dependency trees, including a *Label-Ensemble* model which takes the majority vote from three models each trained on one kind of parses, and a *Feature-Ensemble* model which applies three sets of RGAT parameters, one for each parse, on top of the BERT encoder with their output features concatenated. These models have more parameters and are more computationally expensive compared to the GraphMerge model when operating on the same parses.

Parameter Setting. We use Pytorch (Paszke et al., 2019) to implement our models. The GAT implementation is based on Deep Graph Library (Wang et al., 2019). During training, we set the learning rate = 10^{-5}, batch size = 4. We use dev data to select the hidden dimension d_h for GAT/RGAT from $\{64, 128, 256\}$, the head number in the multi-head self-attention from $\{4, 8\}$, and GAT/RGAT layer from $\{2, 3, 4\}$. The 2-layer GAT/RGAT models turn out to be the best based on the dev set. We apply dropout (Srivastava et al., 2014) and select the best setting from the dropout rate range = $[0.1, 0.3]$. We set the weight of L2 regularization as 10^{-6}. We train the model up to 5 epochs.[3]

4.2 Results

We first compare our model to previous work following the evaluation protocol in previous work, and report results in Table 2. As we can see, the GraphMerge model achieves best performances on all three datasets. On the Laptop dataset, the GraphMerge model further outperforms baselines by at least 1.42 accuracy and 2.34 Macro-F1 respectively.

Table 3 shows performance comparisons of the GraphMerge model with other baselines in terms of accuracy and Macro-F1. We observe that:

Syntax information benefits aspect-level sentiment classification. All GAT and RGAT models based on dependency trees outperform *BERT-*

[2]https://alt.qcri.org/semeval2014/task4/

[3]Our code will be released at the time of publication.

Category	Model	14Rest		14Lap		Twitter	
		Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
BERT	BERT-SPC (Song et al., 2019)	84.46	76.98	78.99	75.03	73.55	72.14
	AEN-BERT (Song et al., 2019)	83.12	73.76	79.93	76.31	74.71	73.13
BERT+DT*	DGEDT-BERT (Tang et al., 2020)	86.3	80.0	79.8	75.6	77.9	75.4
BERT+RDT$^\diamond$	R-GAT+BERT (Wang et al., 2020b)	86.60	81.35	78.21	74.07	76.15	74.88
Ours	**GraphMerge**	**87.32**	**81.95**	**81.35**	**78.65**	**78.18**	**76.52**

$\star$ DT: Dependency Tree; $\diamond$ RDT: Reshaped Dependency Tree.

Table 2: Comparison of our GraphMerge model to different published numbers on three datasets, with the same setup of train and test data – no dev data. The bold text indicates the best results.

Model	14Rest		14Lap		Twitter	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
BERT-baseline	83.43 $\pm$ 0.52	74.94 $\pm$ 1.37	77.34 $\pm$ 0.90	72.77 $\pm$ 1.96	73.47 $\pm$ 0.89	72.63 $\pm$ 0.82
GAT-baseline with Stanza	84.29 $\pm$ 0.30	75.75 $\pm$ 1.11	77.84 $\pm$ 0.27	74.0 $\pm$ 0.55	73.82 $\pm$ 0.70	72.72 $\pm$ 0.42
RGAT with Stanza	84.53 $\pm$ 0.66	77.29 $\pm$ 1.42	77.99 $\pm$ 0.62	74.35 $\pm$ 0.60	73.99 $\pm$ 0.48	72.76 $\pm$ 0.33
RGAT with Berkeley	84.41 $\pm$ 0.86	76.63 $\pm$ 1.38	78.09 $\pm$ 1.24	73.65 $\pm$ 1.76	74.07 $\pm$ 0.67	72.65 $\pm$ 0.74
RGAT with CoreNLP	83.86 $\pm$ 0.32	76.03 $\pm$ 0.88	78.12 $\pm$ 1.02	73.86 $\pm$ 1.72	73.96 $\pm$ 0.93	72.83 $\pm$ 0.95
Label-Ensemble	84.68 $\pm$ 0.95	77.21 $\pm$ 1.54	78.40 $\pm$ 1.51	74.36 $\pm$2.45	74.59 $\pm$ 0.46	73.51 $\pm$ 0.43
Feature-Ensemble	84.64 $\pm$ 0.77	77.06 $\pm$ 1.45	78.68 $\pm$ 0.69	74.80 $\pm$ 0.92	74.62 $\pm$ 0.76	73.61 $\pm$ 0.73
GraphMerge	**85.16** $\pm$ 0.53	**77.91** $\pm$ 0.87	**80.00** $\pm$ 0.63	**76.50** $\pm$ 0.64	**74.74** $\pm$ 0.93	**73.66** $\pm$ 0.88

Table 3: Comparison of our GraphMerge model to different baselines on three datasets, with 5% dev data set aside. The bold text indicates the best results.

baseline on all three datasets. This demonstrates that leveraging syntax structure information is beneficial to this task.

Ensemble models benefit from multiple parses. The *Label-Ensemble*, *Feature-Ensemble*, and GraphMerge models achieve better performance compared to their single dependency tree counterparts. This shows that ensemble models benefit from the presence of different parses and thus less sensitive to parse errors from any single parser.

GraphMerge achieves the best performance overall. Our proposed *GraphMerge* model not only shows consistent improvements over all single dependency tree models, but also surpasses the other two ensemble models without additional parameters or computational overhead, when compared to the single-tree models. Note that although in this specific task, the best results are achieved using three trees in GraphMerge. The number of trees for ensemble depends on different tasks and datasets.

4.3 Model Analysis

We analyze the proposed GraphMerge model from two perspectives: an ablative analysis of model components and an analysis of the change in the dependency graphs after GraphMerge is applied.

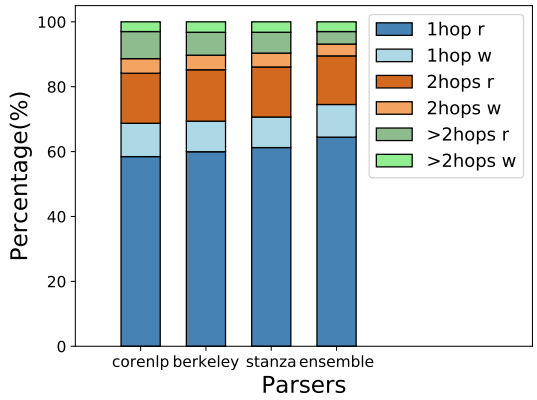
4.3.1 Ablation Study

Model components. We conduct ablation studies of our modeling for edge type and position information in Table 4. We observe that: (1) On three datasets, ablating the edge type degrades the performances. It indicates that the syntactic dependency information in original dependency trees is important. Differentiating edges in the ensemble graph provides more guidance to the model about selecting useful connections among nodes. (2) Removing the position embeddings hurts the performances as well. Although the BERT encoder already incorporates position information at its input, this information is dampened over the layers of Transformers. Emphasizing sequence order again before applying RGAT benefits the task.

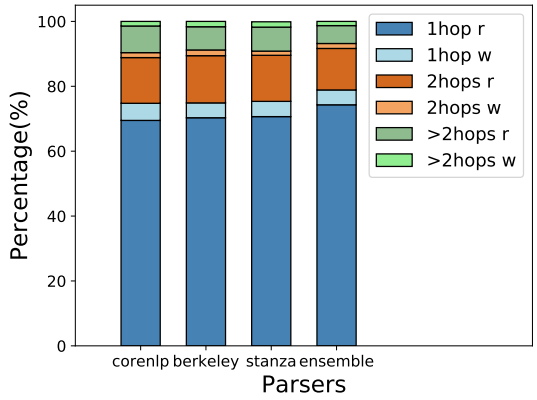
Edge Union vs. Edge Intersection. While GraphMerge keeps all edges from different dependency parsing trees for the RGAT model to learn to use, this could also result in too much structural noise and adversely impact performance. We therefore compare GraphMerge to edge intersection, which only retains edges that shared by all individual trees when constructing the ensemble graph, which can be thought of distilling syntactic

Model	14Rest		14Lap		Twitter	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
GraphMerge	**85.16** $\pm$ 0.53	**77.91** $\pm$ 0.87	**80.00** $\pm$ 0.63	**76.50** $\pm$ 0.64	**74.74** $\pm$ 0.93	**73.66** $\pm$ 0.88
- Edge type	84.25 $\pm$ 0.59	76.15 $\pm$ 1.24	78.65 $\pm$ 0.51	74.76 $\pm$ 0.71	74.37 $\pm$ 1.08	73.25 $\pm$ 0.85
- Position	84.36 $\pm$ 0.36	75.92 $\pm$ 1.18	78.37 $\pm$ 0.31	74.51 $\pm$ 0.48	74.28 $\pm$ 1.39	73.34 $\pm$ 1.35
- (Edge type + Position)	84.16 $\pm$ 0.31	75.38 $\pm$ 0.69	78.09 $\pm$ 0.27	74.29 $\pm$ 0.64	73.41 $\pm$ 0.63	72.52 $\pm$ 0.62
Edge Intersection	84.59 $\pm$ 0.61	77.06 $\pm$ 1.07	78.65 $\pm$ 0.94	74.86 $\pm$ 1.42	74.68 $\pm$ 0.83	73.45 $\pm$ 0.73

Table 4: Ablation study of the GraphMerge model over three datasets. We report the average and standard deviation over five runs, where Edge Intersection means taking intersection of edges from multiple dependency trees.



(a) Accuracy w.r.t different hop number on 14Lap.



(b) Accuracy w.r.t different hop number on 14Rest.

Figure 3: Hop analysis on 14Lap and 14Rest, where "r" and "w" denote predictions that are right or wrong, respectively.

information that an ensemble parser is confident about. We observe from the last row in Table 4 that edge intersection strategy underperforms GraphMerge on average accuracy and Marco-F1. We postulate that this is because edge intersection over-prunes edges in the ensemble graph and might introduce more disjoint connected components where parsers disagree, which the RGAT model cannot easily recover from.

	Aspect Terms	Opinion Words	Coverage
Laptop	638	467	73.20%
Restaurant	1120	852	76.07%

Table 5: Statistics of the Opinion datasets. Aspect Terms denotes as the total number of aspect terms. Opinion Words represents total number of terms that have labeled opinion words. Coverage is the proportion of terms with labeled opinion words.

4.3.2 Graph Structure Analysis

Effect of GraphMerge on Graph Structure. To better understand the effect of GraphMerge on dependency graphs, we conduct statistical analysis on the test set of 14Lap and 14Rest. Specifically, we are interested in the change in the shortest distance between the aspect term and its opinion words on the dependency graphs. For this analysis, we use the test sets with opinion words labeled by Fan et al. (2019) (see Table 5 for dataset statistics).

We summarize analysis results in Figure 3. We observe that: (1) Compared with single dependency tree, the ensemble graph effectively increases the number of one-hop and two-hops cases, meaning the overall distance between the term and opinion words is shortened on both datasets. (2) Shorter distance between the term and opinion words correlates with better performance. With the ensemble graph, the accuracy of one-hop and two-hops cases beats all single dependency tree models. These observations suggest that the ensemble graph from GraphMerge introduces important connectivity to help alleviate overparameterization from stacking RGAT layers, and that the RGAT model is able to make use of the diversity of edges in the resulting graph to improve classification performance.

Note that although shortening distance correlates with improved results, it does not mean that the closer distance is sufficient for better performance. This is because although the BERT model can be seen as a GAT over a fully-connected graph where

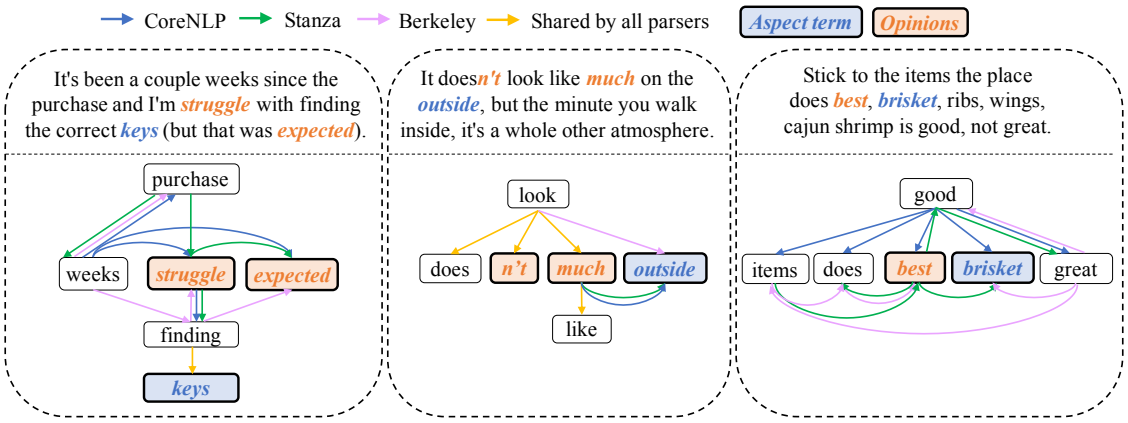


Figure 4: Examples of partial dependency trees on which the single dependency tree models make wrong prediction, but the GraphMerge model makes correct prediction.

a word is reachable for all other context words within one hop (Wang et al., 2020a), the *BERT-baseline* model performs worse than dependency-based models. Therefore, encoding the syntactic structure information in dependency trees is crucial for this task. Our GraphMerge model achieves the best results by shortening the graph distance between the aspect term and opinion words with syntactic information.

Case Study. To gain more insight into the GraphMerge model's behaviour, we find several examples and visualize their dependency trees from three parsers (Figure 4). Due to the space limit, we only show partial dependency trees that contain essential aspect terms and opinion words. These examples are selected from cases that all single dependency tree RGAT models predict incorrectly, but the *GraphMerge* model predicts correctly.

We observe that in general, the three parsers do not agree in the neighborhood around the aspect term and opinion words in these sentences. As a result, GraphMerge tends to shorten the distance between the aspect term and the opinion words on the resulting graph. For instance, for all examples in Figure 4, the shortest distances between the aspect term and the opinion words are no more than two in the ensemble graphs, while they vary from 2 to 4 in the original parse trees. This could allow the RGAT model to capture the relation between the words without an exessive amount of layers, thus avoiding overfitting.

On the other hand, we observe that the resulting ensemble graph from GraphMerge is more likely to contain the gold parse for the words in question. For instance, in the first two examples, the gold parse for the words visualized in the figure can be found in the ensemble graph (despite no individual parser predicting it in the first example); the third example also has a higher recall of gold parse edges than each parser despite being difficult to parse. This offers the RGAT model with the correct semantic relationship between these words in more examples during training and evaluation, which is often not accessible with those single parse trees.

Dataset	Positive	Neutral	Negative	Total
Laptop	883	407	587	1877
Restaurant	1953	473	1104	3530

Table 6: Statistics of robustness testing data ARTS.

Aspect Robustness. To study the aspect robustness of the GraphMerge model, we test our model on the Aspect Robustness Test Set (ARTS) datasets proposed by Xing et al. (2020) (see Table 6 for statistics). The datasets enrich the original 14Lap and 14Rest datasets following three strategies: reverse the sentiment of the aspect term; reverse the sentiment of the non-target terms with originally the same sentiment as target term; generate more non-target aspect terms that have opposite sentiment polarities to the target one. They propose a novel metric, Aspect Robustness Score (ARS), that counts the correct classification of the source example and all its variations generated by the above three strategies as one unit of correctness.

We compare three single dependency tree models with the GraphMerge model in Table 7. We directly evaluate the models trained on the original SemEval datasets on ARTS without further tuning. The results indicate that the GraphMerge model shows better aspect robustness than single dependency tree and BERT models.

Model	14Rest	14Lap
	Acc → ARS	Acc → ARS
BERT	83.04 → 54.82 (↓28.22)	77.59 → 50.94 (↓26.65)
RGAT with Berkeley	84.41 → 56.54 (↓27.87)	78.09 → 51.37 (↓26.72)
RGAT with CoreNLP	83.86 → 55.76 (↓28.10)	78.12 → 52.27 (↓25.85)
RGAT with Stanza	84.53 → 56.34 (↓28.19)	77.99 → 51.20 (↓26.79)
GraphMerge	85.16 → **57.46** (↓27.70)	80.00 → **52.90** (↓27.10)

Table 7: Comparison of GraphMerge model to the single dependency tree based models and BERT model in terms of Aspect Robustness Score (ARS), on the ARTS dataset (Xing et al., 2020).

5 Conclusion

We propose a simple yet effective graph-ensemble technique, GraphMerge, to combine multiple dependency trees for aspect-level sentiment analysis. By taking the union of edges from different parsers, GraphMerge allows graph neural model to be robust to parse errors without additional parameters or computational cost. With different edge types to capture the original syntactic dependency in parse trees, our model outperforms previous state-of-the-art models, single-parse models, as well as traditional ensemble models on three aspect-level sentiment classification benchmark datasets.

Acknowledgement

This work was supported by the National Key R&D Program of China under Grant No.2018YFB2100802 and No.2020AAA108600.

References

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. 2019. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, page pages 4171–4186.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 49–54.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3433–3442.

Zhifang Fan, Zhen Wu, Xin-Yu Dai, Shujian Huang, and Jiajun Chen. 2019. Target-oriented opinion words extraction with target-fused neural sequence labeling. 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)*, pages 2509–2518, Minneapolis, Minnesota. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1121–1131.

Binxuan Huang and Kathleen M Carley. 2019. Syntax-aware aspect level sentiment classification with graph attention 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)*, pages 5472–5480.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Lishuang Li, Yang Liu, and AnQiao Zhou. 2018a. Hierarchical attention based position-aware network for aspect-level sentiment analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 181–189.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018b. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018c. Transformation networks for target-oriented sentiment classification. *arXiv preprint arXiv:1805.01086*.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2019. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. *arXiv preprint arXiv:1908.11860*.

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2021. Do syntax trees help pre-trained transformers extract information?

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 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)*, pages 5683–5692.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 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*, pages 6578–6588, Online. Association for Computational Linguistics.

Zhaopeng Tu, Wenbin Jiang, Qun Liu, and Shouxun Lin. 2012. Dependency forest for sentiment analysis. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 69–77. Springer.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Bailin Wang and Wei Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. 2020a. Direct multi-hop attention based graph neural network. *arXiv preprint arXiv:2009.14332*.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020b. Relational graph attention network for aspect-based sentiment analysis. *arXiv preprint arXiv:2004.12362*.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander Smola, and Zheng Zhang. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967.

Xiaoyu Xing, Zhijing Jin, Di Jin, Bingning Wang, Qi Zhang, and Xuan-Jing Huang. 2020. Tasty burgers, soggy fries: Probing aspect robustness in aspect-based sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3594–3605.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232*.

Chen Zhang, Qiuchi Li, and Dawei Song. 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)*, pages 4560–4570.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. *EMNLP*.

Shiliang Zheng and Rui Xia. 2018. Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention. *arXiv preprint arXiv:1802.00892*.