Encoding Syntactic Information into Transformers for Aspect-Based Sentiment Triplet Extraction

Li Yuan, Jin Wang, *Member, IEEE*, Liang-Chih Yu, *Member, IEEE*, Xuejie Zhang, *Member, IEEE*

Abstract—*Aspect-based sentiment triplet extraction* (ASTE) aims to extract triplets consisting of aspect terms and their associated opinion terms and sentiment polarities from sentences, a relatively new and challenging subtask of aspect-based sentiment analysis (ABSA). Previous studies have used either pipeline models or unified tagging schema models. These models ignore the syntactic relationships between the aspect and its corresponding opinion words, which leads them to mistakenly focus on syntactically unrelated words. One feasible option is to use a graph convolution network (GCN) to exploit syntactic information by propagating the representation from the opinion words to the aspect. However, such a method considers all syntactic dependencies to be of the same type and thus may still incorrectly associate unrelated words to the target aspect through the iterations of graph convolutional propagation. Herein, a syntax-aware transformer (SA-Transformer) is proposed to extend the GCN strategy by fully exploiting the dependency types of edges to block inappropriate propagation. The proposed approach can obtain different representations and weights even for edges with the same dependency type according to their adjacent dependency type of edges. Instead of using a GCN layer, we used an *L*-layer SA transformer to encode syntactic information in the word-pair representation to improve performance. Experimental results on four benchmark datasets show that the proposed model outperforms various previous models for ASTE.

Index Terms—Aspect Sentiment Triplet Extraction, Sentiment Analysis, Syntactic Information, Transformers

1 Introduction

Aspect-based sentiment analysis (ABSA) [1] aims to recognize the sentiment polarity and opinion of targeted aspects in a given sentence [1], [2], [3], which is a useful technique for various sentiment applications [4], [5], [6], [7], [8]. ABSA is composed of several related subtasks, such as aspect term extraction (ATE), opinion term extraction (OTE), and aspect sentiment classification (ASC). Here, ATE indicates **what** aspect is being discussed, ASC shows **how** the sentiment polarity impacts the aspect, and OTE explains **why** the polarity is associated [9].

Previous works have attempted to either solve the above subtasks individually or solve two of the subtasks jointly, such as ATE and ASC [10], [11], [12], [13], [14], [15] or ATE and OTE [16], [17]. To further integrate the tree subtasks, Peng et al. [9] pioneered a unified task, namely, aspect-based sentiment triplet extraction (ASTE), which aims to provide a complete analysis of a user-generated text by producing all triplets (*aspect term*, *opinion term*, and corresponding *sentiment polarity*) from sentences. Fig. 1 shows an example review. The ASTE task requires a model to generate three triplets: (*staff*, *very courteous*, Pos), (*staff*, *great*, Pos), and (*food*, *terrible*, Neg), where *staff* and *food* are aspect terms; *very courteous*, *great*, and *terrible* are corresponding opinion terms; and Pos and Neg denote their sentiment polarity.

Previous studies have typically accomplished ASTE tasks by using a two-stage pipeline approach with sequence labeling models [9]. This approach first identifies the aspect terms with their sentiment, as well as the opinion terms. The extracted aspect terms are then matched with each opinion term to determine their consistency. Unfortunately, the pipeline approach ignores the relationships between elements and is prone to error propagation. Alternatively, another viable option is to apply a multitask strategy to integrate both stages into a joint framework [18], [19], [20], [21], [22]. The main limitation of the joint approach is that it cannot efficiently handle scenarios in which a review contains multiple relational triplets that overlap with each other; e.g., in the previous example sentence, both opinion terms *very courteous* and *great* should be associated with the same aspect term, *staff*.

Several recent works have studied the overlapping triplet problem by applying a grid tagging scheme (GTS) [23], [24]. Therefore, the ASTE task is converted to predict the relation tags of word pairs, as shown in the lower part of Fig. 1. The tags A and O denote that the word pair represents the same aspect term and opinion term, the tag N denotes no relation between the word pair, and Pos, Neg and Neu are the sentiment labels. For example, the polarities between word pairs (*staff*, *courteous*) and (*staff*, *great*) are both positive. However, the equivalence classification between word pairs may lead to an inappropriate association between the aspect terms and opinion terms. For example, *great* could be simultaneously associated with both aspects terms *staff* and *food*.

To address the limitations of the above models, graph-

Corresponding authors: Jin Wang and Liang-Chih Yu.

- *L. Yuan, J. Wang and X. Zhang are with the School of Information Science and Engineering, Yunnan University, Kunming 650000, China E-mail: yuanli@mail.ynu.edu.cn; wangjin@ynu.edu.cn; xjzhang@ynu.edu.cn*
- *L.-C. Yu is with the Department of Information Management, Yuan Ze University, Taoyuan 32003, Taiwan E-mail: lcyu@saturn.yzu.edu.tw*

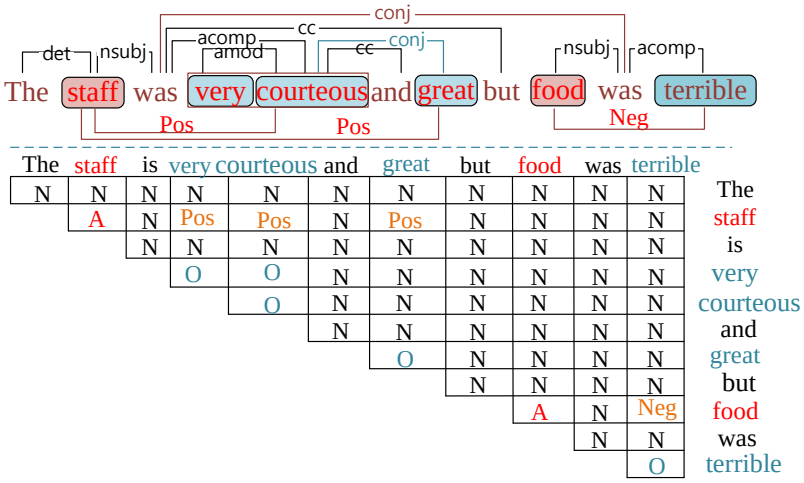


Fig. 1. : Dependency parsing and grid tagging of a given sentence. N, A, O, Pos, Neg, and Neu respectively denote the word-pair tags of none, aspect opinion, positive, negative, and neutral.

based methods have been proposed to introduce syntactic dependencies to model the relationship between words [25], [26]. By parsing the text into a dependency tree, a special type of graph is constructed based on the adjacency matrix. Graph convolution networks (GCNs) can then propagate the representations through the edges from opinion words to the corresponding aspects. However, these models consider all syntactic dependencies to be of the same type and assign an equal weight to each edge. The inappropriate association of less important words may still occur through multiple iterations of graph convolution propagation. In the example shown in Fig. 1, the representation of *courteous* can be correctly propagated to *staff* through the path of edges *courteous-acomp-was-nsubj-staff*, but it can also be incorrectly propagated to food through *courteous-acomp-was-conj-was-nsubj-food*.

Dependency types are useful features to model word relationships from the syntactic aspect, and different dependency types should be assigned different weights. For instance, the dependency types **nusbj** and **acomp** indicate a subject-object relation, and increasing their weights can help accomplish correct propagation (e.g., from *courteous* to *staff* and *terrible* to *food*). On the other hand, even the same dependency type may necessitate different weights. For instance, the example sentence in Fig. 1 contains two edges with **conj**. The **conj** between *was* and *was* should be assigned a lower weight to block inappropriate propagation (e.g., from *courteous* to *food* and *terrible* to *staff*), but the **conj** between *courteous* and *great* should be assigned a higher weight to help propagation from *great* to *staff*.

Based on this notion, this study proposes a syntax-aware transformer (SA-Transformer) to incorporate the knowledge of dependency types into graph neural networks for triplet extraction. The proposed method extends graph neural networks in three aspects. First, it can distinguish not only between edges with different dependency types but also those with the same dependency type to achieve more accurate graph propagation. This is accomplished by developing an adjacent edge attention (AEA) mechanism to learn the edge representation for each edge according to the dependency types of its adjacent edges. That is, the edges that have adjacent edges with different dependency types may have different representations and weights. Second, the edge representations are encoded into contextual word representations to learn the syntactic and positional relationships between the words to enhance word pair representations. Third, given that a multiword aspect/opinion term (e.g., *very courteous*) is divided by multiple consecutive word pairs for prediction, this study devises an adjacency inference strategy to improve triplet extraction for multiword aspect/opinion terms. This strategy can iteratively predict the tag of each word pair according to the predicted results of its adjacent word pairs instead of predicting each word pair independently. The proposed SA-Transformer model is evaluated with respect to four benchmark datasets. Experimental results show that the proposed method outperforms various previous models for ASTE.

The main contributions of this study are summarized as follows.

- We propose the SA-Transformer, which incorporates the knowledge of dependency types to extend graph neural networks for the ASTE task.
- We design the AEA mechanism that can learn different representations and weights for different edges, even for those with the same dependency type, thus achieving more accurate graph propagation.
- Experiments conducted on four benchmark datasets show that the proposed method outperforms existing methods for ASTE.

The rest of this paper is organized as follows. Section 2 briefly reviews existing methods for ASTE. Section 3 presents a detailed description of the proposed SA-Transformer model. Section 4 summarizes the implementation details and experimental results. Conclusions are finally drawn in Section 5.

2 Related Works

Previous ABSA works can be broadly divided into three independent extraction subtasks (ATE, OTE and ASC), the

TABLE 1
Different subtasks and corresponding methods for ABSA.

Tasks		Methods	Example Outputs
Independent Extraction	ATE	Attention [27], [28], [29]; Seq2Seq [30]	Staff; Food
	OTE	LSTM [31]; Attention [32]	Courteous; Great; Terrible
	ASC	Machine learning [33], [34], [35]; LSTM [36], [37], [38], [39]; Graph-based [40], [41], [42], [43], [44]	Positive; Negative
Joint Pair Extraction	ATE+ASC	Unified Tagging [11], [12], [13], [14], [15]	(Staff, Positive); (Food, Negative)
	ATE+OTE	Attention [16]; Graph-based [17]	(Staff, Courteous); (Staff, Great); (Food, Terrible)
Triplet Extraction	ASTE	Pipeline [9]; Multitask [18], [19], [20], [21], [22]; Word-Pair [23], [24], [25], [26]	(Staff, Courteous, Positive); (Staff, Great, Positive); (Food, Terrible, Negative)

joint pair extraction subtask, and ASTE subtask. This section briefly reviews different methods for these subtasks, which are summarized in Table 1.

2.1 Independent Extraction

ATE [27], [28], [29], [30] and OTE [31], [32] tasks are usually addressed by using a named entity recognition (NER) model to extract the target terms. In contrast to ATE and OTE tasks, ASC tasks aim to predict the sentiment polarity for a given aspect [33]. Most prior methods for ASC tasks employ statistical machine learning models [33], [34], [35]. However, obtaining features through these models is time- and labor-intensive. Later, after the rapid development of NLP tasks, various methods based on neural networks were proposed for ASC [36], [37], [38], [39].

Recent methods for ASC tasks mostly use graph-based models [40], [41], [42], [43], [44], which encode syntactic information to block the inappropriate propagation of unrelated contextual information to the aspect. For example, Tian et al. [41] proposed a type-aware graph convolutional network to capture the syntactic relation between context and target aspect. Xiao et al. [44] presented a syntactic edge-enhanced network with interactive attention, which leverages the edge information of a dependency parsing tree to interactively learn the representations of aspect terms with context.

2.2 Joint Pair Extraction

Recently, many researchers have focused on designing effective models to jointly extract aspect terms and sentiment polarity [10], [11], [12], [13], [14], [15], [16], [17]. For example, Li et al. [11] designed a multigranularity alignment network to decrease the false alignment of features in ASC and ATE tasks. Li et al. [12] designed a two-layer stacked LSTM model in which the lower-layer network guides the upper-layer network to improve performance on ATE and ASC tasks. Hu et al. [13] proposed a span-based model that outperforms joint and collapsed models.

To efficiently align the features of aspect granularity and domains, Wang et al. [16] and Dai et al. [17] attempted to coextract both aspect and opinion terms. Wang et al. [16] proposed a coupled multilayer attention network that uses a couple of attentions in each layer to extract aspect and opinion terms. The multilayer structure can capture both direct and indirect relations between words to achieve more precise extraction. Dai et al. [17] developed a weakly supervised method to extract aspect and opinion terms. It first mined the extraction rules based on the dependencies between words and then used the mined rules to expand the training data for neural model training.

2.3 Triplet Extraction

Aspect sentiment triplet extraction aims to jointly extract aspect terms, opinion terms, and their corresponding sentiment polarity, presenting a greater challenge than the independent subtasks. Previous works can be separated into the pipeline, multitask, and word-pair methods

Peng et al. [9] proposed a pipeline model for ASTE. It first extracted aspect terms, opinion terms, and sentiment polarities using the mutual influence between aspect and opinion terms and then employed a classifier to pair the extracted terms to obtain the final triplets. Peng et al. [9] also extended several joint pair extraction models [12], [16], [17] as a pipeline model.

Several studies have proposed multitask frameworks to jointly extract triplets [18], [19], [20], [21], [22]. Zhang et al. [18] used a sequence tagging strategy to extract aspect and opinion terms and predicted sentiment polarities using a table filling method. Chen et al. [19] converted the ASTE task into a machine reading comprehension (MRC) task and proposed a bidirectional MRC framework to gather information useful for triplet extraction from both the aspect-to-opinion and opinion-to-aspect directions. Xu et al. [20] designed a span-level model that can capture the span-to-span interactions instead of word-to-word interactions between the aspects and opinions for ASTE. Dai et al. [21] presented a bidirectional sentiment-dependence detector with double embeddings to obtain better sentence representations and gather information from both the aspect-to-opinion and opinion-to-aspect directions. Zhang et al. [22] proposed a dual decoder with a span copy mechanism that can extract multiple and overlapped triplets based on multitype information.

Wu et al. [23] designed a grid tagging schema to formalize the ASTE task into a word-pair task where classifications are applied between word pairs. Moreover, Xu et al. [24] used a model with a position-aware tagging scheme to jointly extract triplets. However, these methods may associate unrelated opinion terms with the target aspect, even if they are syntactically irrelevant. To address this limitation, Chen et al. [25] proposed the $\mathrm{S^3E^2}$ model based on a GCN

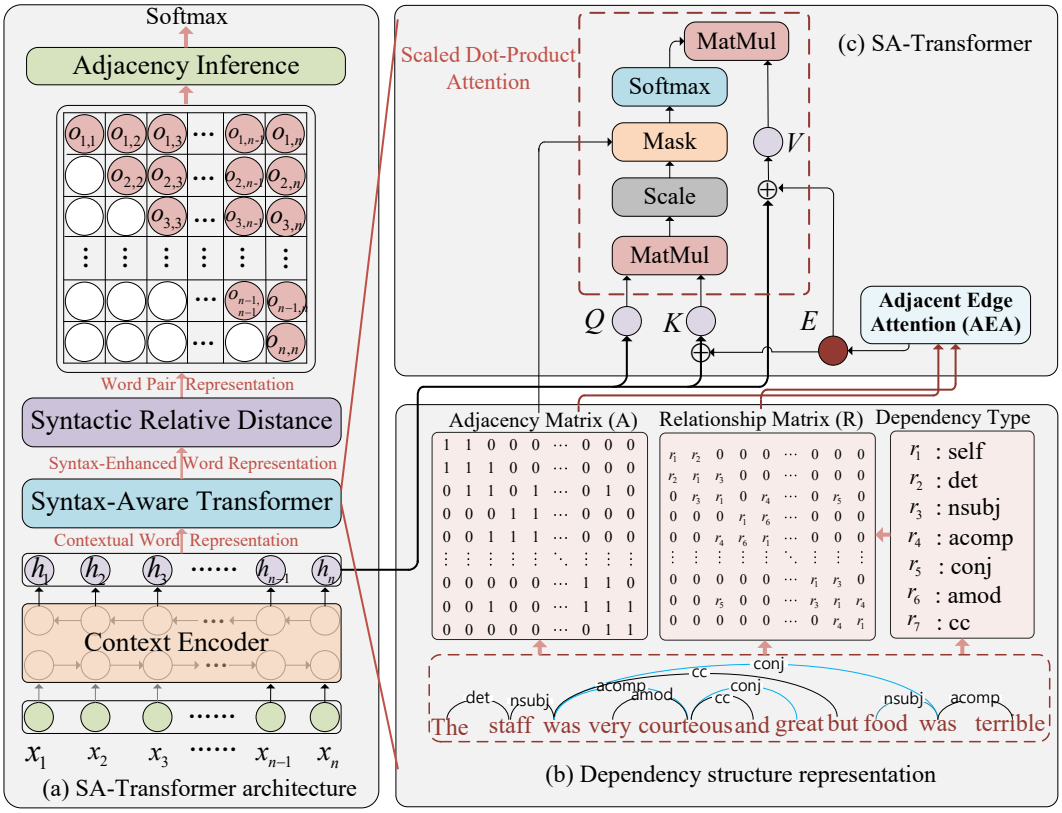


Fig. 2. The overall framework of the proposed SA-Transformer.

to learn dependency information. However, this model only considers the semantic information of syntactic adjacent contexts and ignores edge attributes. Zhao et al. [26] also developed a pointer-specific tagging method to integrate dependency information into GCN for ASTE. A triplet alignment scheme was then proposed to extract triplets by aligning the corresponding positions of the aspect and opinion terms.

3 Syntax-aware Tramsforer Network

Given an input sentence $X = \{x_1, x_2, \cdots, x_n\}$ with n tokens, the goal is to extract a set of opinion triplets $\{(a, o, s)_\omega\}_{\omega=1}^{\Omega}$, where $(a, o, s)_\omega$ is the ω-th opinion triplet, which consists of an aspect term of length $a_\omega = \{x_{l_a}$, an opinion term $o_\omega = \{x_{l_o}, \cdots, x_{r_o}\}$ of length $r_o - l_o + 1$, and the corresponding sentiment polarity $s \in \{\text{Pos}, \text{Neg}, \text{Neu}\}$. Table 2 lists the notations used throughout the paper.

Fig. 2(a) shows the overall architecture of the proposed method, which is composed of four parts: a context encoder, SA-Transformer, syntactic relative distance, and adjacent inference strategy. The context encoder is used to produce the contextual word representations for an input sentence. SA-Transformer then uses dependency parsing to obtain the dependency structure of the sentence and represents it using an adjacency matrix and relationship matrix to record whether an edge exists between two words and the kind of dependency type, as shown in Fig. 2(b). Both matrices are used by the AEA to learn the edge representations (E) for each edge based on the dependency types of its adjacent edges, as shown in Fig. 2(c). The edge representations (E) are then added into the key (K) and value (V) of a scaled dot-product attention to be integrated into the contextual word representations. The syntax-enhanced representations of any two words can then be used to constitute a word-pair representation. In addition, the distance between the two words is calculated by their syntactic relative distance

TABLE 2
Notations used in the paper and their descriptions.

Notations	Descriptions
W	The trainable weight matrix.
b	The trainable biases.
x_i	The input of the ith word.
w_i	The embedding of the ith word.
h_i	The hidden state of the ith word.
A	The adjacency matrix of the input sentence.
R	The relationship matrix of the input sentence.
$r_{i,j}$	The dependency type between the words x_i and x_j.
$z_{i,j}$	The initial edge embedding of words i to j.
$e_{i,j}$	The final edge representation of the words x_i and x_j.
D_i^g	The ith word representation in the gth attention head
$\widetilde{e}_{i,j}^i$	The edge representation of $e_{i,j}$ learned from its adjacent of the e_i.
$U_{i,j}^{i,m}$	The edge representation of the mth attention head in AEA.
$S_i^{(l)}$	The ith word representation in the lth SA-Transformer layer.
$f_d(i,j)$	The representation of the syntactic relative distance between words i and j.
$o_{i,j}$	The final representation of the word pair x_i to x_j.
$p_{i,j}^T$	The final tag probability distribution of the word pair x_i to x_j.

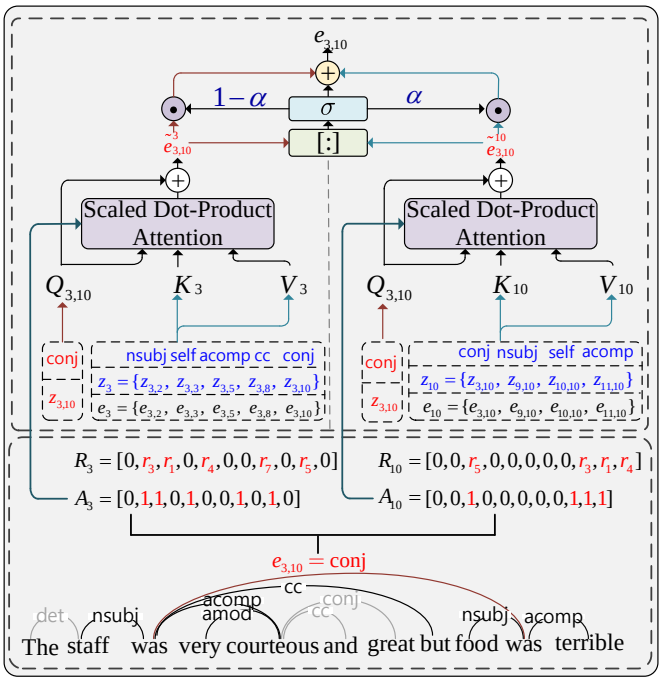


Fig. 3. Illustrative example of AEA to learn edge representations with dependency types. The dependency type "self" denotes the edge between a word itself.

in a dependency tree and encoded into the word-pair representation as an extra feature. Finally, the adjacent inference strategy is used to iteratively predict the tag of each word pair from those of its adjacent word pairs. The details of each component are described as follows.

3.1 Context Encoder

To obtain the contextual word representations for each sentence, 300-dimensional GloVe [45] vectors are used as the initial word embeddings $\{w_1, w_2, \cdots, w_N\}$, where w_i denotes the word vector of word x_i. A bidirectional LSTM (BiLSTM) model [46] is then used as a context encoder to produce the hidden representations of the word vectors, defined as

$$\begin{aligned}(\overrightarrow{h}_i, \overrightarrow{c}_i) &= \text{LSTM}(w_i, \overrightarrow{h}_{i-1}, \overrightarrow{c}_{i-1}) \\ (\overleftarrow{h}_i, \overleftarrow{c}_i) &= \text{LSTM}(w_i, \overleftarrow{h}_{i+1}, \overleftarrow{c}_{i+1})\end{aligned} \tag{1}$$

where $\overrightarrow{h}_i \in \mathbb{R}^{d_h/2}$ and $\overleftarrow{h}_i \in \mathbb{R}^{d_h/2}$ respectively denote the forward and backward hidden representations of w_i, $\overrightarrow{c}_i$ and $\overleftarrow{c}_i$ respectively denote the forward and backward LSTM unit states, and d_h denotes the dimensionality of the hidden representations. The forward and backward hidden representations are then concatenated to comprise the final hidden representations, defined as

$$h_i = [\overrightarrow{h}_i : \overleftarrow{h}_i] \tag{2}$$

where $h_i \in \mathbb{R}^{d_h}$ denotes the final hidden representations of w_i, and $[:]$ denotes a concatenation operation.

3.2 SA-Transformer

Once we obtain the contextual hidden representations of each word, SA-Transformer encodes syntactic dependency information into them in three steps: representation of the dependency structure, learning of edge representations with dependency types using AEA, and injection of edge representations into contextual representations. The details of each step are described as follows.

Dependency Structure Representation. A given sentence is first parsed as a dependency tree. Each dependency is represented as a tuple $(x_i, x_j, r_{i,j})$, where $r_{i,j}$ denotes the dependency type between the words x_i and x_j. The dependency structure can then be represented as an adjacency matrix A and relationship matrix R, where $A = \{a_{i,j} \in \{0,1\}\} \in \mathbb{R}^{n\times n}$ records whether an edge exists between two words, and $R = \{r_{i,j}\} \in \mathbb{R}^{n\times n}$ records the dependency type of each edge. Both A and R are symmetric matrices.

Adjacent Edge Attention (AEA). Both adjacency matrix $A = \{a_{i,j}\} \in \mathbb{R}^{n\times n}$ and relationship matrix $R = \{r_{i,j}\} \in \mathbb{R}^{n\times n}$ are taken as input to learn the edge representations $E = \{e_{i,j}\} \in \mathbb{R}^{n\times n\times d}$, where $e_{i,j} \in \mathbb{R}^d$ denotes the representation of the edge between words x_i and x_j, and d is the dimensionality of the edge representations. To accomplish this goal, an embedding layer is first applied to map $R = \{r_{i,j}\} \in \mathbb{R}^{n\times n}$ to obtain the initial edge embeddings $Z = \{z_{i,j}\} \in \mathbb{R}^{n\times n\times d_z}$, where $z_{i,j} \in \mathbb{R}^{d_z}$ denotes the initial edge embedding of $e_{i,j} \in \mathbb{R}^d$, and d_z is the dimensionality of the initial edge representations. Each edge representation $e_{i,j}$ is determined based on the dependency types of the edges adjacent to x_i and x_j. In the example shown in Fig. 3, to learn the edge representation $e_{3,10} = \text{conj}$, the AEA first looks up the matrices $A = \{a_{i,j}\}$ and $R = \{r_{i,j}\}$ to identify the edge representations adjacent to $x_3 = was$, i.e., $e_3 = \{e_{3,2}, e_{3,3}, e_{3,5}, e_{3,8}, e_{3,10}\}$ with dependency types {nsubj, self, acomp, cc, conj}, and those adjacent to $x_{10} = was$, i.e., $e_{10} = \{e_{3,10}, e_{9,10}, e_{10,10}, e_{11,10}\}$ with dependency types {conj, nsubj, self, acomp}. AEA then takes the initial edge embeddings $z_{3,10}$, z_3 and z_{10} as inputs, uses scaled dot-product attention to learn the hidden edge representation $\widetilde{e}^3_{3,10}$ based on e_3 and $\widetilde{e}^{10}_{3,10}$ based on e_{10}, and finally uses a gate function to combine $\widetilde{e}^3_{3,10}$ and $\widetilde{e}^{10}_{3,10}$ as the final representation of $e_{3,10}$. By considering the dependency types of the adjacent edges, even two edges with the same dependency type can have different representations and weights.

The formal description of AEA is presented as follows. Let $z_{i,j}$ and z_i be the initial edge embeddings of $e_{i,j}$ and e_i, respectively (i.e., the adjacent edge representations of x_i); thus, the AEA learns the hidden representation $\widetilde{e}^i_{i,j}$ as

$$\widetilde{e}^i_{i,j} = AEA(z_{i,j}, z_i) = [U^{i,1}_{i,j}, U^{i,2}_{i,j}, \cdots, U^{i,M}_{i,j}]W_z \tag{3}$$

where $\widetilde{e}^i_{i,j} \in \mathbb{R}^{d_z}$ denotes the hidden representation of $e_{i,j}$ learned from its adjacent edge representations e_i, $W_z \in \mathbb{R}^{d_z\times d_z}$ is a trainable weight matrix, and $U^{i,m}_{i,j} \in \mathbb{R}^{1\times d_e}$ in $[U^{i,1}_{i,j}, U^{i,2}_{i,j}, \cdots, U^{i,M}_{i,j}]$ denotes the edge representation learned by the m-th attention head of scaled dot-product attention, defined as

$$U^m_{i,j} = \text{softmax}(\text{A}_i \cdot \frac{\widetilde{Q}^m_{i,j}(\widetilde{K}^m_i)^T}{\sqrt{d_e}})\widetilde{V}^m_i \tag{4}$$

$$\begin{aligned} \widetilde{Q}_{i,j}^{m} &= W_{\widetilde{Q}} z_{i,j} \\ \widetilde{K}_{i}^{m} &= W_{\widetilde{K}} z_{i} \\ \widetilde{V}_{i}^{m} &= W_{\widetilde{V}} z_{i} \end{aligned} \tag{5}$$

where $\widetilde{Q}_{i,j}^{m} \in \mathbb{R}^{1\times d_e}$ denotes a query regarding the current edge representation $z_{i,j}$, $\widetilde{K}_{i}^{m} \in \mathbb{R}^{n\times d_e}$ and $\widetilde{V}_{i}^{m} \in \mathbb{R}^{n\times d_e}$ respectively denote the key and value both regarding the adjacent edge representations z_i of $z_{i,j}$, $W_{\widetilde{Q}} \in \mathbb{R}^{d_e\times d_z}$, $W_{\widetilde{K}} \in \mathbb{R}^{d_e\times d_z}$, and $W_{\widetilde{V}} \in \mathbb{R}^{d_e\times d_z}$ are trainable weight matrices, $d_e = d_z/M$ is the dimensionality of the edge representations in each head, $A_i \in \mathbb{R}^{n\times 1}$ denotes a mask vector used to help $\widetilde{Q}_{i,j}^{m}$ query the key $\widetilde{K}_{i}^{m}$ to identify the edges connected to x_i in the value $\widetilde{V}_{i}^{m}$, and $\mathrm{softmax}(\cdot)$ denotes the attention weights for these adjacent edges of x_i, which can be obtained in the training process according to their contribution to learning the current edge representation. The attention weight is then used to aggregate the adjacent edge representations of x_i in $\widetilde{V}_{i}^{m}$ to generate the edge representation of the m-th attention head $U_{i,j}^{i,m}$. By concatenating the edge representations of all attention heads using Eq. (3), the hidden representation $\widetilde{e}_{i,j}^{i}$ can be obtained.

Similarly, the hidden representation $\widetilde{e}_{i,j}^{j}$ can be learned from e_j (i.e., the adjacent edge representations of x_j) using Eqs. (3)-(5) with $z_{i,j}$ and z_j as inputs. Once $\widetilde{e}_{i,j}^{i}$ and $\widetilde{e}_{i,j}^{j}$ are obtained, the AEA uses a gate function to combine the two hidden edge representations as the final representation of $e_{i,j}$. That is,

$$e_{i,j} = \alpha \widetilde{e}_{i,j}^{i} + (1-\alpha)\widetilde{e}_{i,j}^{j} \tag{6}$$

$$\alpha = \sigma(W_r([\widetilde{e}_{i,j}^{i} : \widetilde{e}_{i,j}^{j}]) + b_r) \tag{7}$$

where α is a combination coefficient, σ is the sigmoid activation function, [:] is a concatenation operation, and $W_r \in \mathbb{R}^{1\times 2d}$ and $b_r \in \mathbb{R}^{1}$ are the trainable weight and bias, respectively.

SA-Transformer. Once the edge representations $E = \{e_{i,j}\}$ are learned according to the dependency types, SA-Transformer adds them into contextual word representations. SA-Transformer is composed of L similar layers, i.e., $S = [S^{(1)}, S^{(2)}, \cdots, S^{(L)}]$. Each layer $S^{(l)} = [S_1^{(l)}, S_2^{(l)}, \cdots, S_n^{(l)}]$ represents the contextual representations of the words in the sentence, which are computed by combining the hidden representations of the current layer $\tilde{S}^{(l)}$ and the output of the previous layer $S^{(l-1)}$ using layer normalization. That is,

$$S^{(l)} = \mathrm{LayerNorm}(\tilde{S}^{(l)} + S^{(l-1)}) \tag{8}$$

Note that $S^0 = h$ is the hidden word representation output by the context encoder. The hidden representations of each layer $\tilde{S}^{(l)}$ are generated by injecting the edge representations $E = \{e_{i,j}\}$ into the output of the previous layer $S^{(l-1)}$, defined as

$$\begin{aligned} \tilde{S}^{(l)} &= \mathrm{SA} - Transformer(S^{(l-1)}, E) \\ &= [D^1, \cdots, D^G]W_s \end{aligned} \tag{9}$$

where $D^g = [D_1^g, D_2^g, \cdots, D_n^g] \in \mathbb{R}^{n\times d_s}$ denotes the word representations of all words learned by the g-th attention head, and $W_s \in \mathbb{R}^{(G\cdot d_s)\times d_h}$ denotes the output linear projection. Each word representation D_i^g in D^g is injected with the edge representations by a scaled dot-product attention, defined as

$$D_i^g = \mathrm{softmax}(A_i \cdot \frac{Q_i^g (K_i^g)^T}{\sqrt{d_s}}) V_i^g \tag{10}$$

$$Q_i^g = W_Q S_i^{(l-1)} \tag{11}$$

$$K_i^g = W_K S^{(l-1)} + \beta W_{e,k} e_i \tag{12}$$

$$V_i^g = W_V S^{(l-1)} + \beta W_{e,v} e_i \tag{13}$$

where $Q_i^g \in \mathbb{R}^{1\times d_s}$ denotes a query regarding $S_i^{(l-1)}$, which represents the current word representation of x_i in the (*l-1*)-th layer; $K_i^g \in \mathbb{R}^{n\times d_s}$ and $V_i^g \in \mathbb{R}^{n\times d_s}$ denote the key and value, respectively, regarding $S^{(l-1)}$ and e_i, which represent the word representations of all words in the (*l-1*)-th layer and adjacent edge representations of x_i, respectively; β is a balance coefficient; $W_Q \in \mathbb{R}^{d_s\times d_h}$, $W_K \in \mathbb{R}^{d_s\times d_h}$, $W_V \in \mathbb{R}^{d_s\times d_h}$, $W_{e,k} \in \mathbb{R}^{d_s\times d}$, and $W_{e,v} \in \mathbb{R}^{d_s\times d}$ are trainable weight matrices; $d_s = d_h/G$ is the dimensionality of the word representations in each head; and $A_i \in \mathbb{R}^{n\times 1}$ denotes a mask vector used to help Q_i^g query K_i^g identify both the words and edges connected to x_i in V_i^g. For each word connected to x_i, the contextual representation in $S^{(l-1)}$ is enhanced by combining its corresponding edge representation in e_i using β. These syntax-enhanced word representations are then aggregated using the attention weight $\mathrm{softmax}(\cdot)$ to generate the word representation of x_i in the g-th attention head D_i^g.

3.3 Syntactic Relative Distance

Once the dependency types are incorporated into the contextual word representations, the syntactic relative distance between words [47] is further introduced as an extra feature to enhance word pair representations. The syntactic relative distance between two words, denoted as $dist(x_i, x_j)$, is calculated by the number of hops in the path from word x_i to x_j in a dependency tree. As the example shows in Fig. 2, there are 4 hops between *great* to *food*, i.e., $dist(great, food) = 4$.

The representation of the syntactic relative distance between words, denoted as $f_d(i,j) \in \mathbb{R}^{d_d}$, is generated using an embedding layer with $dist(x_i, x_j)$ as input, where d_d is the dimensionality of the syntactic relative distance representation. The representation of a word pair (x_i, x_j) is generated based on the representations of the two words output by SA-Transformer, i.e., $S_i^{(L)}$ and $S_j^{(L)}$. The syntactic relative distance representation is then concatenated with the word pair representation to generate the final representation of the word pair, denoted as $S_i^{(L)}$ and $S_j^{(L)}$. The syntactic relative distance representation is then concatenated with the word pair representation to generate the final representation of the word pair, denoted as

$$o_{i,j} = [S_i^{(L)} : S_j^{(L)} : f_d(i,j)] \tag{14}$$

where $o_{i,j} \in \mathbb{R}^{2d_h+d_d}$ denotes the final representation of the word pair (x_i, x_j), and [:] denotes a concatenation operation.

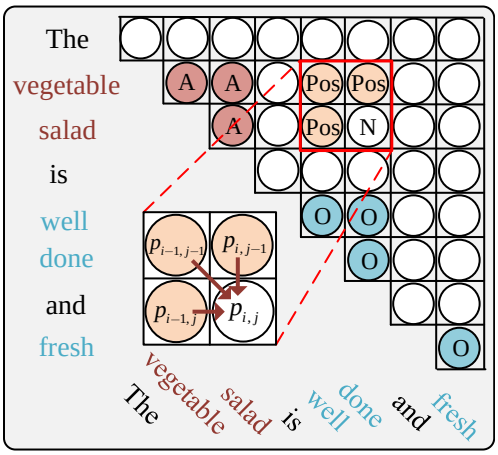


Fig. 4. Illustrative example of the adjacency inference strategy.

TABLE 3
Statistics of datasets (#S, #T, #Pos, #Neu, #Neg, Mean, and Max respectively denote the numbers of sentences, triplets, positive triplets, neutral triplets, negative triplets, mean length and max length.)

Datasets		#S	#T	#Pos	#Neu	#Neg	Mean	Max
Res14	Train	1259	2356	1693	172	491	17	79
	Dev	315	580	427	46	107	17	66
	Test	493	1008	784	68	156	16	70
Lap14	Train	899	1452	808	111	533	19	78
	Dev	225	383	199	48	136	19	83
	Test	332	547	364	67	336	16	71
Res15	Train	603	1038	799	29	210	15	68
	Dev	151	239	181	9	49	15	39
	Test	325	493	324	25	144	16	63
Res16	Train	863	1421	1036	55	330	15	68
	Dev	216	348	263	8	77	15	56
	Test	328	525	416	30	79	15	78

3.4 Adjacency Inference

The last step is to predict the relation tag of each word pair in the sentence as one of the six classes: aspect term (A), opinion term (O), positive (Pos), negative (Neg), neutral (Neu), and no relation (N). Generally, each word pair is predicted independently without considering other word pairs. In fact, other word pairs, such as adjacent word pairs, also contribute to tag prediction, especially for multiword aspect/opinion terms. Considering the example sentence in Fig. 4, both the aspect term *vegetable salad* and opinion term *well done* consist of two words. Each element in the matrix denotes a word pair representation, and the red rectangle contains the word pair representations for the two-word aspect and opinion terms, i.e., (*vegetable*, *well*), (*vegetable*, *done*), (*salad*, *well*), and (*salad*, *done*). Any of the four word pairs can be predicted using the information provided by the other three. For instance, suppose that the model correctly predicts the first three word pairs as (*vegetable*, *well*, Pos), (*vegetable*, *done*, Pos), and (*salad*, *well*, Pos) but incorrectly predicts the last one as no relation (*salad*, *done*, N). The three correctly predicted adjacent word pairs provide useful information that *done* is highly likely to be an opinion term of *salad* with a positive sentiment. The model can thus correct the prediction of (*salad*, *done*, N) as (*salad*, *done*, Pos) in the next prediction iteration.

Based on this notion, we devise an adjacency inference strategy that can predict the tag of each word pair by leveraging the predicted results of its adjacent word pairs to effectively extract the triplets for multiword aspect/opinion terms. Given a word pair (x_i, x_j), the adjacency inference calculates its tag probability distribution of the six classes {A, O, Pos, Neg, Neu, N} using an iterative process, defined as

$$\begin{aligned} p^t_{i,j} &= \gamma^t c^t_{i,j} + (1-\gamma^t)(\widetilde{c}^t_{i,j}) \\ \gamma^t &= \sigma(W_p[c^t_{i,j} : \widetilde{c}^t_{i,j}]) \end{aligned} \tag{15}$$

where $p^t_{i,j}$ denotes the final tag probability distribution of (x_i, x_j) in the t-th iteration, which is calculated by combining its current tag probability distribution $c^t_{i,j}$ and that of its adjacent word pairs $\widetilde{c}^t_{i,j}$ using a balance coefficient γ^t, σ denotes a sigmod function, $W_p \in \mathbb{R}^{1\times 2d_y}$ denotes a trainable weight matrix, and $[:]$ denotes a concatenation operation. The adjacent tag probability distribution $\widetilde{c}^t_{i,j}$ is calculated as

$$\widetilde{c}^t_{i,j} = W_o[c^{t-1}_{i-1,j} : c^{t-1}_{i-1,j} : c^{t-1}_{i-1,j-1}] \tag{16}$$

where $c^{t-1}_{i-1,j}, c^{t-1}_{i-1,j}, c^{t-1}_{i-1,j-1}$ denotes the three adjacent tag probability distributions of (x_i, x_j) in the (t-1)-th iteration, $W_o \in \mathbb{R}^{d_y \times 3d_y}$ is a trainable weight matrix and $d_y = 6$ is the number of tags. The current tag probability distribution $c^t_{i,j}$ is calculated as

$$\begin{aligned} c^t_{i,j} &= \mathrm{softmax}(W_c \widetilde{o}^t_{i,j} + b_c) \\ \widetilde{o}^t_{i,j} &= W_{\widetilde{o}}[\widetilde{o}^{t-1}_{i,j} : p^{t-1}_{i,j}] + b_{\widetilde{o}} \end{aligned} \tag{17}$$

where $\widetilde{o}^t_{i,j}$ denotes the hidden representation of (x_i, x_j) which is initialized by its word pair representation $o_{i,j}$, i.e., $\widetilde{o}^0_{i,j} = o_{i,j}$, $W_c \in \mathbb{R}^{d_y \times d_h}$, and $W_{\widetilde{o}} \in \mathbb{R}^{(2d_h+d_d)\times(2d_h+d_d+d_y)}$ are trainable weight matrices, and $b_c \in \mathbb{R}^{d_y}$ and $b_{\widetilde{o}} \in \mathbb{R}^{2d_h+d_d}$ are trainable biases. After T iterations, the final tag probability distribution of all word pairs is denoted as $p^T = [p^T_{1,1}, p^T_{1,2}, \cdots, p^T_{n,1}, \cdots, p^T_{n,n}]$.

3.5 Training

The training objective is to minimize the cross-entropy error of the ground-truth distribution $\mathcal{Y}_{i,j} \in Y$ and the predicted tag distribution $p^T_{i,j}$ of all word pairs:

$$\mathcal{L}(\theta) = \sum_{\phi=1}^{\Phi} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{Y}^{(\phi)}_{i,j} \log(p^{(\phi),T}_{i,j}|\theta) \tag{18}$$

where Φ and θ respectively denote the number of training samples and all trainable parameters.

4 EXPERIMENTS

4.1 Datasets and Evaluation Metrics

To evaluate the proposed SA-Transformer, four ASTE benchmark datasets were used, including **Rest14**, **Lap14**, **Rest15**, and **Rest16**, which mainly contain consumer reviews of laptop computers and restaurants. These datasets have been used for SemEval-2014 [48], SemEval-2015 [49] and SemEval-2016 [50]. The statistics of the datasets are presented in Table 3.

The precision (P), recall (R), and micro F_1-score (F_1) are used as evaluation metrics for triplet extraction. Compared

TABLE 4
Hyperparameter settings.

Parameter Name	Value
Maximum sequence Length	100
Batch Size	16
Initial Learning Rate	1e-3
Dimension of hidden state d_h	200
Dimension of d_d	100
Dimension of syntactic features d_z	200
The keep dropout rate	0.2

with precision and recall, F_1 is a more appropriate metric because it considers both precision and recall. A triplet is regarded as correctly predicted only if the predicted aspect term, opinion term, and sentiment polarity match the ground-truth aspect term, opinion term, and corresponding polarity, respectively.

4.2 Baselines

The baseline models used for comparison include the pipeline, multitask, and word-pair methods. The implementation details of each method are described as follows.

Pipeline Methods

- **CLMA+** is the extended version of CLMA [16], which proposes a coupled multilayer attention network that can capture both direct and indirect relations between words to coextract aspect and opinion terms. Peng et al. [9] modified this method as CLMA+ by using CLMA in the first stage, followed by pairing the extracted aspect and opinion terms and identifying sentiment polarities to generate triplets.
- **RINATE+** is an extended version of RINATE [17] that uses a weakly supervised method to extract aspect and opinion terms. It uses a set of extraction rules mined based on the dependencies between words to expand the training data for neural model training. Peng et al. [9] modified this method as RINATE+ using the same method as CLMA+.
- **Li-unified-R+** is the extended version of Li-unified [12], a unified method that implements a two-layer stacked LSTM model to extract the aspect terms and their sentiment polarities. Peng et al. [9] modified this method as Li-unified-R+ by additionally extracting the opinion terms in the first stage and pairing the extracted terms to generate triplets in the second stage.
- **TSF** [9] is a two-stage pipeline model for ASTE. In the first stage, it extracts aspect terms, opinion terms and sentiment polarities using the mutual influence between aspect and opinion terms. A classifier is then used to pair the extracted terms to generate triplets in the second stage.

Multitask Methods

- **OTE-MTL** [18] uses a multitask learning framework to jointly extract aspect terms, opinion terms and sentiment polarities. It first uses a sequence tagging strategy to extract aspect and opinion terms, then predicts the sentiment polarities using a table filling method, and finally applies a decoding process to generate triplets based on heuristic rules.
- **BMRC** [19] proposes a bidirectional machine reading comprehension framework with multiturn queries that are designed to gather information useful for extracting the aspect terms, opinion terms and sentiment polarities. The bidirectional structure can further ensure that information can be gathered from both the aspect-to-opinion and opinion-to-aspect directions.
- **Span-ASTE** [20] proposes a span-level model that can capture the span-to-span interactions instead of word-to-word interactions between the aspects and opinions for ASTE. It first enumerates all possible aspect and opinion spans, then uses a dual-channel span pruning strategy to filter out the invalid spans, and finally determines the sentiment relations between each valid aspect span and opinion span.
- **DE-OTE-BISDD** [21] presents a method based on double embeddings and bidirectional sentiment-dependence detection. The double embeddings fuse character- and word-level embeddings to obtain sentence representations. Multitask learning is then applied to extract aspect and opinion terms, using the bidirectional sentiment-dependence detector to determine the sentiment polarities by leveraging information gathering from both aspect-to-opinion and opinion-to-aspect directions.
- **CopyMTL** [22] presents a method to extract multiple and overlapped triplets using a span copy mechanism and a dual decoder. The span copy mechanism can capture the multitoken aspect and opinion words through multihead attention. The dual decoder is used to generate aspect and opinion words separately based on multitype information.

Word-Pair Methods

- **GTS** [23] pioneered the use of a grid tagging scheme for ASTE. It first enumerates all possible word pairs in a sentence and represents them as a grid. A classifier is then used to classify the relation tags of the word pairs to generate the triplets.
- **JET** [24] converts the ASET task into a structured prediction problem with a position-aware tagging scheme to jointly extract triplets. It develops a joint extraction model based on conditional random field (CRF) and semi-Markov CRF, which can effectively capture the interactions among aspect terms, opinion terms and sentiment polarities based on factorized features.
- **$\mathbf{S^3E^2}$** [25] proposes a graph neural network to exploit the semantic and syntactic information for ASTE. It first uses BiLSTM to learn the contextual semantics of sentences and then encodes the syntactic dependencies between words into graph representations to jointly extract the triplets.
- **MAS** [26] integrates syntactic dependencies into graph neural networks for ASTE. It proposes a pointer-specific tagging method to identify the relationships between the aspect and opinion terms. A

TABLE 5
Experimental results for triplet extraction. Each model was run five times to report the average result. The best scores are in bold and the second best are underlined.

Model		Rest14			Lap14			Rest15			Rest16		
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Pipline Models	RINANTE+	31.07	37.63	34.03	23.10	17.60	20.00	29.40	26.90	28.00	27.10	20.50	23.30
	CMLA+	40.11	46.63	43.12	31.40	34.60	32.90	34.40	37.60	35.90	43.60	39.80	41.60
	Li-unified-R+	41.44	68.79	51.68	42.25	42.78	42.47	43.34	50.73	46.69	38.19	53.47	44.51
	TSF	44.18	62.99	51.89	40.40	47.24	43.50	49.97	54.68	46.79	46.76	62.97	53.62
Multitask Methods	OTE-MTL	64.54	55.57	59.67	54.18	45.20	48.97	58.16	54.02	55.83	48.17	42.43	45.05
	BMRC	-	-	63.32	-	-	48.15	-	-	53.77	-	-	63.16
	Span-ASTE	**72.52**	62.43	67.08	59.85	45.67	51.80	64.29	52.12	57.56	67.25	61.75	64.37
	DE-OTE-BISDD	68.57	59.17	63.53	56.17	46.20	50.70	61.54	48.43	54.21	65.20	61.34	63.21
	CopyMTL	64.25	63.85	64.05	48.55	47.72	48.13	54.81	55.49	55.14	64.33	**65.61**	64.96
Word-Pair Methods	GTS	67.28	61.91	64.48	59.42	45.13	51.30	63.26	50.71	56.29	66.07	65.05	65.56
	JET	61.50	55.13	58.14	53.03	33.89	41.35	64.37	44.33	52.50	70.94	57.00	63.21
	MAS	70.70	64.20	67.30	60.50	47.10	53.00	**64.70**	53.70	58.70	67.40	63.30	65.30
	S^3E^2	69.08	64.55	66.74	59.43	46.23	52.01	61.06	56.44	58.66	71.08	63.13	66.87
	SA-Transformer	70.76	**65.85**	**68.22**	**61.28**	**48.98**	**54.44**	62.82	**58.31**	**60.48**	**72.01**	62.87	**67.13**

TABLE 6
Ablation study results of the proposed method. Each model was run five times to report its average result.

Model	Rest14			Lap14			Rest15			Rest16			Training Time(s)	Test Time(s)
	P	R	F1	P	R	F1	P	R	F1	P	R	F1		
Proposed Model	70.76	65.85	68.22	61.28	48.98	54.44	62.82	58.31	60.48	72.01	62.87	67.13	4.28	0.85
GTS	67.28	61.91	64.48	59.42	45.13	51.30	63.26	50.71	56.29	66.07	65.05	65.56	2.58	0.51
Proposed Model w/o SA-Trans	68.01	62.32	65.03	59.32	45.77	51.67	63.01	50.59	56.12	66.33	64.74	65.53	2.79	0.54
Proposed Model w/o SA	68.56	62.78	65.54	59.18	46.14	51.85	62.35	53.13	57.37	67.81	64.21	65.96	3.62	0.71
Proposed Model w/o AEA	69.72	64.03	66.75	60.12	46.97	52.74	61.79	56.13	58.82	70.52	62.78	66.42	3.82	0.78
Proposed Model w/o SRD	70.12	64.58	67.23	60.77	47.69	53.44	62.05	57.96	59.94	71.01	62.32	66.38	4.12	0.83
Proposed Model w/o AF	70.25	65.12	67.59	60.29	48.19	53.57	60.79	58.21	59.47	71.29	62.32	66.48	3.74	0.75

triplet alignment scheme is then designed to extract triplets by aligning the corresponding positions of the aspect and opinion terms.

4.3 Implementation Details

The 300-dimensional GloVe [45] vectors were used as the initial word embedding, and the uniform distribution of U (-0.25, 0.25) was initialized to words that do not appear in the GloVe vectors. The dimension d_h of the hidden state was set to 200. The spaCy[1] with the **en_core_web_trf** version was used to parse each given sentence into a dependency tree and then build both a relationship matrix and an adjacency matrix from the dependency tree. Adam [51] was used as the optimizer with a maximum learning rate of 1e-3 and a decay factor of 0.5. The dimensions of the syntactic relative distance embedding d_d and syntactic dependency features d_z in AEA were 100 and 200, respectively, which were initialized by the uniform distribution of U(-0.5, 0.5). The grid search strategy was implemented to select the optimal values for the model hyperparameters. We ran each model five times and report the average results. Table 4 summarizes the hyperparameter settings of the proposed method. The code of this paper is available at: https://github.com/YuanLi95/SA-Transformer-for-ASTE.

4.4 Comparative Results

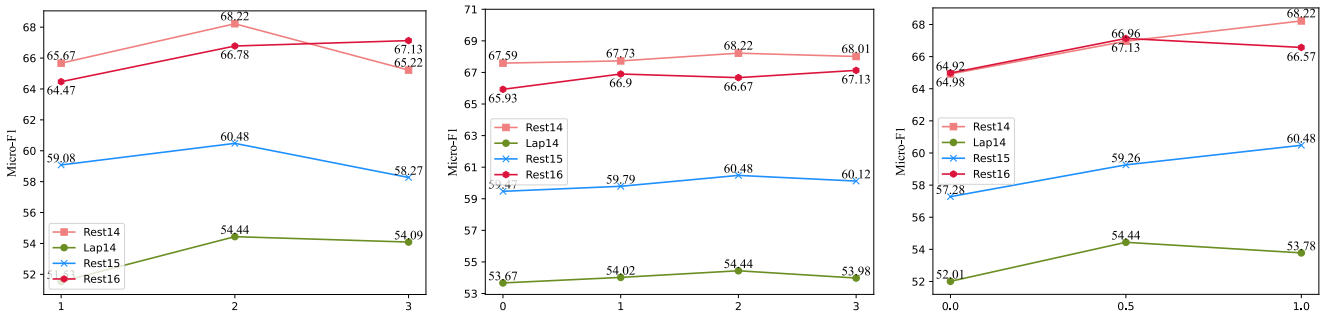
Table 5 summarizes the comparative results of the proposed model and previous methods in terms of P, R, and F_1. For F_1, both the multitask (OTE-MTL, BMRC, Span-ASTE, DE-OTE-BISDD, and CopyMTL) and word-pair models (GTS, JET, S^3E^2, and MAS) notably outperformed the pipeline models (RINANTE+, CMLA+, Li-unified-R+, and TSF) for all datasets since the joint prediction of subtasks can significantly address the error propagation in the pipeline models. In addition, the word-pair models outperformed the multitask models for most datasets, indicating that word-pair classification can effectively extract the relationships for the nested labels.

The proposed SA-Transformer outperformed the previous methods with respect to F_1 for all datasets. There are three possible reasons to explain this. First, SA-Transformer incorporates the knowledge of dependency types into contextual word representations and thus can effectively reduce the number of syntactically irrelevant word pairs. Second, AEA enables the model to learn an appropriate representation for the dependency type of each edge, and the syntactic relative distance further learns the syntactic and positional information between words. Third, the adjacent inference can iteratively refine the predicted tag distribution of each word pair according to those of its adjacent word pairs and thus can more effectively extract the triplets for multiword aspect/opinion terms.

4.5 Ablation Studies

Ablation studies were conducted to investigate the effectiveness of each component in the proposed model: SA-Transformer, adjacent edge attention (AEA), syntactic rel-

1. https://spacy.io/models



(a) Layer number L of SA-Transformer. (b) Number of inferences T. (c) Balance coefficient β.

Fig. 5. The effect of different parameters on different datasets.

TABLE 7
Effects of using different parsing toolkits for triplet extraction.

Parsing Toolkits	English PTB	F1			
		Rest14	Lap14	Rest15	Rest16
Random	-	57.92	46.06	49.17	55.45
Deep biaffine	95.74	67.75	54.01	60.85	66.57
En_core_web_sm	92.01	66.48	53.49	59.36	66.26
En_core_web_trf	95.1	68.22	54.44	60.48	67.13

ative distance (SRD), and adjacent inference (AF). Table 6 shows the ablation results with the GTS model as the baseline. The various ablation models produced different degrees of performance decline, indicating that each component makes its own unique contribution to the proposed model. By removing the entire SA-Transformer (w/o SA-Trans), i.e., removing both the syntactic dependency module and transformer architecture, the proposed method was degraded to become similar to the GTS model, which thus caused the largest performance decline. Instead of removing the entire SA-Transformer, we replaced the SA-Transformer with the vanilla transformer [52] to retain the transformer architecture while removing the syntactic dependency module (w/o SA). This also resulted in a sharp performance decline, indicating that encoding the dependency type information into the weight and distribution can improve the model's ability to learn the relationships between word pairs.

In addition, the removal of AEA (w/o AEA) also resulted in a decline in performance because AEA can learn appropriate edge representations to achieve more accurate graph propagation. Although the model can work properly without SRD and AF, the performance still decreased because SRD can further capture syntactic and positional information, and AF can better handle multiword aspect/opinion terms.

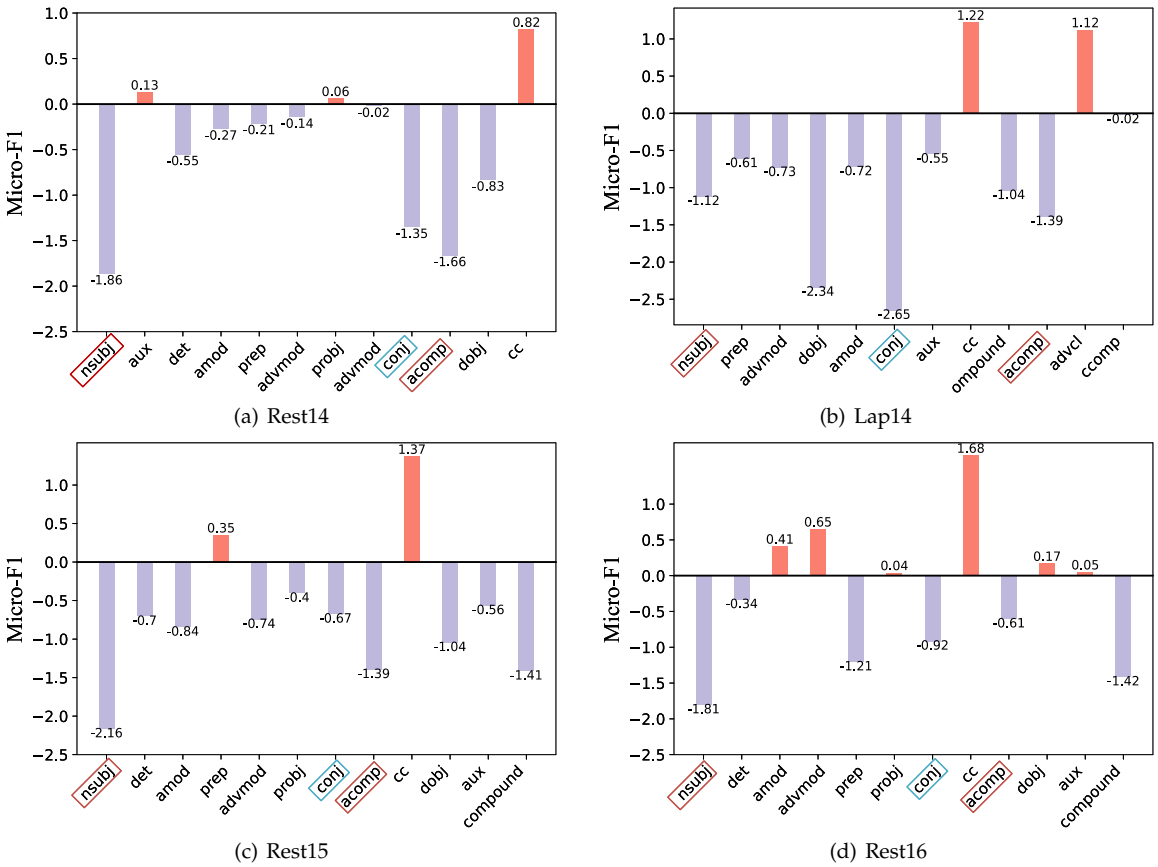
To further investigate the computational cost of each component, the last two columns in Table 6 show the average training and test times per epoch across all datasets for each component. For w/o SA-Trans, the computational cost was reduced by 46% (1.49 seconds) because of removal of the entire SA-Transformer, which thus required a lower computational cost similar to that of GTS. For w/o SA, the computational cost was reduced by 15% (0.66 second) because of removal of the syntactic dependency module, namely, both the adjacency matrix and relationship matrix used for dependency structure representation and their related operations. For w/o AEA, the computational cost was reduced by 11% (0.46 second) which is lower than that of w/o SA because only the adjacency matrix and its related operations were removed. Once the adjacency matrix is removed, the weight and representation of each edge can only be learned using the relationship matrix according to its contribution to the prediction and cannot be learned from its adjacent edges. For w/o SRD, it produced the least reduction in computational cost of 4% (0.16 second) among all components, indicating that calculating the syntactic relative distance to enhance the word pair representation is efficient. For w/o AF, it reduced computational cost by 13% (0.54 second) because it removed the inference strategy that can predict from the adjacent word pairs.

4.6 Effects of Dependency Parsing

To investigate the effects of using different dependency parsing toolkits for triplet extraction, we selected three dependency parsers including **Deep biaffine** [53], **En_core_web_sm** and **En_core_web_trf**. **Deep biaffine** uses biaffine attention to predict the dependencies and their types between words. **En_core_web_sm** and **En_core_web_trf** represent different versions of the Spacy toolkit, which respectively use the token2vector and a transformer such as RoBERTa [54] as the context encoder. A **random** parser was also implemented to randomly generate a dependency tree for each sentence. Since **En_core_web_trf** was used in the previous experiments, this experiment replaced it with the other three parsers to rerun the experiment on triplet extraction. Table 7 shows the comparative results. The parsing performance on the English Penn Treebank (PTB) is also provided for reference. The results show that the three parsers **Deep biaffine**, **En_core_web_sm**, and **En_core_web_trf** achieved comparable results, and all significantly outperformed **Random**. This indicates that randomly generated dependencies and their types contain many errors that degrade the extraction performance, while the parsed results of the other three parsers can still maintain the extraction performance at a certain level.

4.7 Effects of Parameters

Since we used L layers in SA-Transformer, we investigate the effect of the number of layers on the performance of the proposed model, as presented in Fig. 5(a). As indicated, the



(a) Rest14 (b) Lap14 (c) Rest15 (d) Rest16

Fig. 6. Performance change of removing different dependency types.

best performance was achieved at L=2 on **Rest14**, **Lap14**, and **Rest15**, and L=3 on **Rest16**.

Furthermore, we investigate the effect of the number of inferences T over the range of 0 to 3 for all datasets. As presented in Fig. 5(b), the best performance was achieved at T=2 on **Rest14**, **Lap14**, and **Rest15** and T=3 on **Rest16**. T=0 means that AF is not used and the proposed model will degenerate to SA-Transformer w/o AF, thus performing worst for all datasets.

In addition, Fig. 5(c) presents the influence of the balance coefficient β in Eqs.(12) and (13). $\beta = 0$ means that none of the syntactic dependency information is incorporated into the contextual word representations, thus performing worst for all datasets. The best performance was achieved at $\beta = 0.5$ on **Lap14** and **Rest16** and $\beta = 1$ on **Rest14** and **Res15**.

4.8 Effects of Dependency Types

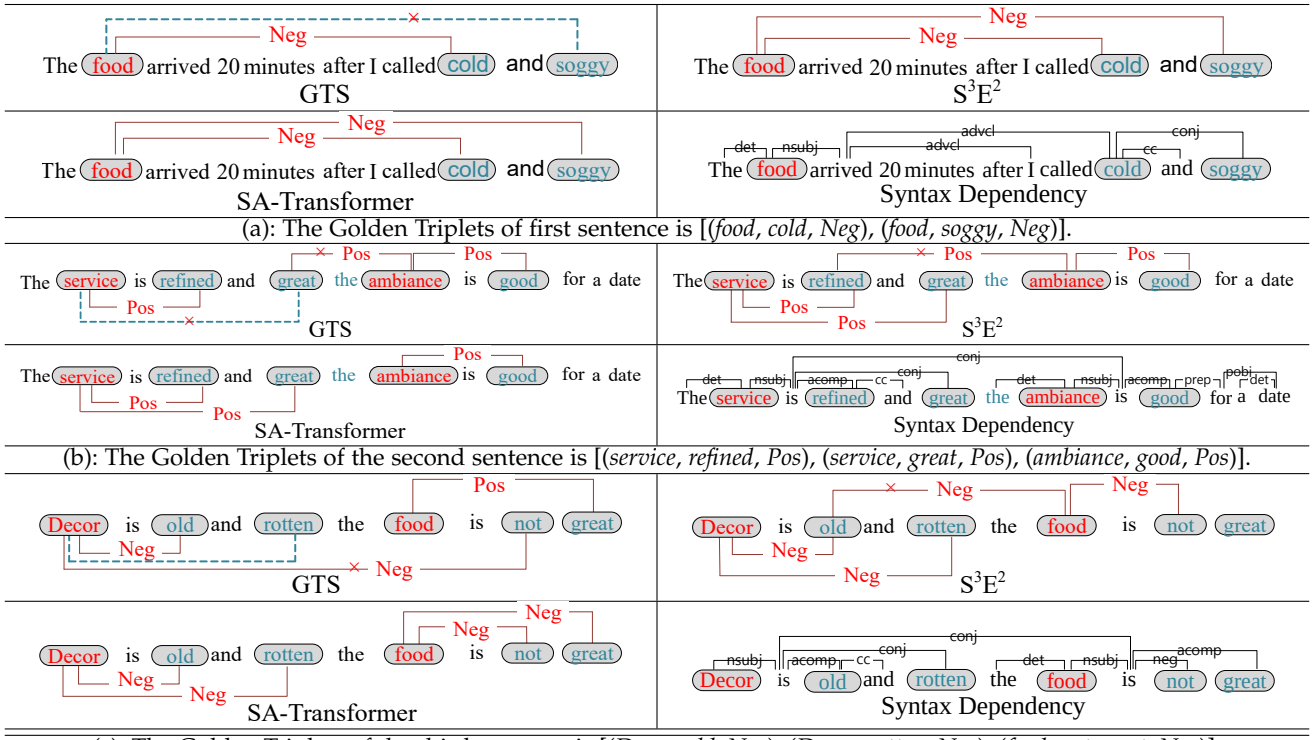
Different dependency types may yield different contributions to prediction performance. To investigate their effects, we removed one dependency type at a time to examine the performance change. Fig. 6 shows the change in F_1-scores after removing the top 12 most frequently occurring dependency types in the datasets. The results show that most dependency types (e.g., *nsubj*, *acomp*, *conj* etc.) yield a positive contribution because removing them led to a certain degree of performance decline. Only selected dependency types (e.g., *cc*) caused a negative contribution to performance. For example, the dependency types *nsubj* and *acomp* are highly useful features because they can capture the subject-object relation between the aspect and opinion words (e.g., (*staff*, *courteous*) and (*food*, *terrible*) in Fig. 1). Conversely, the dependency type *cc* typically captures redundant relations (e.g., (*was*, *but*) and (*courteous*, *and*) in Fig. 1).

4.9 Case Study

To further explain the effectiveness of the proposed SA-Transformer, three test examples were selected from **Rest14** and **Rest15** for the case study. Table 8 shows the golden triplets, the predicted triplets of GTS, S^3E^2 and our model, and the dependency structure of the three test examples. In the first example, GTS correctly extracts the aspect term *food* with the opinion term *cold* and the corresponding sentiment polarity. However, the word pair (*food*, *soggy*) is not considered a triplet because the distance between *soggy* and *food* is too far. It is difficult for GTS to effectively learn the potential relationship between them. S^3E^2 and our model correctly predict all the triplets because both incorporate syntactic dependencies and thus can effectively aggregate syntax-related information during prediction.

In the second example, the irrelevant context word *great* is equally close to the aspect term *ambiance* as it is to the opinion term *good*. GTS regards both *great* and *good* as opinion terms for *ambiance*, thus producing the incorrect triplet (*ambiance*, *great*, Pos). The same situation occurs for S^3E^2. After several iterations, GTS associates the aspect term *ambiance* with the irrelevant word *refined*, thus mistakenly predicting (*ambiance*, *refined*, Pos) as a triplet. The

TABLE 8
Case study. The aspect and opinion terms are respectively highlighted in orange and blue. The red line and the blue dotted line respectively indicate the predicted triplets and missed correct triplets.



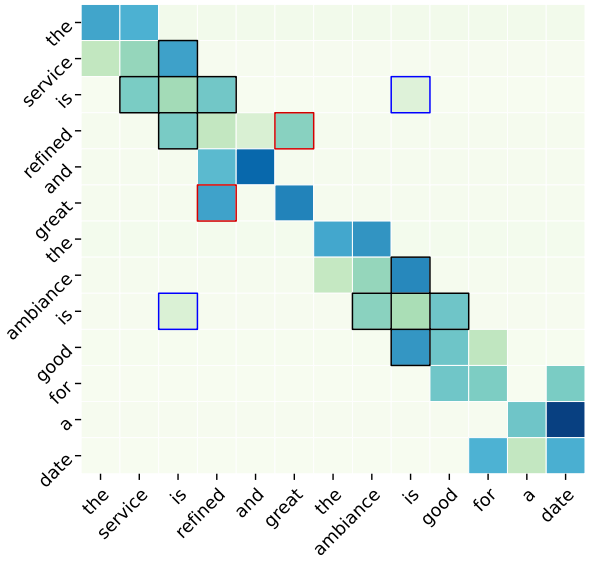


Fig. 7. Visualization of the attention of a given sentence.

proposed SA-Transformer model can avoid the generation of the incorrect triplet (*ambiance*, *refined*, Pos) because it can assign different weights to different edges even if they have the same dependency type. For instance, it can assign a lower weight to the **conj** between *is* and *is* to block inappropriate propagation from *refined* to *ambiance*. It can also assign a higher weight to the **conj** between *is* and *great* to successfully aggregate information between *service* and *great*.

In the third example, the opinion term *not great* consists of multiple words. Both GTS and S^3E^2 fail to completely extract the opinion term. SA-Transformer can do this because it applies adjacent inference to deal with multiword aspect/opinion terms. For this case, the sentiment polarity of the word pair (*food*, *great*) is correctly predicted as negative by learning the predicted result of its adjacent word pair (*food*, *not*).

4.10 Visualization

To further demonstrate how SA-Transformer improves the ASET task, we select the second example in Table 8 to visualize the attention weights of its words, as shown in Fig. 7. For the example sentence, SA-Transformer correctly predicts the tag of (*service*, *refined*) as positive because it assigns a higher weight to **nsubj** and **acomp** (black lines) and thus can effectively align *service* with *refined* through two graph propagation iterations. The above situation also occurs in the case of (*ambiance*, *good*). On the other hand, although the two edges **conj** between *is* and *is* and **conj** between *refined* and *great* have the same dependency type, they are assigned different weights (respectively lower and higher). This example demonstrates that learning edge representations for each edge by querying its adjacency edges can obtain more appropriate weights and representations and thereby result in more accurate graph propagation.

5 CONCLUSIONS

This study proposes a syntax-aware transformer that can encode dependency type information into both edge and

word representations to improve graph neural networks for ASTE. By encoding the dependency types into edge representations, the proposed method can learn different representations and weights for different edges, even for those with the same dependency type, thus achieving more accurate graph propagation. Incorporating edge representations into contextual word representations can further learn syntactic and positional relationships between words to enhance word pair representations. To effectively extract triplets for multiword aspect/opinion terms, an adjacency inference strategy is developed to iteratively predict the tag of each word pair from the predicted results of its adjacent word pairs. Experiments on four benchmark datasets demonstrate the effectiveness of the proposed method. A series of experiments was also conducted for in-depth analysis, including an ablation study that showed that each component contributes to triplet extraction; a dependency parsing experiment that examined the effects of using different dependency parsing toolkits on extraction performance; a case study that presented several missed and correctly extracted triplets to discuss the effectiveness and limitations of different methods; and a visualization experiment that illustrated the attention weights of each dependency type for an example sentence to explain how the proposed method can accomplish proper graph propagation through weight assignment.

Future work will focus on incorporating other useful external knowledge to improve graph propagation and consider long-range information between word pairs to extend the adjacent inference strategy. Another direction is to investigate recent advancements in ABSA tasks such as large language models (LLMs) [55], prompt-based methods [56] and neurosymbolic AI frameworks [57] to improve ASTE.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under grant no. 61966038, in part by the Ministry of Science and Technology, Taiwan, ROC, under grant no. MOST110-2628-E-155-002. The authors would like to thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] K. Schouten and F. Frasincar, "Survey on Aspect-Level Sentiment Analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813–830, 2016.

[2] A. Nazir, Y. Rao, L. Wu, and L. Sun, "Issues and challenges of aspect-based sentiment analysis: A comprehensive survey," *IEEE Transactions on Affective Computing*, vol. 13, no. 2, pp. 845–863, 2022.

[3] H. Liu, I. Chatterjee, M. Zhou, X. S. Lu, and A. Abusorrah, "Aspect-based sentiment analysis: A survey of deep learning methods," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 6, pp. 1358–1375, 2020.

[4] S. Poria, D. Hazarika, N. Majumder, and R. Mihalcea, "Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research," *IEEE Transactions on Affective Computing*, vol. 14, no. 1, pp. 108–132, 2023.

[5] M. Soleymani, D. Garcia, B. Jou, B. Schuller, S. F. Chang, and M. Pantic, "A survey of multimodal sentiment analysis," *Image and Vision Computing*, vol. 65, pp. 3–14, 2017.

[6] I. Chaturvedi, E. Cambria, R. E. Welsch, and F. Herrera, "Distinguishing between facts and opinions for sentiment analysis: Survey and challenges," *Information Fusion*, vol. 44, pp. 65–77, 2018.

[7] F. Z. Xing, E. Cambria, and R. E. Welsch, "Natural language based financial forecasting: a survey," *Artificial Intelligence Review*, vol. 50, no. 1, pp. 49–73, 2018.

[8] G. Y. Zhou and J. X. Huang, "Modeling and mining domain shared knowledge for sentiment analysis," *ACM Transactions on Information Systems*, vol. 36, no. 2, pp. 1–36, 2017.

[9] H. Peng, L. Xu, L. Bing, F. Huang, W. Lu, and L. Si, "Knowing What, How and Why: A Near Complete Solution for Aspect-based Sentiment Analysis," in *Proceedings of the AAAI 2019*, 2019, pp. 8600–8607.

[10] W. Xue and T. Li, "Aspect based sentiment analysis with gated convolutional networks," in *Proceedings of the ACL 2018*, 2018, pp. 2514–2523.

[11] Z. Li, Y. Wei, Y. Zhang, X. Zhang, and X. Li, "Exploiting coarse-to-fine task transfer for aspect-level sentiment classification," in *Proceedings of the AAAI 2019*, 2019, pp. 4253–4260.

[12] X. Li, L. Bing, P. Li, and W. Lam, "A unified model for opinion target extraction and target sentiment prediction," in *Proceedings of the AAAI 2019*, 2019, pp. 6714–6721.

[13] M. Hu, Y. Peng, Z. Huang, D. Li, and Y. Lv, "Open-domain targeted sentiment analysis via span-based extraction and classification," in *Proceedings of the ACL 2020*, 2020, pp. 537–546.

[14] H. Li and W. Lu, "Learning explicit and implicit structures for targeted sentiment analysis," in *Proceedings of the EMNLP 2020*, 2020, pp. 5478–5488.

[15] N. Majumder, R. Bhardwaj, S. Poria, A. Gelbukh, and A. Hussain, "Improving aspect-level sentiment analysis with aspect extraction," *Neural Computing and Applications*, vol. 34, pp. 8333–8343, 2022.

[16] W. Wang, S. J. Pan, D. Dahlmeier, and X. Xiao, "Coupled multi-layer attentions for co-extraction of aspect and opinion terms," in *Proceedings of the AAAI 2017*, 2017, pp. 3316–3322.

[17] H. Dai and Y. Song, "Neural aspect and opinion term extraction with mined rules as weak supervision," in *Proceedings of the ACL 2020*, 2020, pp. 5268–5277.

[18] C. Zhang, Q. Li, D. Song, and B. Wang, "A multi-task learning framework for opinion triplet extraction," in *Findings of the EMNLP 2020*, 2020, pp. 819–828.

[19] S. Chen, Y. Wang, J. Liu, and Y. Wang, "Bidirectional machine reading comprehension for aspect sentiment triplet extraction," in *Proceedings of the AAAI 21*, 2021, pp. 12 666–12 674.

[20] L. Xu, Y. K. Chia, and L. Bing, "Learning span-level interactions for aspect sentiment triplet extraction," in *Proceedings of the ACL 2021*, 2021, pp. 4755–4766.

[21] D. Dai, T. Chen, S. Xia, G. Wang, and Z. Chen, "Double embedding and bidirectional sentiment dependence detector for aspect sentiment triplet extraction," *Knowledge-Based Systems*, vol. 253, pp. 109 506–109 514, 2022.

[22] J. Zhang, Zhihao and Zuo, Yuan and Wu, "Aspect Sentiment Triplet Extraction: A Seq2Seq Approach With Span Copy Enhanced Dual Decoder," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 30, pp. 2729–2742, 2022.

[23] Z. Wu, C. Ying, F. Zhao, Z. Fan, X. Dai, and R. Xia, "Grid tagging scheme for aspect-oriented fine-grained opinion extraction," in *Findings of the EMNLP 2020*, 2020, pp. 2576–2585.

[24] L. Xu, H. Li, W. Lu, and L. Bing, "Position-aware tagging for aspect sentiment triplet extraction," in *Proceedings of the EMNLP 2020*, 2020, pp. 2339–2349.

[25] Z. Chen, H. Huang, B. Liu, X. Shi, and H. Jin, "Semantic and syntactic enhanced aspect sentiment triplet extraction," in *Findings of the ACL 2021*, 2021, pp. 1474–1483.

[26] Z. Zhao, Y. Liu, H. Wu, Z. Yue, and J. Li, "Multi-task Alignment Scheme for Span-level Aspect Sentiment Triplet Extraction," in *Proceedings of the ICANN 2022*, 2022, pp. 282–293.

[27] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "An unsupervised neural attention model for aspect extraction," in *Proceedings of the ACL 2017*, 2017, pp. 388–397.

[28] X. Li, L. Bing, P. Li, W. Lam, and Z. Yang, "Aspect term extraction with history attention and selective transformation," in *Proceedings of the IJCAI 2018*, 2018, pp. 4194–4200.

[29] A. Kumar, A. S. Veerubhotla, V. T. Narapareddy, V. Aruru, L. B. M. Neti, and A. Malapati, "Aspect term extraction for opinion mining using aa Hierarchical Self-Attention Network," *Neurocomputing*, vol. 465, pp. 195–204, 2021.

[30] D. Ma, S. Li, F. Wu, X. Xie, and H. Wang, "Exploring sequence-to-sequence learning in aspect term extraction," in *Proceedings of the ACL 2020*, 2020, pp. 3538–3547.

[31] Z. Wu, F. Zhao, X. Y. Dai, S. Huang, and J. Chen, "Latent opinions transfer network for target-oriented opinion words extraction," in *Proceedings of the AAAI 2020*, 2020, pp. 9298–9305.
[32] Z. Fan, Z. Wu, X. Y. Dai, S. Huang, and J. Chen, "Target-oriented opinion words extraction with target-fused neural sequence labeling," in *Proceedings of the NAACL 2019*, 2019, pp. 2509–2518.
[33] Y. Zhang, R. Jin, and Z. H. Zhou, "Understanding bag-of-words model: A statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, pp. 43–52, 2010.
[34] J. Wagner, P. Arora, S. Cortes, U. Barman, D. Bogdanova, J. Foster, and L. Tounsi, "DCU: Aspect-based polarity classification for semeval task 4," in *Proceedings of the SemEval 2014*, 2014, pp. 223–229.
[35] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the ACL 2011*, 2011, pp. 151–160.
[36] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective lstms for target-dependent sentiment classification," in *Proceedings of the COLING 2016*, 2016, pp. 3298–3307.
[37] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *Proceedings of the EMNLP 2016*, 2016, pp. 606–615.
[38] P. Chen, Z. Sun, L. Bing, and W. Yang, "Recurrent attention network on memory for aspect sentiment analysis," in *Proceedings of the EMNLP 2017*, 2017, pp. 452–461.
[39] B. Liang, R. Yin, L. Gui, J. Du, Y. He, and R. Xu, "Aspect-invariant sentiment features learning: Adversarial multi-task learning for aspect-based sentiment analysis," in *Proceedings of the CIKM 2020*, 2020, pp. 825–834.
[40] A. Pouran Ben Veyseh, N. Nouri, F. Dernoncourt, Q. H. Tran, D. Dou, and T. H. Nguyen, "Improving aspect-based sentiment analysis with gated graph convolutional networks and syntax-based regulation," in *Findings of the EMNLP 2020*, 2020, pp. 4543–4548.
[41] Y. Tian, G. Chen, and Y. Song, "Aspect-based sentiment analysis with type-aware graph convolutional networks and layer ensemble," in *Proceedings of the NAACL 2021*, 2021, pp. 2910–2922.
[42] X. Hou, J. Huang, G. Wang, P. Qi, X. He, and B. Zhou, "Selective attention based graph convolutional networks for aspect-level sentiment classification," in *Proceedings of the Workshop on TextGraphs-15*, 2021, pp. 83–93.
[43] B. Liang, H. Su, L. Gui, E. Cambria, and R. Xu, "Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks," *Knowledge-Based Systems*, vol. 235, pp. 107 643–107 653, 2022.
[44] Y. Xiao and G. Zhou, "Syntactic edge-enhanced graph convolutional networks for aspect-level sentiment classification with interactive attention," *IEEE Access*, vol. 8, pp. 157 068–157 080, 2020.
[45] C. D. M. Jeffrey Pennington, Richard Socher, "GloVe: Global vectors for word representation," in *Proceedings of the EMNLP 2014*, 2014, pp. 1532–1543.
[46] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[47] B. Zeng, H. Yang, R. Xu, W. Zhou, and X. Han, "LCF: A Local Context Focus Mechanism for Aspect-Based Sentiment Classification," *Applied Sciences*, vol. 9, no. 16, pp. 3389–3410, 2019.
[48] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: aspect based sentiment analysis," in *Proceedings of the SemEval 2014*, 2014, pp. 27–35.
[49] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos, "SemEval-2015 task 12: aspect based sentiment analysis," in *Proceedings of the SemEval 2015*, 2015, pp. 486–495.
[50] M. Pontiki, S. Galanis, Dimitrios Papageorgiou, Haris Androutsopoulos, Ion Manandhar, M. Al-Smadi, M. Al-Ayyoub, Y. Zhao, and B. Qin, "SemEval-2016 task 5: aspect based sentiment analysis," in *Proceedings of the SemEval 2016*, 2016, pp. 342–349.
[51] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the ICLR 2015*, 2015, pp. 1–15.
[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you Need," in *Proceedings of the NIPS 2017*, 2017, pp. 5998–6008.
[53] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," in *Proceedings of the ICLR 2017*, 2017, pp. 1–8.
[54] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[55] M. M. Amin, E. Cambria, B. W. Schuller, and E. Cambria, "Will affective computing emerge from foundation models and general artificial intelligence? a first evaluation of chatgpt," *IEEE Intelligent Systems*, vol. 38, no. 2, p. 15–23, 2023.
[56] R. Mao, Q. Liu, K. He, W. Li, and E. Cambria, "The biases of pre-trained language models: An empirical study on prompt-based sentiment analysis and emotion detection," *IEEE Transactions on Affective Computing*, pp. 1–11, 2022 (early access).
[57] E. Cambria, Q. Liu, S. Decherchi, F. Xing, and K. Kwok, "SenticNet 7: A commonsense-based neurosymbolic AI framework for explainable sentiment analysis," in *Proceedings of the IREC 2022*, 2022, pp. 3829–3839.

Li Yuan is currently pursuing his master's degree in the School of Information Science and Engineering, Yunnan University, China. He received a bachelor's degree in Information Management and Information Systems from Tianjin University of Technology, China. His research interests include natural language processing, text mining, and machine learning.

Jin Wang is an associate professor in the School of Information Science and Engineering, Yunnan University, China. He holds one Ph.D. in Computer Science and Engineering from Yuan Ze University, Taoyuan, Taiwan and one in Communication and Information Systems from Yunnan University, Kunming, China. His research interests include natural language processing, text mining, and machine learning.

Liang-Chih Yu is a professor in the Department of Information Management at Yuan Ze University in Taiwan, R.O.C. He received his Ph.D. in Computer Science and Information Engineering from National Cheng Kung University in Taiwan, R.O.C. He was a visiting scholar at the Natural Language Group, Information Sciences Institute, University of Southern California (USC/ISI) from 2007 to 2008, and at DOCOMO Innovations for three months in 2018. He is currently Board Member and Convener of SIGCALL of the Association for Computational Linguistics and Chinese Language Processing (ACLCLP), and serves as an editorial board member of *International Journal of Computational Linguistics and Chinese Language Processing*. His research interests include natural language processing, sentiment analysis, computer-assisted language learning. His team has developed systems that ranked first in *IJCNLP 2017 Task 4: Customer Feedback Analysis*, and second in the recent *SemEval* and *BEA* shared task competitions.

Xuejie Zhang is a professor in the School of Information Science and Engineering, and Director of High-Performance Computing Center, Yunnan University, China. He received his Ph.D. in Computer Science and Engineering from Chinese University of Hong Kong in 1998. His research interests include high performance computing, cloud computing, and big data analytics.