Inducing Target-specific Latent Structures for Aspect Sentiment Classification

Chenhua Chen, Zhiyang Teng and Yue Zhang
School of Engineering, Westlake University, China
Institute of Advanced Technology, Westlake Institute for Advanced Study
{chenchenhua, tengzhiyang}@westlake.edu.cn, yue.zhang@wias.org.cn

Abstract

Aspect-level sentiment analysis aims to recognize the sentiment polarity of an aspect or a target in a comment. Recently, graph convolutional networks based on linguistic dependency trees have been studied for this task. However, the dependency parsing accuracy of commercial product comments or tweets might be unsatisfactory. To tackle this problem, we associate linguistic dependency trees with automatically induced aspect-specific graphs. We propose gating mechanisms to dynamically combine information from word dependency graphs and latent graphs which are learned by self-attention networks. Our model can complement supervised syntactic features with latent semantic dependencies. Experimental results on five benchmarks show the effectiveness of our proposed latent models, giving significantly better results than models without using latent graphs.

1 Introduction

Aspect-level sentiment analysis aims to classify the sentiment polarities towards specific aspect terms in a given sentence (Jiang et al., 2011; Dong et al., 2014; Vo and Zhang, 2015). Aspects are also called opinion targets, which can typically be product or service features in customer reviews. For example, in the user comment “The environment is romantic, but the food is horrible”, the sentiments of the two aspects “environment” and “food” are positive and negative, respectively. The main challenge of aspect-level sentiment analysis is to effectively model the interaction between the aspect and its surrounding contexts. For example, identifying that “romantic” instead of “horrible” as the opinion word is the key to correctly classifying the sentiment of “environment”.

Recently, graph convolutional networks (GCNs, (Kipf and Welling, 2017)) over dependency trees (Marcheggiani and Titov, 2017; Zhang et al., 2019; Sun et al., 2019b; Wang et al., 2020) have received much research attention. They have been shown to be more effective to learn aspect-specific representations than traditional sentence encoders without considering graph structures (Tang et al., 2016a,b; Liu and Zhang, 2017; Li et al., 2018a). Intuitively, dependency trees allow a model to better represent the correlation between aspect terms and their relevant opinion words. However, dependency syntax suffers from two potential limitations. First, dependency parsing accuracies can be relatively low on noisy texts such as tweets, blogs and review comments, which are the main sources of aspect-level sentiment data. Second, dependency syntax may not be the most effective structure for capturing interaction between aspect terms and opinion words. Take Figure 1(a) for example. The aspect term “manager” is syntactically related to “not apologetic” through complained → manager and complained → not apologetic, though semantically they are directly related.

One intuitive solution to the aforementioned problems is to automatically induce semantic structures during the optimization process for sentiment classification. To this end, existing work has inves-
tigated latent structures sentence-level sentiment classification (Yogatama et al., 2016; Kim et al., 2017; Choi et al., 2017; Zhang et al., 2018; Corro and Titov, 2019), but no existing work has considered aspect-level sentiment classification. For the aspect-level task, a different structure should be ideally learned for each aspect. As shown in Figure 1(b) and 1(c), when given the sentence “the portions are small but being that the food was so good makes up for that.”, the ideal structure for the aspects “portions” and “food” consist of links relevant to the terms and their opinion words only, without introducing additional information.

We empirically investigate three different methods to induce semantic dependencies, including attention (Vaswani et al., 2017), sparse attention (Correia et al., 2019) and hard Kumar discrete structures (Bastings et al., 2019). In particular, attention has been used as soft alignment structures for tasks such as machine translation (Bahdanau et al., 2014), and sparse attention has been used for text generation (Martins et al., 2020). The Hard Kumaraswamy distribution has been used to induce discrete structures with full differentiability (Bastings et al., 2019). We build a unified self-attentive-network (SAN) framework (Vaswani et al., 2017) for investigating the three structure induction methods, using a graph convolutional network on top of the induced aspect-specific structure for aspect level sentiment classification. In addition, to exploit mutual benefit with dependency syntax, we further investigate a novel gate mechanism for merging multiple tree structures during GCN encoding.

Experiments on five benchmarks including Twitter, laptop and restaurant comments show the effectiveness of our proposed latent variable models. Our final methods give state-of-the-art in the literature, achieving the significantly better results than models without using latent graphs. To our knowledge, we are the first to investigate automatically inducing tree structures for targeted sentiment classification. We will release our code at http://annoymized_url.

2 Related Work

Aspect-level sentiment analysis Aspect-level sentiment analysis includes three main sub-tasks, namely aspect term sentiment classification (ATSC) (Jiang et al., 2011; Dong et al., 2014), aspect category sentiment classification (ACSC) (Jo and Oh, 2011; Pontiki et al., 2015, 2016) and aspect-term or opinion words extractions (Li et al., 2018b; Fan et al., 2019; Wan et al., 2020). In this paper, we focus on ATSC. To model relationships between the aspect terms and the context words, Vo and Zhang (2015) designed target-aware pooling functions to extract discriminative contexts. Tang et al. (2016a) modeled the interaction of targets and context words by using target-dependent LSTMs. Tang et al. (2016b) used multi-hop attention and memory networks to correlate the aspect with its opinion words. Attention networks are further explored by sequent work (Ma et al., 2017; Liu and Zhang, 2017). Li et al. (2018a) used target-specific transformation networks to learn target-specific word representations. Liang et al. (2019) used aspect-guided recurrent transition networks to generate aspect-specific sentence representations. Sun et al. (2019a) constructed aspect related auxiliary sentences as inputs to BERT (Devlin et al., 2019) for strong contextual encoders. Xu et al. (2019) proposed BERT-based post training for enhancing domain-specific contextual representations for aspect sentiment analysis.

Recently, there is a line of work considering dependency tree information for ATSC. Lin et al. (2019) proposed deep mask memory network based on dependency trees. Zhang et al. (2019) and Sun et al. (2019b) encoded dependency tree using GCNs for aspect-level sentiment analysis. Zhao et al. (2019) used GCNs to model fully connected graphs between aspect terms, so that all targets can be classified using a shared representation. Huang and Carley (2019) proposed graph attention networks based on dependency trees for modeling structural relations. Wang et al. (2020) use relational graph attention networks to incorporate the dependency edge type information, and construct aspect-specific graph structures by heuristically reshaping dependency trees.

Latent graph induction Latent graphs can be induced to learn task-specific structures by end-to-end models jointly with downstream tasks. Kim et al. (2017) proposed structural attention networks to introduce latent dependency graphs as intermediate layers for neural encoders. Niculae et al. (2018) used SparseMAP to obtain a sparse distribution over latent dependency trees. Peng et al. (2018) implemented a differentiable proxy to the argmax operator over latent dependency trees, which can be regarded as a special case of introducing spar-


3 Model

The overall model structure is shown in Figure 1. The model consists of four main components, including a sequence encoder layer for the input sentence, a structural representation layer that learns a latent induced structure $A$, a GCN network that represents the latent structure and an aspect oriented classification layer. Below we discuss each component in detail in the bottom-up direction.

### 3.1 Sentence Encoder

We separately explore two sentence encoders including a bidirectional long short-term memory networks (BiLSTM) encoder and a BERT encoder. Given an input sentence $s = w_1 w_2 \ldots w_n$. We first obtain the embedding vector $x_i$ of each $w_i$ using a lookup table $E \in \mathbb{R}^{V \times d_w}$ (where $V$ is the vocabulary size and $d_w$ is the dimension of word vectors) and then use a standard BiLSTM encoder to obtain the contextual vectors of the input sentence. For the BERT encoder, we follow the conventional practice by feeding the input “[CLS] $w_1 w_2 \ldots w_n$ [SEP] $w_f w_{f+1} \ldots w_e$” to BERT to obtain aspect specific representations, where $c = w_f w_{f+1} \ldots w_e$ is the corresponding aspect sequence in $s$. Since BERT uses a subword encoding mechanism (Sennrich et al., 2015), we apply average pooling over the subword-level representations to obtain the corresponding word-level representations. The output vectors from the sentence encoder are denoted as $c^0_i$ for each $w_i$.

**Aspect Mask** In order to make the encoder learn aspect-specific representations, we use distance-based masks on the word representation $c^0_i$. Formally, given an aspect $w_f w_{f+1} \ldots w_e$, the masked $c^0_i$ is $H_i = m_i c^0_i$, where $m_i$ is given by,

$$ m_i = \begin{cases} 
1 - \frac{i - f}{e - f} & 1 \leq i < f, \\
0 & f \leq i \leq e, \\
1 - \frac{n - e}{n - f} & e < i \leq n.
\end{cases} \quad (1)
$$

In this way, the more similar the context words are to the aspect, the higher their weights are. We denote the sentence representation as $H = [H_0, H_1, \ldots, H_n]$, which is used for inducing corresponding word-level representations.

### 3.2 Dependency Tree Representation

Given a sentence $s = w_1 w_2 \ldots w_n$ and the corresponding dependency tree $t$ over $s$ (obtained using parser), an undirected graph $G$ is built by taking each word as a node and representing head-dependent relations in $t$ as edges. Each head-dependent arc is converted into two undirected edges. In addition, self loops are included for each word. Formally, the adjacent matrix $A_{dep}$ is given by

$$ A_{dep}[i, j] = \begin{cases} 
1 & \text{if } i \rightarrow j \text{ or } i \leftarrow j, \\
1 & \text{if } i = j, \\
0 & \text{otherwise}.
\end{cases} \quad (2)
$$

$A_{dep}$ represents the syntactic dependencies between word pairs.
3.3 Latent Graph

We propose to learn latent graphs $A_{lat}$ for each aspect, investigating three methods, namely self-attention, sparse self-attention and hard Kuma.

Self-attention Based Latent Graph

Self-attention networks (SANs) compute similarity scores between two arbitrary nodes in a graph. Formally, given a sentence representation $H$, the similarity score $\alpha_{ij}$ can be regarded as the interaction strength between node $i$ and node $j$. $A_{lat}$ is given by

$$ A_{lat} = \text{softmax} \left( \frac{(QW_q)(KW_k)^T}{\sqrt{d}} \right) $$

where $Q$ and $K$ are two copies of $H$, representing the query and key vectors, respectively. $W_q \in \mathbb{R}^{d \times d}$ and $W_k \in \mathbb{R}^{d \times d}$ are model parameters. The denominator $\sqrt{d}$ is a scale constant for controlling the magnitude of the dot-product operation. The softmax function normalizes the similarity scores by column so that the sum of each row in $A_{lat}$ equals to 1.

Multi-head SANs partition the graph representation $H$ into multiple non-overlapped heads $H = [H_1, H_2, \ldots, H_K]$, where $K$ is the number of heads and $H_i \in \mathbb{R}^{n \times \frac{d}{K}}$. For the $i$-th head, Eq 3 is independently applied to generate $A_{head}^{i}$. The final latent graph averages the latent graphs of all heads,

$$ A_{lat} = \frac{\sum_{k=1}^{K} A_{head}^{k}}{K}. $$

Sparse Self-attention Based Latent Graph

SANs learn a fully connected latent graph, where dense attention weight can bring noise from irrelevant context. To address this issue, sparse SANs potentially enables each node to attend to highly relevant contextual nodes. To achieve this goal, we replace the softmax operation in Eq 3 with the 1.5-entmax function (Niculae et al., 2018; Peters et al., 2019; Correia et al., 2019), which can project a real-valued vector into a sparse probability simplex. Formally,

$$ A_{lat} = 1.5-\text{entmax} \left( \frac{(QW_q)(KW_k)^T}{\sqrt{d}} \right). $$

Similar to Eq 4, multi-head SANs are used for the sparse latent graph learning.

HardKuma Based Latent Graph

HardKuma (Bastings et al., 2019) is a method which can produce stochastic graphs by sampling. Suppose that each edge $\alpha_{ij} \in [0, 1]$ between nodes $i$ and $j$ is a stochastic random variable and $\alpha_{ij} \sim \text{HardKuma}(a, b, l, r)$, where $a > 0$ and $b > 0$ are parameters to control the shape of a Hard Kumaraswamy probability distribution, $l < 0$ and $r > 1$ define the supporting open interval $(l, r)$. A sample of $\alpha_{ij}$ can be obtained by gradient reparameterization tricks (Kingma and Welling, 2013; Jang et al., 2016),

$$ s_1 = F_{\text{Kuma}}^{-1}(u; a, b), \ s_2 = l + (r - l) \cdot s_1, \ z = \min(1, \max(0, s_2)),$$

where $u$ is a uniform random variable and $u \sim U(0, 1)$ which replaces the HardKuma sample, $F_{\text{Kuma}}^{-1}$ is the inverse c.d.f. of the Kumaraswamy distribution, $s_1$ is a sample of the Kumaraswamy distribution, $s_2$ is the stretched sample for the supporting interval after shift and scale operations. $s_2$ is converted to $z$ by a hard-sigmoid function, which can ensure the value of $z$ falls into $[0, 1]$. $z$ is differentiable with respect to the shape parameters $a$ and $b$. Denote the shape parameters for all edges as $a$ and $b$. With reparameterization, the sampling is independent of the model and the main goal is to represent $a$ and $b$ using neural networks. Specifically, $a$ and $b$ can be calculated by SANs. Formally, $a \in \mathbb{R}^{n \times n}$ for the whole graph is given by

$$ H_a = \text{MHSAN}(H, H, H), \ C_a = \text{LN}(\text{FFN}(H_a) + H_a) $$

$$ s_a = C_a C_a^T, \ n_a = \frac{s_a - \text{mean}(s_a)}{\text{std}(s_a)}, \ a = \text{softplus}(n_a) $$

$$ A_{lat} \sim \text{HardKuma}(a, b, l, r), $$

where MHSAN, LN and FFN denote multi-head self-attention networks, layer normalizations and position-wise feed-forward networks, respectively. In our model, the particular networks of MHSAN, LN and FFN are taken from Transformers (Vaswani et al., 2017). $b$ is defined in a similar way, but with different parameters. Here $H_a$ is the initial result of MHSAN, $C_a$ considers residual connections and feature transformations, $s_a$ is the initial similarity score calculated by self attentions, $n_a$ is the normalized similarity scores and $a$ is ensured to be non-negative by applying the softplus activation function over $n_a$.

3.4 Graph Convolutional Networks

Graph convolutional networks (GCNs) (Kipf and Welling, 2017) encode graph-structured data with convolution operations. The representation of each
node $v$ in a graph $G$ is aggregated from its neighbors. Suppose that the node set is $V = \{v_i\}_{i=1}^n$, where $n$ is the number of nodes and the graph is $G = \{V, A\}$. $A \in \mathbb{R}^{n \times n}$ is the adjacent matrix between nodes. Let the representation vector of $v_i$ at the $l$-th layer be $h_i^l$ and $h_i^0 \in \mathbb{R}^d$, where $d$ is the node vector dimension. The whole graph representation of the $l$-th layer $H^l$ is the concatenation of all the node vectors of this layer, namely $H^l = \{h_1^l, h_2^l, ..., h_n^l\}$ and $H^l \in \mathbb{R}^{n \times d}$. The graph convolution for $H^l$ is given by:

$$H^l = \rho(AH^{l-1}W^l + b^l),$$

where $W^l \in \mathbb{R}^{d \times d}$ and $b^l \in \mathbb{R}^d$ are model parameters for the $l$-th layer. $\rho$ is an activation function over the input graph representation $H^{l-1}$ and typically set to be the ReLU function. The initial input $H^0$ is the sentence representation $H$.

### 3.5 Gated Combination

Given two graphs $A_{dep}$ and $A_{lat}$, we design gating mechanisms to combine the strengths of both. Formally, suppose that the input graph representation is $H_{in}$, the graph convolution weight matrix and bias are $W$ and $b$ respectively, we propose a gated GCN to output $H_{out}$ by considering both $A_{dep}$ and $A_{lat}$,

$$
\begin{align*}
I_{dep} &= A_{dep}H_{in}W, \\
I_{lat} &= A_{lat}H_{in}W, \\
I_{com} &= (1 - \lambda g) \odot I_{dep} + \lambda g \odot I_{lat}, \\
H_{out} &= \rho(I_{com} + b),
\end{align*}
$$

where $g$ is a gating function learned automatically from data and $0 \leq \lambda \leq 1$ is a hyper-parameter for prior knowledge. The graph convolutional matrix $W$ is the same for $A_{dep}$ and $A_{lat}$, which suggests that our model does not introduce any additional parameters. $AHW$ in Eq 8 can be replaced with $I_{com}$, which is a gated combination of $A_{dep}HW$ and $A_{lat}HW$. This combination equals that we first merge $A_{dep}$ and $A_{lat}$ into a single graph $A_{com}$ using dynamic gating mechanisms and then directly use $A_{com}$ as $A$ in Eq 7 to obtain the graph representations.

**Gated GCN Blocks** In practice, we stack multiple GCN layers. The input to the first block is the aspect-aware sentence representation $H^0$. We repeat the GCN layer $N$ times. A highway network is used to combine the feature representations in adjacent layers. Formally, given the input representation of the $l$-th block $H^{l-1}$, the output of the $l$-th block $H^l$ is given by,

$$
\begin{align*}
g_l &= \sigma(H^{l-1}), \\
H_{com} &= \text{gatedcombine}(H^{l-1}, A_{dep}, A_{lat}), \\
H^l &= g_l \odot H_{com} + (1 - g_l) \odot H^{l-1},
\end{align*}
$$

where gatedcombine is the GCN function defined in Eq 8. We apply the highway gate to all the GCN blocks except for the last one.

### 3.6 Sentiment Classifier

**Aspect-specific attention** Based on the output of the last GCN block $H^N$, we obtain the representations for the aspect $w_1 w_2 w_{f+1} ... w_w$ using $H_1^N, H_2^N, ..., H_w^N$. The final aspect-specific feature representation $z$ is given by an attention network over the sentence representation vectors $ce_i^0$

$$
\gamma_i = \sum_{i=f}^we_i^0 H_i^N, \quad \alpha = \text{softmax}(\gamma), \quad z = \alpha C,
$$

where $C = [ce_1^0, ce_2^0, ..., ce_w^0]$ is the contextualized representations produced by the sentence coder, $\gamma_i$ is the attention scores of the $i$-th context word with respect to the aspect term, $\alpha$ is the normalized attention scores and $z$ is the final aspect-specific representation.

**Softmax classifier** The aspect-specific representation vector $z$ is then used to calculate the sentiment score by a linear transformation. A softmax classifier is used to predict the probability of the sentimental class according to the learned sentiment scores. Formally,

$$
p = \text{softmax}(W_oz + b_o),
$$

where $W_o$ and $b_o$ are model parameters and $p$ is the predicted sentiment probability distribution.

### 3.7 Training

The classifier is trained by maximizing the negative log-likelihood of the training samples. Formally, the training objective is given by

$$
L(\theta) = - \sum_{i=1}^{[D]} \sum_{c \in x_i} \log p_{y_i} + \lambda \frac{1}{2} ||\theta||^2,
$$

where $[D]$ is the number of training data, $\theta$ is the set of model parameters, $\lambda$ is a regularization hyperparameter, $y_i$ is the training label of the aspect $c$ in the $i$-th data $x_i$ and $p_{y_i}$ is the aspect classification probability for $c$, which is given by Eq 10.
Table 1: Dataset Statistics.

<table>
<thead>
<tr>
<th>Model</th>
<th>depGCN</th>
<th>sanGCN</th>
<th>sparseGCN</th>
<th>kumaGCN full</th>
<th>kumaGCN -latent</th>
<th>kumaGCN -dep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>88.99</td>
<td>88.64</td>
<td>89.29</td>
<td>89.39</td>
<td>89.12</td>
<td>89.23</td>
</tr>
<tr>
<td>F1</td>
<td>67.48</td>
<td>69.37</td>
<td>72.14</td>
<td>73.19</td>
<td>70.89</td>
<td>72.04</td>
</tr>
</tbody>
</table>

Table 2: Model performances on REST16.

4 Experiments

We conduct experiments on five benchmark datasets for aspect-level sentiment analysis, including twitter posts (TWITTER) from (Dong et al., 2014), laptop comments (LAP14) provided by (Pontiki et al., 2014), restaurant reviews of SemEval 2014 task 4 (REST14; Pontiki et al. 2014), SemEval 2015 task 12 (REST15; Pontiki et al. 2015) and SemEval 2016 task 5 (REST16; Pontiki et al. 2016). We pre-process these dataset in the same way as presented in (Tang et al., 2016b) and (Zhang et al., 2019). Table 1 shows the statistics.

Settings. We initialize word embeddings with 300-dimensional pretrained GloVe (Pennington et al., 2014) embeddings. The number of gated GCN blocks is 2. The head number is 8. The hidden dimension is 300. We parse the data using Stanza (Qi et al., 2020). No dependency labels are used. For the other settings, we follow (Zhang et al., 2019).

Following previous conventions, we repeat each model three times and average the results, reports accuracy (Acc.) and macro-F1 (F1) scores.

4.1 Development Results

Effect of latent graphs Table 2 shows the performances of on REST16. We enhance dependency tree based graphs with self-attention based latent graph models (sanGCN), sparse self-attention based latent graph models (sparseGCN) and hard kuma based latent graph models (kumaGCN). sparseGCN significantly outperforms depGCN. sanGCN is also better than depGCN in terms of F1 scores. kumaGCN performs the best, achieving 89.39 accuracy scores and 73.19 F1 scores, which empirically shows the importance of introducing stochastic semantic directly related connections between aspect word and the context words.

We additionally test two model variants, −latent and −dep, which denote kumaGCN models without using latent graphs or dependency trees. They both underperform the full model, which demonstrates the strength of combining the two graphs for learning better aspect-specific graph representations. Additionally, −latent is worse than −dep especially in terms of F1, which somehow shows that the automatically induced latent graph can be better than the dependency graph. As a result, we use kumaGCN as our final model.

Effect of λ To investigate how the trade-off between using automatically latent graphs and dependency tree may affect the ATSC performance, we vary λ in Eq 8 from 0 to 1 using a step size 0.1. Figure 2 shows the F1 scores achieved by kumaGCN on REST16 and REST15 with different λ. When λ = 0, the model degrades to depGCN; when λ = 1 the model solely relies on automatically learned latent graphs. λ = 0.2 gives the best results, which shows that the two of structures are complementary to each other. We therefore set λ to be 0.2.

4.2 Main Results

We compare our models with:

- LSTM. Tang et al. (2016a) present target-dependent LSTMs to model the interaction of the target and the context words.
- MemNet. Tang et al. (2016b) leverage multi-hops of attention layers on the context word embeddings for sentence representation.
- IAN. Ma et al. (2017) use interactive attention networks to interactively learn the relationship between the targets and their contexts.
<table>
<thead>
<tr>
<th>Model</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>69.56</td>
<td>67.70</td>
<td>69.28</td>
<td>63.09</td>
<td>78.13</td>
<td>67.47</td>
<td>77.37</td>
<td>55.17</td>
<td>86.80</td>
<td>63.88</td>
<td>76.23</td>
<td>63.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MemNet</td>
<td>71.48</td>
<td>69.90</td>
<td>70.64</td>
<td>65.17</td>
<td>79.61</td>
<td>69.64</td>
<td>77.31</td>
<td>58.28</td>
<td>85.44</td>
<td>65.99</td>
<td>76.90</td>
<td>65.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IAN</td>
<td>72.50</td>
<td>70.81</td>
<td>72.05</td>
<td>67.38</td>
<td>79.26</td>
<td>70.09</td>
<td>78.54</td>
<td>62.65</td>
<td>84.74</td>
<td>55.21</td>
<td>77.42</td>
<td>65.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNet-LF</td>
<td>72.98</td>
<td>71.43</td>
<td>74.61</td>
<td>70.14</td>
<td>80.42</td>
<td>71.03</td>
<td>78.47</td>
<td>59.47</td>
<td>89.07</td>
<td>70.43</td>
<td>79.11</td>
<td>68.50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Main results on five benchmark datasets: averaged accuracy (Acc.) and F1 score.

<table>
<thead>
<tr>
<th>Model</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
<th>ACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEN_BERT (Song et al., 2019)</td>
<td>75.14</td>
<td>74.15</td>
<td>76.96</td>
<td>73.67</td>
<td>84.29</td>
<td>77.22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RGAT+BERT (Wang et al., 2020)</td>
<td>76.15</td>
<td>74.88</td>
<td>78.21</td>
<td>74.07</td>
<td>86.60</td>
<td>81.35</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT-SPC (Devlin et al., 2019)</td>
<td>73.41</td>
<td>72.38</td>
<td>80.56</td>
<td>77.20</td>
<td>84.55</td>
<td>75.74</td>
<td>83.03</td>
<td>63.92</td>
<td>90.75</td>
<td>74.00</td>
</tr>
<tr>
<td>depGCN + BERT</td>
<td>75.58</td>
<td>74.58</td>
<td>81.19</td>
<td>77.67</td>
<td>85.00</td>
<td>78.79</td>
<td>84.13</td>
<td>67.28</td>
<td>91.39</td>
<td>74.25</td>
</tr>
<tr>
<td>kumaGCN + BERT</td>
<td>77.89</td>
<td>77.03</td>
<td>81.98</td>
<td>78.81</td>
<td>86.43</td>
<td>80.30</td>
<td>86.35</td>
<td>70.76</td>
<td>92.53</td>
<td>79.24</td>
</tr>
</tbody>
</table>

Table 4: Main results on five benchmark datasets when BERT is used.

- **TNet-LF**: Li et al. (2018a) adopt context-preserving transformations on a convolutional neural network enhanced model.
- **depGCN**: Zhang et al. (2019) apply aspect-specific GCNs based on dependency trees to extract syntactic features.
- **BERT-SPC**: This is a simple baseline by fine-tuning BERT for sentence classification.
- **AEN_BERT**: Song et al. (2019) employ an attentional encoder network and apply pre-trained BERT to the task.
- **RGAT+BERT**: Wang et al. (2020) use relational graph attention networks to incorporate the dependency edge type information.

**Without BERT, Using Glove Embeddings** Table 3 shows the results. KumaGCN outperforms all the baselines in terms of both averaged accuracy scores and averaged F1 scores. In particular, it improves the performance by 2.77 F1 points compared with the depGCN. The performance gain compared to depGCN can empirically demonstrate the effectiveness of introducing latent graphs for aspect sentiment analysis tasks. Considering the running time, the self-attention module and the gated combination module can make our model slower compared to depGCN. In practice, we compared our model with depGCN on Rest16 test dataset (616 examples), the inference time costs are 0.32s and 0.48s for depGCN and our model respectively, which does not add too much computation overhead.

Our model also significantly outperforms the state-of-the-art non-depGCN model TNet-LF\(^5\) in all the datasets except for Twitter. On Twitter, sparseGCN gives 72.64 accuracy, which is comparable to the performances of TNet-LF (72.98), which applies a topmost convolution 2D layer over the BiLSTM encoder to capture local n-grams and is thus less sensitive to informal texts without strong sequential patterns. We believe that the slight performance deficiency compared to TNet-LF is because of specific network settings. In particular, TNet-LF applies an attention-based context-preserving transformation to enhance the contextual representations produced by the BiLSTM encoder. For fair comparisons with baselines, we do not use such modules. If it were used for kumaGCN, the ACC/F1 scores are 73.51/72.06 for the twitter dataset, which is better than the performance of TNet-LF (72.98/71.43). To our knowledge, our model gives the best results without using BERT.

**Comparison with Sun et al. (2019b)'s model** Sun et al. (2019b) also proposed a GCN model based on dependency trees for aspect sentiment analysis similar to depGCN of Zhang et al. (2019).

---

\(^5\)The results of TNet-LF does not match the original paper because the original paper of TNet-LF potentially filters out some training examples. For example, the positive/negative/neutral examples of TNet-LF on Laptop14 are 980/858/454, while ours are 994/870/464, respectively.
Table 5: F1 scores comparisons between Sun et al. (2019b)’s model and our kumaGCN model using their settings.

<table>
<thead>
<tr>
<th>Model</th>
<th>Rest14</th>
<th>Laptop</th>
<th>Twitter</th>
<th>Rest16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re-run of Sun’s code</td>
<td>71.92</td>
<td>70.16</td>
<td>71.71</td>
<td>67.29</td>
</tr>
<tr>
<td>Our kumaGCN model</td>
<td>72.31</td>
<td>71.91</td>
<td>74.24</td>
<td>70.96</td>
</tr>
</tbody>
</table>

Sun et al. (2019b) use aspect-specific pooling over the dependency tree nodes to obtain the final representation vector, instead of using aspect mask and aspect-specific attention of Zhang et al. (2019). The data settings of Sun et al. (2019b)’s model are different from ours. For example, the positive/negative/neutral examples of their settings on Laptop14 are 976/851/455, while ours are 994/870/464. In addition, they include POS tags in the input. A direct comparison without reimplementation is unfair. Preliminary results show that the results of Sun et al. (2019b) is slightly worse than that of depGCN based on the Zhang et al. (2019)’s data settings.

We also add a head-to-head comparison with (Sun et al., 2019b) as shown in Table 5 using their data settings. In fact, based on their settings, our model can still achieve better F1 scores on all the datasets.

With BERT We compare our kumaGCN + BERT models with the state-of-the-art BERT-based models, also implement the depGCN+BERT model as baseline. Table 4 shows the results. depGCN+BERT generally performs better than BERT-SPC. Our model outperforms both depGCN+BERT and BERT-SPC on all the datasets. Compared to the current state-of-the-art dependency tree based models RGAT+BERT, our model are better on TWITTER and LAP14. On REST14, the accuracy score of our model is comparable to RGAT+BERT without using dependency label information. In addition, our model gives 86.35/70.76 (Rest15) and 92.53/79.24 (Rest16) Acc./F1 scores, which are the best results on the datasets to our knowledge.

4.3 Parameter-based Transfer Learning

We further perform experiments using parameter-based transfer learning by training one source model on the Twitter dataset, and testing the trained model on the restaurant datasets. Table 6 shows the results. Our model outperforms BERT-SPC on all the target domains, which empirically demon-
defined in Eq 9 of each context words with respect to “atmosphere” between depGCN and kumaGCN. For the target “atmosphere”, depGCN assigns the highest weight to the word “terrible”, which is a wrong sentiment word for this target, leading to a wrong prediction. In contrast, our model assigns the largest weight to the key sentiment word “cozy”, classifying it correctly.

Figure 4(b) shows the pruned dependency trees by only keeping dependency edges related to these two aspects. We observe that the current parse contains an edge between “cozy” and “horrible”, which might confuse depGCN to produce inappropriate representations. For further comparisons, we also extract the links of each target word $i$ from the latent graph $A_{lat}$ (Eq 6) by only keeping edges between $i$ and $j$ if and only if $j = \text{arg max}_{j \neq i} A_{i,j}$. If $j$ is not unique, we return all the index which corresponds to the same highest value. Figure 4(c) and Figure 4(d) show the two latent graphs for “atmosphere” and “service”, respectively. First, we observe that the two latent graphs are significantly different. Second, each of them contains only a few edges related to the semantic contexts for sentiment classification. We also verify that there is no such an edge between “cozy” and “horrible” when inducing the latent graph of “atmosphere”. This can be an example to show that our model can learn aspect-specific latent graphs. With these automatically induced graphs, our model can learn better aspect-aware representations, providing better attention weights than depGCN.

5 Conclusion

We considered latent graph structures for aspect sentiment classification by investigating a variety of neural networks for structure induction, and novel gated mechanisms to dynamically combine different structures. Compared with dependency tree GCN baselines, the model does not introduce additional model parameters, yet significantly enhances the representation power. Experiments on five benchmarks show effectiveness of our model. To our knowledge, we are the first to investigate latent structures for aspect level sentiment classification, achieving the best-reported accuracy on five benchmark datasets.

Acknowledgment

Yue Zhang is the corresponding author. Thanks to anonymous reviewers for their insightful comments and suggestions. This project is supported by XXXX.

References


