Inducing Target-Specific Latent Structures for Aspect Sentiment Classification

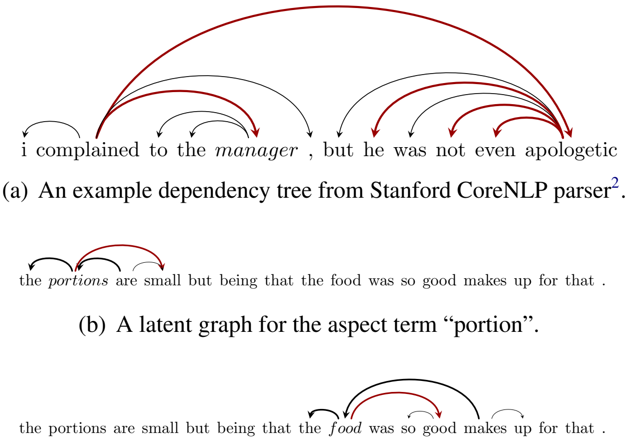
Chenhua Chen, Zhiyang Teng and **Yue Zhang**
School of Engineering, Westlake University, China
Institute of Advanced Technology, Westlake Institute for Advanced Study
{chenchenhua, tengzhiyang}@westlake.edu.cn, yue.zhang@wias.org.cn

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".


i complained to the *manager* , but he was not even apologetic
the *portions* are small but being that the food was so good makes up for that .
the portions are small but being that the *food* was so good makes up for that .

(a) An example dependency tree from Stanford CoreNLP parser[2].

(b) A latent graph for the aspect term "portion".

(c) A latent graph for the aspect term "food".

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. It has been shown to be more effective for learning 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, the existing methods suffer 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 according to a treebank 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 $complained \rightarrow manager$ and $complained \rightarrow not\ apologetic$, 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

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 5596–5607,
November 16–20, 2020. ©2020 Association for Computational Linguistics

classification. To this end, existing work has investigated 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.", ideal structures for the aspects "portions" and "food" can consist of links relevant to the terms and their opinion words only, without introducing additional information.

We empirically investigate three different methods for inducing semantic dependencies, including attention (Vaswani et al., 2017), sparse attention (Correia et al., 2019) and hard Kuma discrete structures (Bastings et al., 2019). In particular, attention has been used as a soft alignment structure 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 consider 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 the state-of-the-art results in the literature, achieving significantly better accuracies than models without using latent graphs. To our knowledge, we are the first to investigate automatically inducing tree structures for targeted sentiment classification. We release our code at https://github.com/CCSoleil/latent_graph_atsc.

2 Related Work

Aspect-level sentiment analysis Aspect-level sentiment analysis includes three main subtasks, 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 word 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 an aspect with its opinion words. Zhang et al. (2016) design gating mechanisms to select useful contextual information for each target. 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) used relational graph attention networks to incorporate 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

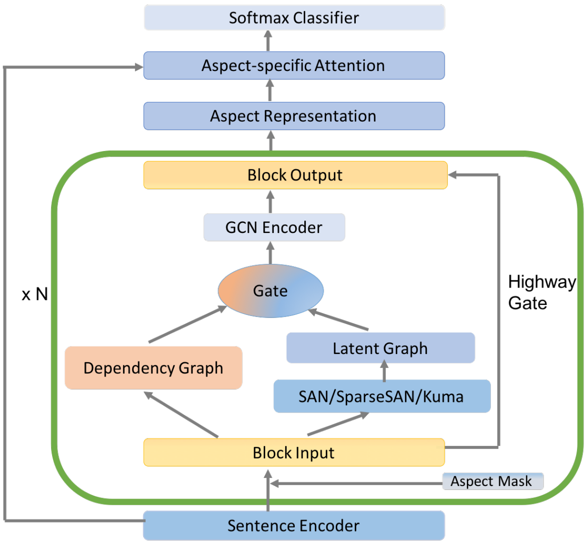


Figure 1: Model architecture.

operator over latent dependency trees, which can be regarded as a special case of introducing sparsity constraints into the softmax function (Niculae et al., 2018; Peters et al., 2019; Correia et al., 2019). Bastings et al. (2019) used HardKuma to sample stochastic interpretable discrete graphs for interpreting the classification results. Corro and Titov (2018) induced dependency structure for unsupervised parsing with a differentiable perturb-and-parsing method. While previous work obtains different structures using different methods, we investigate multiple methods for ATSC.

More in line with our work, Yogatama et al. (2016) and Zhang et al. (2018) considered reinforcement learning for inducing latent structures for text classification. Our work is in line but differs in two main aspects. First, we consider aspect-based sentiment, learning a different structure for each aspect term in the same sentence. Second, we empirically compare different methods for latent graph induction, and investigate complementary effects with dependency trees. To our knowledge, we are the first to consider inducing structures automatically for aspect-based sentiment classification.

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 $\mathbf{A}$, 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 $s = w_1 w_2 \dots w_n$, we first obtain the embedding vector $\mathbf{x}_i$ of each w_i using a lookup table $\mathbf{E} \in \mathbb{R}^{|V| \times d_w}$ (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 standard practice by feeding the input "[CLS] w_1 w_2 $\dots w_n$ [SEP] w_f $w_{f+1} \dots w_e$" to BERT to obtain aspect specific representations, where $c = w_f$ $w_{f+1} \dots w_e$ 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 $\mathbf{ce}_i^0$ 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 $\mathbf{ce}_i^0$. Formally, given an aspect $w_f w_{f+1} \dots w_e$, the masked $\mathbf{ce}_i^0$ is $\mathbf{h}_i^0 = m_i \mathbf{ce}_i^0$, where m_i is given by,

$$m_i = \begin{cases} 1 - \frac{f-i}{n} & 1 \le i < f, \\ 0 & f \le i \le e, \\ 1 - \frac{i-e}{n} & e < i \le n. \end{cases} \tag{1}$$

In this way, the more similar the context words are to the aspect, the higher their weights are. We denote the sentence representation as $\mathbf{H} = [\mathbf{h}_i^0, \mathbf{h}_1^0, \dots, \mathbf{h}_n^0]$, which is used for inducing latent graphs later.

3.2 Dependency Tree Representation

Given a sentence $s = w_1 w_2 \dots w_n$ 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 $\mathbf{A}_{dep}$ is given by

$$\mathbf{A}_{dep}[i,j] = \begin{cases} 1 & \text{if } i \rightarrow j \text{ or } i \leftarrow j, \\ 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$\mathbf{A}_{dep}$ represents the syntactic dependencies between word pairs.

3.3 Latent Graph

We propose to learn latent graphs $\mathbf{A}_{lat}$ 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 $\mathbf{H}$, the similarity score α_{ij} can be regarded as the interaction strength between node i and node j. $\mathbf{A}_{lat}$ is given by

$$\mathbf{A}_{lat} = \mathrm{softmax}\Big(\frac{(\mathbf{Q}\mathbf{W}_q)(\mathbf{K}\mathbf{W}_k)^T}{\sqrt{d}}\Big) \tag{3}$$

where $\mathbf{Q}$ and $\mathbf{K}$ are two copies of $\mathbf{H}$, representing the query and key vectors, respectively. $\mathbf{W}_q \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_k \in \mathbb{R}^{d\times d}$ 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 the column so that the sum of each row in $\mathbf{A}_{lat}$ equals to 1.

Multi-head SANs partition the graph representation $\mathbf{H}$ into multiple non-overlapping heads $\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_K]$, where K is the number of heads and $\mathbf{H}_i \in \mathbb{R}^{n\times \frac{d}{K}}$. For the i-th head, Eq 3 is independently applied to generate $\mathbf{A}^i_{head}$. The final latent graph averages the latent graphs of all heads,

$$\mathbf{A}_{lat} = \frac{\sum_{i=1}^{K} \mathbf{A}^i_{head}}{K}. \tag{4}$$

Sparse-self-attention-based latent graph SANs learn a fully connected latent graph, where dense attention weights 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,

$$\mathbf{A}_{lat} = 1.5\text{-entmax}\Big(\frac{(\mathbf{Q}\mathbf{W}_q)(\mathbf{K}\mathbf{W}_k)^T}{\sqrt{d}}\Big), \tag{5}$$

where 1.5-entmax[3] is applied to each row of the resulted matrix, with 1.5-entmax$(\mathbf{x})$ = $\arg\max_{\mathbf{p}\in\triangle^d} \mathbf{p}^T\mathbf{x} + \mathbf{H}^T_{1.5}(\mathbf{p})$. Here $\mathbf{H}^T_{1.5}(\mathbf{p})$ is an entropy function and $\mathbf{H}^T_{1.5}(\mathbf{p}) = \frac{1}{1.5\times(1.5-1)}\sum_{j=1}^{d}(p_j - p_j^{1.5})$. For more details, readers can refer to Peters et al. (2019).

Similar to Eq 4, multi-head SANs are used for 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 $\alpha_{ij} \sim \mathrm{HardKuma}(a, b, l, r)$, where HardKuma is a rectified Kumaraswamy distribution which includes both 0 and 1 in the support of Kumaraswamy distribution[4], $a > 0$ and $b > 0$ are parameters to control the shape of the Hard Kumaraswamy probability distribution, $l < 0$ and $r > 1$ define the supporting open interval (l, r). A sample of α_{ij} can be obtained by gradient reparameterization tricks (Kingma and Welling, 2013; Jang et al., 2016),

$$\begin{aligned} s_1 &= F^{-1}_{Kuma}(u; a, b), \\ s_2 &= l + (r-l) \times s_1, \\ z &= \min(1, \max(0, s_2)), \end{aligned}$$

where u is a uniform random variable and $u \sim \mathcal{U}(0, 1)$ which replaces the HardKuma sample, F^{-1}_{Kuma} is the inverse c.d.f. of the Kumaraswamy distribution and $F^{-1}_{Kuma}(u, a, b) = (1-(1-u)^{1/b})^{1/a}$. s_1 is a sample of the Kumaraswamy distribution, s_2 is a 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 that 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 $\mathbf{a}$ and $\mathbf{b}$. With reparameterization, the sampling is independent of the model and the main goal is to represent $\mathbf{a}$ and $\mathbf{b}$ using neural networks. Specifically, $\mathbf{a}$ and $\mathbf{b}$ can be calculated by SANs. Formally,

[3] We use the implementation of 1.5-entmax from https://github.com/deep-spin/entmax.

[4] For more details, refer to Bastings et al. (2019).

$\mathbf{a} \in \mathbb{R}^{n \times n}$ for the whole graph is given by

$$\begin{aligned}
\mathbf{H}_a &= \text{MHSAN}(\mathbf{H}, \mathbf{H}, \mathbf{H}), \\
\mathbf{C}_a &= \text{LN}(\text{FFN}(\mathbf{H}_a) + \mathbf{H}_a), \\
\mathbf{s}_a &= \mathbf{C}_a \mathbf{C}_a^T, \\
\mathbf{n}_a &= \frac{\mathbf{s}_a - \text{mean}(\mathbf{s}_a)}{\text{std}(\mathbf{s}_a)}, \\
\mathbf{a} &= \text{softplus}(\mathbf{n}_a), \\
\mathbf{A}_{lat} &\sim \text{HardKuma}(\mathbf{a}, \mathbf{b}, l, r),
\end{aligned} \tag{6}$$

where MHSAN, LN and FFN denote multi-head self attention networks, layer normalization and position-wise feed-forward networks, respectively. In our model, the particular networks of MHSAN, LN and FFN are taken from Transformer (Vaswani et al., 2017). $\mathbf{b}$ is defined in a similar way, but with different parameters. Here $\mathbf{H}_a$ is the initial result of MHSAN, $\mathbf{C}_a$ considers residual connections and feature transformations, $\mathbf{s}_a$ is the initial similarity score calculated by self attention, $\mathbf{n}_a$ denotes the normalized similarity scores and $\mathbf{a}$ is ensured to be non-negative by applying the softplus activation function over $\mathbf{n}_a$.

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 $G = \{V, \mathbf{A}\}$. $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacent matrix between nodes. Let the representation vector of v_i at the l-th layer be $\mathbf{h}_i^l$ and $\mathbf{h}_i^l \in \mathbb{R}^d$, where d is the node vector dimension. The whole graph representation of the l-th layer $\mathbf{H}^l$ is the concatenation of all the node vectors of this layer, namely $\mathbf{H}^l = [\mathbf{h}_1^l, \mathbf{h}_2^l, ..., \mathbf{h}_n^l]$ and $\mathbf{H}^l \in \mathbb{R}^{n \times d}$. The graph convolution for $\mathbf{H}^l$ is given by:

$$\mathbf{H}^l = \rho(\mathbf{A}\mathbf{H}^{l-1}\mathbf{W}^l + \mathbf{b}^l), \tag{7}$$

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

3.5 Gated Combination

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

$$\begin{aligned}
\mathbf{I}_{dep} &= \mathbf{A}_{dep}\mathbf{H}_{in}\mathbf{W}, \\
\mathbf{I}_{lat} &= \mathbf{A}_{lat}\mathbf{H}_{in}\mathbf{W}, \\
\mathbf{g} &= \sigma(\mathbf{I}_{lat}), \\
\mathbf{I}_{com} &= (1 - \lambda\mathbf{g}) \odot \mathbf{I}_{dep} + \lambda\mathbf{g} \odot \mathbf{I}_{lat}, \\
\mathbf{H}_{out} &= \rho(\mathbf{I}_{com} + \mathbf{b}),
\end{aligned} \tag{8}$$

where $\mathbf{g}$ 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 $\mathbf{W}$ is the same for $\mathbf{A}_{dep}$ and $\mathbf{A}_{lat}$, which suggests that our model does not introduce any additional parameters. $\mathbf{AHW}$ in Eq 8 can be replaced with $\mathbf{I}_{com}$, which is a gated combination of $\mathbf{A}_{dep}\mathbf{HW}$ and $\mathbf{A}_{lat}\mathbf{HW}$. This combination equals that we first merge $\mathbf{A}_{dep}$ and $\mathbf{A}_{lat}$ into a single graph $\mathbf{A}_{com}$ using dynamic gating mechanisms and then directly use $\mathbf{A}_{com}$ as $\mathbf{A}$ in Eq 7 to obtain the graph representations.

Gated GCN blocks In practice, we stack N GCN layers. For different layers, the convolution parameters are different. A highway network is used to combine the feature representations in adjacent layers. Formally, given the input representation of the l-th block $\mathbf{H}^{l-1}$, the input to the first block is the aspect-aware sentence representation $\mathbf{H}^0$, the output of the l-th block $\mathbf{H}^l$ is given by,

$$\begin{aligned}
\mathbf{g}_l &= \sigma(\mathbf{H}^{l-1}), \\
\mathbf{H}_{com}^l &= \mathit{gatedcombine}(\mathbf{H}^{l-1}, \mathbf{A}_{dep}, \mathbf{A}_{lat}), \\
\mathbf{H}^l &= \mathbf{g}_l \odot \mathbf{H}_{com}^l + (1 - \mathbf{g}_l) \odot \mathbf{H}^{l-1},
\end{aligned}$$

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 $\mathbf{H}^N$, we obtain the representations for the aspect $w_f w_{f+1} \ldots w_e$ using $\mathbf{H}_f^N, \mathbf{H}_{f+1}^N \ldots, \mathbf{H}_e^N$. The final aspect-specific feature representation $\mathbf{z}$ is given by an attention network over the sentence representation vectors $\mathbf{ce}_i^0$

$$\begin{aligned}
\gamma_t &= \sum_{i=f}^{e} \mathbf{ce}_t^0 \mathbf{H}_i^N, \\
\boldsymbol{\alpha} &= \text{softmax}(\boldsymbol{\gamma}), \\
\mathbf{z} &= \boldsymbol{\alpha}\mathbf{C},
\end{aligned} \tag{9}$$

where $\mathbf{C} = [\mathbf{ce}_1^0, \mathbf{ce}_2^0, \ldots, \mathbf{ce}_n^0]$ is the contextualized representations produced by the sentence coder, γ_t is the attention scores of the t-th context word with respect to the aspect term, $\boldsymbol{\alpha}$ denotes the normalized attention scores and $\mathbf{z}$ is the final aspect-specific representation.

Softmax classifier The aspect-specific representation vector $\mathbf{z}$ 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,

$$\mathbf{p} = \mathrm{softmax}(\mathbf{W}_o \mathbf{z} + \mathbf{b}_o), \tag{10}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are model parameters and $\mathbf{p}$ is the predicted sentiment probability distribution.

3.7 Training

The classifier is trained by maximizing the negative log-likelihood of a set of training samples $D = \{x_i, y_i\}_{i=1}^N$, where each x_i contains a set of aspects $c_{i,j}$. Formally, the loss function is given by

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \sum_{c_{i,j}} \log \mathbf{p}_{y_{i,j}} + \frac{\lambda'}{2} ||\boldsymbol{\theta}||^2,$$

where N is the number of training instances, $\boldsymbol{\theta}$ is the set of model parameters, λ' is a regularization hyperparameter, $y_{i,j}$ is the training label of the j-th aspect $c_{i,j}$ in x_i and $\mathbf{p}_{y_{i,j}}$ is the aspect classification probability for $c_{i,j}$, which is given by Eq 10.

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 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[5]. 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, reporting accuracy (Acc.) and macro-f1 (F1).

Dataset		#Pos.	#Neu.	#Neg.
TWITTER	Train/Test	1,561/173	3,127/346	1,560/173
LAP14	Train/Test	994/341	464/169	870/128
REST14	Train/Test	2,164/728	637/196	807/196
REST15	Train/Test	912/326	36/34	256/182
REST16	Train/Test	1,240/469	69/30	439/117

Table 1: Dataset statistics.

Model	depGCN	sanGCN	sparseGCN	kumaGCN		
				full	-latent	-dep
Acc.	88.99	88.64	89.29	**89.39**	89.12	89.23
F1	67.48	69.37	72.14	**73.19**	70.89	72.04

Table 2: Model performances on REST16.

4.1 Development Results

Effect of latent graphs Table 2 shows the performances 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, respectively. 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 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

[5] http://nlp.stanford.edu/data/glove.840B.300d.zip

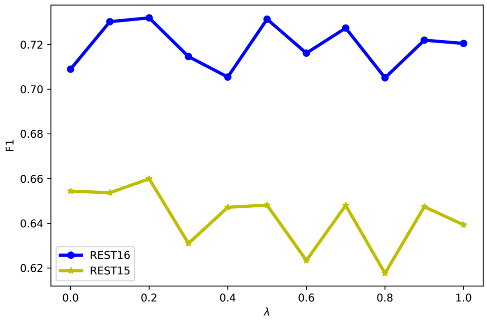


Figure 2: Effect of λ on REST16 and REST15 using kumaGCN.

λ. When $\lambda = 0$, the model degrades to depGCN; when $\lambda = 1$ the model relies solely on automatically learned latent graphs. $\lambda = 0.2$ gives the best results, which shows that the structures are complementary to each other. We thus set λ =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.
- **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**[6]. 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 method. 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 compare our model with depGCN on the Rest16 test dataset (616 examples). The inference time costs are 0.32s and 0.48s for depGCN and our model respectively, which shows that our model does not add too much computation overhead.

Our model also significantly outperforms the state-of-the-art non-depGCN model TNet-LF[7] on 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 a 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 comparison with baselines, we do not use such modules[8]. 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). 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 LAP14 are 976/851/455, while ours are 994/870/464. In addition, they include POS tags in the input. A direct comparison without reimple-

[6] We adopt the widely used implementation https://github.com/songyouwei/ABSA-PyTorch.

[7] 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.

[8] If it were used for kumaGCN, the ACC/F1 scores are 73.51/72.06 on the twitter dataset, which are better than the performance of TNet-LF (72.98/71.43).

Model	Twitter		Lap14		Rest14		Rest15		Rest16		Average	
	Acc.	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
LSTM	69.56	67.70	69.28	63.09	78.13	67.47	77.37	55.17	86.80	63.88	76.23	63.46
MemNet	71.48	69.90	70.64	65.17	79.61	69.64	77.31	58.28	85.44	65.99	76.90	65.80
IAN	72.50	70.81	72.05	67.38	79.26	70.09	78.54	62.65	84.74	55.21	77.42	65.23
TNet-LF	**72.98**	**71.43**	74.61	70.14	80.42	71.03	78.47	59.47	89.07	70.43	79.11	68.50
depGCN	72.15	70.40	75.55	71.05	80.77	72.02	79.89	61.89	88.99	67.48	79.47	68.57
sparseGCN	72.64	71.02	75.91	71.89	81.30	72.68	80.57	65.52	89.29	72.14	79.94	70.65
kumaGCN	72.45	70.77	**76.12**	**72.42**	**81.43**	**73.64**	**80.69**	**65.99**	**89.39**	**73.19**	**80.02**	**71.20**

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

Model	Twitter		Lap14		Rest14		Rest15		Rest16	
	Acc.	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
AEN_BERT (Song et al., 2019)	75.14	74.15	76.96	73.67	84.29	77.22	-	-	-	-
RGAT+BERT (Wang et al., 2020)	76.15	74.88	78.21	74.07	**86.60**	**81.35**	-	-	-	-
BERT-SPC (Devlin et al., 2019)	73.41	72.38	80.56	77.20	84.55	75.74	83.03	63.92	90.75	74.00
depGCN + BERT	75.58	74.58	81.19	77.67	85.00	78.79	85.23	70.13	91.56	77.31
kumaGCN + BERT	**77.89**	**77.03**	**81.98**	**78.81**	86.43	80.30	**86.35**	**70.76**	**92.53**	**79.24**

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

Model	Rest14	Lap14	Twitter	Rest16
Re-run of Sun's code	71.92	70.16	71.71	67.29
Our kumaGCN model	**72.31**	**71.91**	**74.24**	**70.96**

Table 5: F1 comparisons between Sun et al. (2019b)'s model and our kumaGCN model using their settings.

Model\Target	Rest14	Rest15	Rest16
BERT-SPC	49.46/43.54	44.10/39.69	45.45/33.67
depGCN+BERT	63.12/55.83	56.83/47.68	62.99/45.73
kumaGCN+BERT	**72.14/61.77**	**65.31/52.31**	**71.10/49.67**

Table 6: Results for transfer learning from Twitter to the three datasets.

mentation 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. It can be seen that 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, and 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 is 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 demonstrates the strong aspect-specific semantic representation abilities of our proposed model. Compared to depGCN+BERT, our model gives improved results on the three datasets by about 10.0 accuracy points, which suggests that the induced latent structures have strong robustness for capturing aspect-opinion interactions.

4.4 Attention Distance

Figure 3 shows the distribution of the averaged attention weights of the context words according to the aspect terms on the test sentences of Rest16. In both cases, the attention scores defined in Eq 9 are shown. kumaGCN makes the distribution sharper than depGCN, focusing more on the context within 1 or 2 words. This observation also confirms data bias in the training set, where many

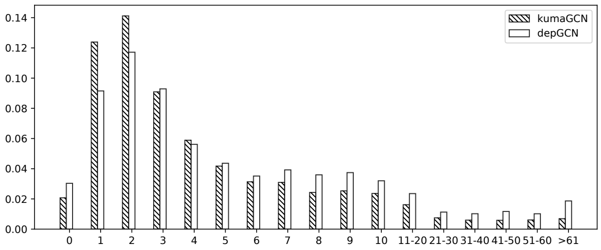


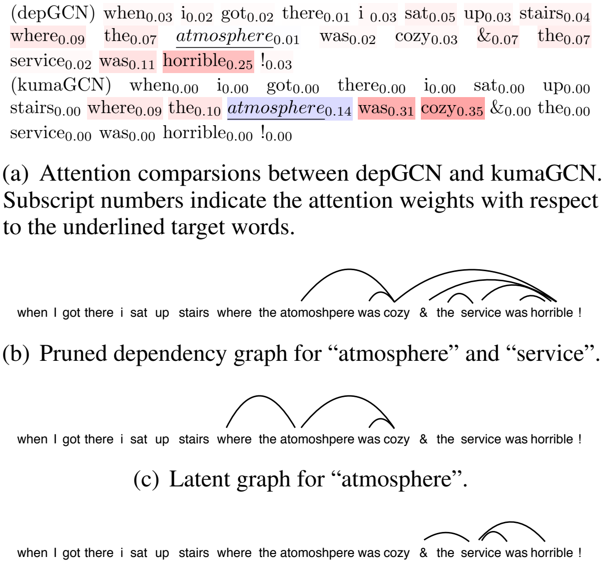
Figure 3: Attention distance distribution. x-axis: distance to the aspect terms; y-axis: attention scores.

opinion words are close to the aspect term (Tang et al., 2016b). Though depGCN can assign high weights to words far away from the target by using syntactic path dependencies, it may also bring in more noise. kumaGCN potentially circumvents this problem.

4.5 Case Study

To gain more insights into our model's behavior, we show one case study in Figure 4 using the example "when i got there i sat up stairs where the **atmosphere** was cozy & the **service** was horrible !". This example contains two aspects "atmosphere" and "service". Both depGCN and kumaGCN can correctly classify the sentiment of "service" as negative. However, depGCN cannot recognize the positive sentiment of "atmosphere" while kumaGCN can. Figure 4(a) compares the attention weights $\boldsymbol{\alpha}$ defined in Eq 9 of each context word with respect to "atmosphere" between depGCN and kumaGCN. For the target "atmosphere", depGCN assigns the highest weight to the word "terrible", which is an irrelevant sentiment word to this target, leading to an incorrect prediction. In contrast, our model assigns the largest weight to the key sentiment word "cozy", classifying it correctly.

Figure 4(b) shows 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 mislead depGCN to produce inappropriate representations. For further comparisons, we also extract the links of each target word i from the latent graph $\mathbf{A}_{lat}$ (Eq 6) by only keeping edges between i and j if and only if $j = \arg\max_{j'} A_{i,j'}$. If j is not unique, we return all the indices which correspond 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 edge between "cozy" and "horriable" 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.



(depGCN) when$_{0.03}$ i$_{0.02}$ got$_{0.02}$ there$_{0.01}$ i $_{0.03}$ sat$_{0.05}$ up$_{0.03}$ stairs$_{0.04}$ where$_{0.09}$ the$_{0.07}$ <u>*atmosphere*</u>$_{0.01}$ was$_{0.02}$ cozy$_{0.03}$ &$_{0.07}$ the$_{0.07}$ service$_{0.02}$ was$_{0.11}$ horrible$_{0.25}$!$_{0.03}$
(kumaGCN) when$_{0.00}$ i$_{0.00}$ got$_{0.00}$ there$_{0.00}$ i$_{0.00}$ sat$_{0.00}$ up$_{0.00}$ stairs$_{0.00}$ where$_{0.09}$ the$_{0.10}$ <u>*atmosphere*</u>$_{0.14}$ was$_{0.31}$ cozy$_{0.35}$ &$_{0.00}$ the$_{0.00}$ service$_{0.00}$ was$_{0.00}$ horrible$_{0.00}$!$_{0.00}$

(a) Attention comparsions between depGCN and kumaGCN. Subscript numbers indicate the attention weights with respect to the underlined target words.

when I got there i sat up stairs where the atomoshpere was cozy & the service was horrible !

(b) Pruned dependency graph for "atmosphere" and "service".

when I got there i sat up stairs where the atomoshpere was cozy & the service was horrible !

(c) Latent graph for "atmosphere".

when I got there i sat up stairs where the atomoshpere was cozy & the service was horrible !

(d) Latent graph for "service".

Figure 4: Comparisons of graph representations.

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.

Acknowledgments

Yue Zhang is the corresponding author. Thanks to anonymous reviewers for their insightful comments and suggestions. This project is supported by the Westlake-BrightDreams Robotics research grant and a research grant from Rxhui Inc.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Jihun Choi, Kang M. Yoo, and Sang-goo Lee. 2017. Learning to compose task-specific tree structures. In *AAAI*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proc. of EMNLP-IJCNLP*, pages 2174–2184, HK, China.

Caio Corro and Ivan Titov. 2018. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *arXiv preprint arXiv:1807.09875*.

Caio Corro and Ivan Titov. 2019. Learning latent trees with stochastic perturbations and differentiable dynamic programming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5508–5521, Florence, Italy. Association for Computational Linguistics.

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

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, Baltimore, Maryland. Association for Computational Linguistics.

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.

Binxuan Huang and Kathleen Carley. 2019. Syntax-aware aspect level sentiment classification with graph attention networks. In *Proc. of EMNLP-IJCNLP*, pages 5468–5476, HK, China.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160, Portland, Oregon, USA. Association for Computational Linguistics.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. *CoRR*, abs/1702.00887.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956, Melbourne, Australia. Association for Computational Linguistics.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018b. Aspect term extraction with history attention and selective transformation. *arXiv preprint arXiv:1805.00760*.

Yunlong Liang, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. A novel aspect-guided deep transition model for aspect based sentiment analysis. In *Proc. of EMNLP-IJCNLP*, pages 5568–5579, HK, China.

Peiqin Lin, Meng Yang, and Jianhuang Lai. 2019. Deep mask memory network with semantic dependency and context moment for aspect level sentiment classification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5088–5094. International Joint Conferences on Artificial Intelligence Organization.

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, Valencia, Spain. Association for Computational Linguistics.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *arXiv:1709.00893*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Pedro Henrique Martins, Zita Marinho, and Andr? F. T. Martins. 2020. Sparse text generation.

Vlad Niculae, André F. T. Martins, and Claire Cardie. 2018. Towards dynamic computation graphs via sparse latent structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 905–911, Brussels, Belgium. Association for Computational Linguistics.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018. Backpropagating through structured argmax using a SPIGOT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1863–1873, Melbourne, Australia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543, Qatar.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, Florence, Italy. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5 : aspect based sentiment analysis. In *Proc. of the Workshop on SemEval*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proc. of the Workshop on SemEval*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proc. of the Workshop on SemEval*, pages 27–35.

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*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

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

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019a. 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)*, pages 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2019b. Aspect-level sentiment analysis via convolution over dependency tree. In *Proc. of EMNLP-IJCNLP*, pages 5678–5687, HK, China.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proc. of COLING*, pages 3298–3307, Japan.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proc. of EMNLP*, pages 214–224, USA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv:1706.03762*.

Duy-Tin Vo and Yue Zhang. 2015. Deep learning for event-driven stock prediction. In *Proc. of IJCAI*, Argentina.

Hai Wan, Yufei Yang, Jianfeng Du, Yanan Liu, Kunxun Qi, and Jeff Z Pan. 2020. Target-aspect-sentiment joint detection for aspect-based sentiment analysis. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. 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 2324–2335, Minneapolis, Minnesota. Association for Computational Linguistics.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proc. of EMNLP-IJCNLP*, pages 4567–4577, HK, China.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proc. of AAAI*, page 3087–3093.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pinlong Zhao, Linlin Hou, and Ou Wu. 2019. Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification. In *arXiv:1906.04501*.