ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240963

Improving Aspect-Based Sentiment Analysis via Tuple-Order Learning

Gongzhen Hu[a], **Yuanjun Liu**[a], **Xiabing Zhou** [a,*] **and Min Zhang**[a]

[a]School of Computer Science and Technology, Soochow University, China
{gzhu, yjliu1}@stu.suda.edu.cn
{zhouxiabing, minzhang}@suda.edu.cn

Abstract. In the field of Natural Language Processing (NLP), Aspect-Based Sentiment Analysis (ABSA) has gained significant attention in recent years due to its ability to perform fine-grained sentiment analysis. Generative methods tackle various ABSA tasks by autoregressively generating the target sequence of sentiment tuples in a specified format. However, the sentiment tuple is intrinsically an unordered set, and the method introduces an order bias between the generated sequence and the original target. Therefore, to investigate the impact of sentiment tuples order on model performance, we conduct a pilot experiment, unveiling that the order of tuples significantly influences the learning outcomes of the Seq2Seq model. Thus, we propose a novel tuple-order learning method that prioritizes tuples from simple to complex, facilitated by a discrete evaluation method that assesses the difficulty of each individual tuple. Specifically, we incorporate positional information on tuples and employ an effective strategy to expedite the assessment of individual tuples. The method optimizes the learning process while maintaining the structural integrity of existing generative models. Extensive experiments show that our approach significantly advances the performance on 14 datasets of 5 benchmark tasks. We will release our code at https://github.com/gongzhenhu/TOL.

1 Introduction

Aspect-based Sentiment Analysis (ABSA) is the fine-grained Sentiment Analysis (SA) task, which aims to extract sentiment tuples for a given text of elements such as the aspect term (at), opinion term (ot), aspect category (ac), and sentiment polarity (sp) [30]. For example, in the sentence "the sushi was awful!", the corresponding elements are "sushi", "food quality", "awful", and "negative". Early studies primarily focus on the identification of single sentiment element such as Aspect Term Extraction (ATE)[28, 10], Opinion Term Extraction (OTE)[13], Aspect Category Detection (ACD) [32] or Aspect Sentiment Polarity Classification (ASC) [9, 20]. Due to the interconnectivity among sentiment elements, recent works propose compound ABSA tasks involving multiple associated elements with several representative tasks such as aspect opinion pair extraction (AOPE) [5], aspect sentiment triplet extraction (ASTE) [15], target aspect sentiment detection (TASD) [22], Aspect Sentiment Quad Extraction (ASQP and ACOS) [3, 23]. Their output formats are shown in Table 1.

* Corresponding Author

Table 1. The Aspect sentiment tuple prediction tasks with their corresponding output formats.

Task type	Task name	output
Quad	Aspect Category Opinion Sentiment (ACOS)	(a, c, o, s)
	Aspect Sentiment Quad Extraction (ASQP)	(a, c, o, s)
Triplet	Aspect Sentiment Triplet Extraction (ASTE)	(a,o,s)
	Target Aspect Sentiment Detection (TASD)	(a,c,s)
Pair	Aspect Opinion Pair Extraction (AOPE)	(a,o)

Recent developments in the field of ABSA have shifted towards creating a unified framework capable of handling various ABSA tasks concurrently, moving away from the traditional approach of employing distinct models for each task. Thus, the Sequence-to-Sequence(Seq2Seq) models have been applied to the ABSA tasks by formulating them as a text-to-text problem [29], where the output is a sequence of sentiment tuples. To be specific, they use class index [26], sentiment element sequence [29], sentence annotation [29], natural language[23], structured extraction schema [11] or opinion tree [1] to construct sequence of sentiment tuples as the target of the generation models.

However, the ABSA task is not inherently a typical generative task. While this paradigm shows promise, it encounters a challenge due to the unidirectional nature of the decoder in the generative pre-trained language model [19]. This unidirectional decoder generates the target sequence starting from the beginning and proceeding to its end. This results in the formation of sequential relationships among the constructed sentiment tuples, which may potentially affect the learning effectiveness of the model. Consider a sentence that includes n sentiment tuples, defined as $\{tuple_1, tuple_2, \ldots, tuple_n\}$, where each tuple consists of the output element as determined by the ABSA task. The process of constructing the targeted sequence of sentiment tuples has $n!$ different permutation possibilities. These varying orders of arrangement may potentially impact the learning effectiveness of the models.

Based on the above, the following issue has arisen: Does the order of the tuples impact the performance of generative pre-trained language models? Thus, we conduct a pilot experiment to investigate the impact of different arrangement orders on the various ABSA tasks. From these experiments, we observed that: (1) The order of tuples significantly influences the learning effectiveness of the Seq2Seq model; (2) In most cases, ordering the tuples according to their natural occurrence within the sentence benefits the learning of the model.

Motivated by the observations and analyses, our objective is to identify an optimized ordering that enhances the learning effectiveness of existing generative paradigms without modifying the structures of the model. Therefore, we propose a tuple-order learning approach for ABSA. Inspired by curriculum learning approaches [2], our method adopts the simple-prioritized principle to adaptively learn the tuple order for each sample, utilizing the generation of the simple tuples to facilitate the generating of more challenging tuples. Specifically, we first convert every sentiment tuple into a sequence with special markers and train a selector based on the format of a single sentence as input and its corresponding single tuple sequence as output, which enables it to initially assess the difficulty level of each tuple. Then, we utilize the selector to categorize the tuple as either simple or difficult. Additionally, a simplified approach is proposed to reduce the time cost of the selector evaluation process. At last, we incorporate positional information and perplexity to sort the tuples and fine-tune the model with the sorted tuples.

The experimental results demonstrate that our method achieves effective performance across multiple datasets and various ABSA tasks. Our contribution can be summarized as follows:

- We study the influence of tuple order in the various ABSA tasks, showing the order of tuples significantly influences the learning effectiveness of the Seq2Seq model. To the best of our knowledge, this work is the first attempt to investigate tuple order for the various ABSA tasks.
- We propose a tuple-order learning method based on a simple-prioritized principle. The method evaluates the difficulty of each independent tuple by training a selector and integrates positional information and prediction accuracy to determine a comprehensive score. Additionally, a simplified approach is proposed to reduce the time cost of the selector evaluation process.
- Extensive experimental validation has demonstrated that our method effectively enhances the learning effectiveness of the model, resulting in favorable outcomes.

2 Related Work

Aspect-based sentiment analysis (ABSA) is the problem of identifying sentiment elements of interest for a concerned text, either a single sentiment element or multiple elements with the dependency relation between them [30]. Early studies focus on the prediction of a single element, such as extracting the aspect term[10], detecting the mentioned aspect category [32], and predicting the sentiment polarity[20] for a given aspect. The four elements do not exist independently, and they have strong connections with each other. Therefore, the focus shifted toward the simultaneous extraction of multiple sentiment elements. Thus, pair extraction, triplet extraction, and quadruple extraction tasks are proposed in the ABSA field.[3, 5, 15, 22, 23].

In the field of ABSA, two main approaches have been primarily explored. Discriminative approaches rely on encoder-based architectures. For instance, GTS employs a tagging scheme to address the AOPE task, subsequently extending it to address the ASTE task [24]. Building on GTS, EMC-GCN improves ASTE task performance by incorporating additional information such as dependency and part-of-speech into the model [4]. Meanwhile, Mirror restructures Information Extraction (IE) problems into a unified framework of multi-slot tuples, thereby facilitating diverse IE tasks [33]. Conversely, generative methods tackle various ABSA tasks by transforming labels into sequences[23, 29], which avoids the complex modeling and inference processes typical of traditional discriminative methods while

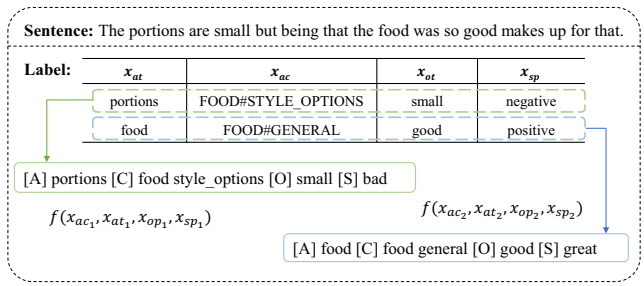

Figure 1. An example of mapping sentiment tuples with the function f

demonstrating excellent results. Specifically, Zhang et al.[29] utilized the T5 model as a generative framework, proposing Extraction-style and Annotation-style paradigms. This framework is capable of unifying various ABSA tasks. Yan et al.[27] achieved a unified approach using the BART model, with the positional indices of sentiment elements in the sequence as the output target, and fused information from both the encoder and decoder for sentiment classification. The design of the output targets has a significant impact on the generative results. To better align these targets with the pre-trained knowledge embedded in the language models, Zhang et al. [23] modeled the outputs as natural language sequences.

However, the generative paradigm exhibits order biases between the generated sequences and the original targets. To address these issues, research has focused on the order of four sentiment elements, referred to as the template order. Hu et al. [8] proposed selecting the sequence from these 24 permutations that yield the minimum entropy at the T5 decoder as the training target. Gou et al. [7] learn information from various template orders and aggregate predictions across different templates during inference. Beyond template order, researchers have attempted to generate all sentiment tuples concurrently. Mao et al. [12] utilized beam search to simultaneously generate all triplets for input in sentiment triplet extraction tasks. Complementarily, Fei et al. [6] proposed a Nonautoregressive Encoder-Decoder Neural Framework, which generates all corresponding sentiment elements in parallel through a parallel decoder design. These methods eliminate the sequential relationships between tuples by modifying the decoder, but they also reduce model universality and make it difficult to combine with existing Pre-trained Language Models(PLMs).

Different from the previous research, we explore the impact of different tuple orders on the model's performance, aiming to identify an optimized ordering that enhances the learning effectiveness of generative paradigms without modifying the decoder and integrates seamlessly with existing PLMs.

3 Preliminary and Pilot Experiment

3.1 Problem Formulation

In this section, we focus on the quadruple task and the other tasks can be regarded as special cases of it. Given a sentence $x = \{x_1, x_2, ..., x_N\}$ with N words, we aim to identify all sentiment tuples $\mathcal{G} = \{(a_i, c_i, o_i, s_i) \mid i = 1, 2, \ldots, |\mathcal{G}|\}$, where a_i, c_i, o_i, s_i represent aspect term (at), aspect category (ac), opinion term (ot), sentiment polarity (sp), respectively. The aspect term (at) and the opinion term (ot) are typically text spans in the sentence x while they can also be null if they are not explicitly mentioned: $a \in V_x \cup \{\emptyset\}$, where V_x denotes the set containing all possible continuous spans of x. The aspect category (ac) falls into a category set

V_c. The sentiment polarity (sp) belongs to one of the sentiment class $\{positive, neutral, negative\}$ denoting the positive, neutral, and negative sentiment, respectively.

3.2 Seq2Seq Learning

The encoder-decoder model is utilized to rephrase the source sentence x into the target sequence y. Therefore, we need to convert the sentiment tuple $\mathcal{G}$ into the target sequence y. To leverage pre-trained knowledge and the semantic information of labels, we follow previous research [23], mapping implicit aspect term (at) and implicit opinion term (ot) to "it" and "null", respectively; the sentiment polarity $\{positive, neutral, negative\}$ are mapped to $\{great, ok, bad\}$. To indicate different sentiment elements, we follow the previous method [8, 7], and design special markers to represent the structure of the information [14]. The markers for $x_{at}, x_{ac}, x_{ot}, x_{sp}$ are $[A]$, $[C]$, $[O]$, $[S]$, respectively. We add the corresponding marker as a prefix to each element and concatenate them according to a specified permutation. The above process is denoted as a function f and is specifically illustrated in Figure 1. When dealing with multiple sentiment tuples for an input sentence, we utilize a special symbol $[SSEP]$ to concatenate each tuple, thereby forming the final target sequence y. Thus we obtain an input-output pair for Seq2Seq training.

The Seq2Seq model further decomposes $p(y|x)$ autoregressively using the chain rule as follows:

$$\begin{aligned} p_\theta(y|x) &= p_\theta(y_1, y_2, \ldots, y_T|x) \\ &= p_\theta(y_1|x) \prod_{t=2}^{T} p_\theta(y_t|y_1, \ldots, y_{t-1}, x), \end{aligned} \tag{1}$$

where θ denotes the parameters of the Seq2Seq model. During training, a pre-trained encoder-decoder model, i.e. T5 [19], is chosen to initialize the parameter θ and fine-tuned with minimizing the cross-entropy loss:

$$\mathcal{L}(x, y) = -\sum_{t=1}^{T} \log p_\theta(y_t|x, y_{<t}) \tag{2}$$

where T is the length of the target sequence y.

3.3 Pilot Study

Due to $\mathcal{G}$ being a set, where tuples are unordered, while sequence y contains an order, our optimization target $p(y|x)$ is an approximation of the original target $p(\mathcal{G}|x)$ in Section 3.2. This introduces an inherent order bias. It raises the question of whether the order of tuples within $\mathcal{G}$ affects the effectiveness of the Seq2Seq model. To explore this, we conducted a pilot experiment focusing on various ABSA tasks. In this experiment, we predefined the order of elements within each tuple to follow "$a \rightarrow o \rightarrow c \rightarrow s$". The results of the experiment are displayed in Table 2.

In this experiment, we employ various ordering strategies to organize tuples and construct target sequences based on these to study the impact of different target sequences.

. The Default Order [8, 23, 29] refers to the construction of target sequences based on the default ordering of tuples within the dataset. The Random Order refers to randomly selecting one valid sequence as the ground truth sequence for a sentence. The MVP Order [7] means we sort the tuples of a sentence in $max(Ind(x_{at}, x), Ind(x_{ot}, x))$ and build ground truth sequence based on the sorted tuples, where $Ind(x_{at}, x)$ represents the index

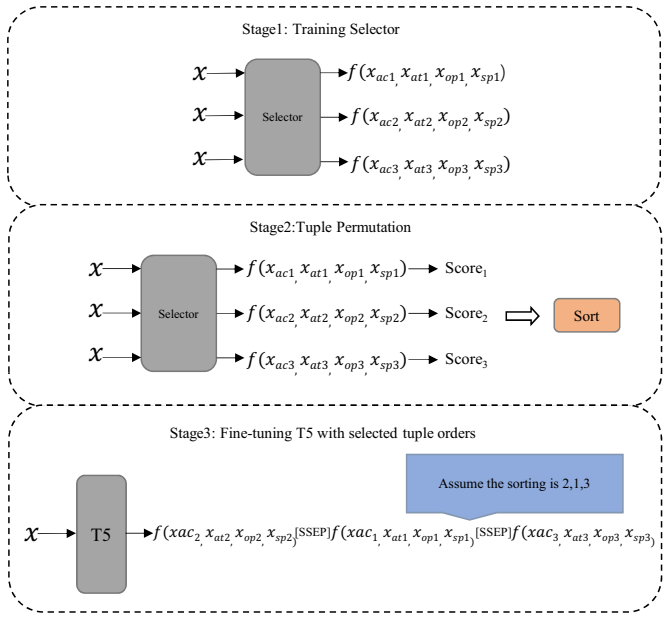

Figure 2. The proposed method is composed of three stages. The objective of the first stage is to train a selector for scoring tuples in the subsequent stage. The second stage utilizes the trained selector to score tuples and sort them based on their scores. The third stage constructs training samples with the sorted tuples and fine-tunes the T5 model.

of aspect term x_{at} in sentence x and $Ind(x_{ot}, x)$ represents the index of opinion term x_{at} in sentence x. The SLGM Order [31] indicates sorting the tuples by $Ind(x_{at}, x)$ and, if they are the same, further sorting them by $Ind(x_{ot}, x)$. The SLGM Order Reversed means building the ground truth sequence in an order that is the reverse of the SLGM Order. Based on the results of the pilot experiment, we have the following findings.

Tuple order influences the learning performance of Seq2Seq Model. In Table 2, the F1 score is from 62.19% to 65.3% for the Rest15 dataset of the ASTE task. Similarly, the F1 score is from 42.97% to 44.1% for the Laptop dataset of the ACOS task. Based on the results of Table 2, we can conclude that the order of tuples impacts the learning of the model. Furthermore, the performance of a particular order varies across different ABSA tasks, indicating that it is challenging to identify an ordering strategy that consistently outperforms all ABSA tasks.

Sorting tuples based on their sequential occurrence within the sentence tends to be an effective sorting strategy.. Further analyzing the results presented in Table 2, we observe that the performance obtained by the SLGM Order, MVP Order, and Default Order frequently ranked at the top compared with other sorting methods in this experiment. Specifically, the SLGM Order and MVP Order first evaluate the position of the tuples within the sentence and then sort the tuples based on those evaluations. In the case of the Default Order, we find that it is mostly similar to the SLGM Order, but there are still some data where tuples that appear at the beginning of the sentence are placed at the end after sorting. To further validate our findings, We further designed an experiment named the SLGM Order Reversed, which involves reversing the ordering results of the SLGM Order. The outcomes of this experiment are summarized in Table 2, demonstrating unsatisfactory performance across various ABSA tasks. Therefore, we can conclude that generating sequences constructed based on the order in which tuples appear in the sentence is an effective approach for the Seq2Seq model.

Table 2. Preliminary experimental results on various tasks in ABSA. All the reported F1($F1, \%$) scores are the average of five runs.

Ordering Method	ACOS		ASQP		ASTE				TASD		AOPE			
	Lap	Rest	R15	R16	L14	R14	R15	R16	R15	R16	L14	R14	R15	R16
Default Order	44.1	59.82	48.37	**59.35**	62.47	72.25	**65.3**	72.39	**63.34**	70.44	70.18	73.87	**69.25**	77.13
Random Order	42.74	58.8	47.81	57.35	61.59	71.71	62.63	70.48	62.39	69.27	70.12	73.67	67.32	75.45
SLGM Order	43.67	**59.96**	48.71	59.22	**63.54**	**73.52**	65.24	72.04	62.5	**71.15**	71.75	74.05	68.78	**77.45**
MVP Order	**43.94**	59.41	**48.9**	59.26	63.21	73.25	65.43	**72.65**	62.5	**71.15**	**71.89**	73.92	68.57	77.34
SLGM Order Reversed	42.97	58.43	46.7	57.75	61.76	71.96	62.19	70.76	61.35	69.61	70.67	**74.98**	66.72	75.47

In summary, these pilot experiments indicate that (1) The tuple order influences the effectiveness of Seq2Seq, which means that there may be an optimal order for model learning, and (2) In general, Constructing target sequences based on the natural order of tuples within sentences enhances the effectiveness of Seq2Seq model. Therefore, it is crucial to develop a learnable target sequence for optimal performance.

4 Methodology

In the previous section, we found the order of tuples affects the learning effectiveness of the Seq2Seq model in various ABSA tasks. Therefore, we aim to identify an optimized ordering that enhances the learning effectiveness of generative paradigms. Inspired by curriculum learning [2], which starts from easy instances and then gradually handles harder ones, we consider organizing tuples in an order based on the simple-prioritized principle. In this pattern, generating information about simple tuples can help and facilitate the generation of more challenging tuples. In addition, as analyzed in the pilot experiment, the order in which context appears also facilitates model generation. Therefore, we adopted a discrete evaluation method to determine the difficulty of tuples and integrate it with positional information and perplexity. Figure 2 depicts the framework of our approach, which will be introduced in detail next.

4.1 Selector Training

To evaluate the difficulty of tuples in various ABSA tasks, we estimate how accurately the model is trained for each tuple. We first reorganize the training data and treat each tuple as an independent learning objective. Give an input sentence x, the corresponding set of labels is denoted by $\mathcal{G} = \{g^1, g^2, \ldots, g^{|\mathcal{G}|}\}$, where each label $g^i = (a_i, c_i, o_i, s_i)$ represents a tuple. We transform the original training dataset into a new set of training samples $\{(x, f(g^1)), (x, f(g^2)), \ldots, (x, f(g^{|\mathcal{G}|}))\}$. Then, to enhance the capability of the model to generate single tuples, we train the T5 model by optimizing the minimizing the Eq. 3:

$$\mathcal{L}(x, f(g^i)) = -\sum_{i=1}^{|G|} \frac{1}{L^i} \log p(f(g^i)|x), \tag{3}$$

where L^i denotes the number of tokens for each $f(g^i)$.

Based on this training mode, the model can gradually show the ability to generate individual tuples, and the tuple with better generated is easier for the model to learn, here we call it a simple tuple, and the reverse is a difficult tuple. This model derived from this process is referred to as the selector and used in the subsequent stage, as shown in Figure 2.

4.2 Tuple Permutation

The core of our methodology is employing a trained selector to assess the difficulty of individual tuple. It is challenging to evaluate how difficult to generate a tuple sequence. The general measurement, like perplexity, is to look at the overall coarse-grained perspective. In our method, the difficulty of tuple requires fine-grained consideration of the accuracy of generating all tokens inside, even if a single token generation error will cause the entire tuple to be generated incorrectly. In addition, it is necessary to avoid simple illusions due to the frequency of certain patterns within the training dataset. Therefore, we propose a tuple permutation approach by fully considering the accuracy of model generation, which is the discrete indicator to assess the difficulty level of tuples. We show the details as follows.

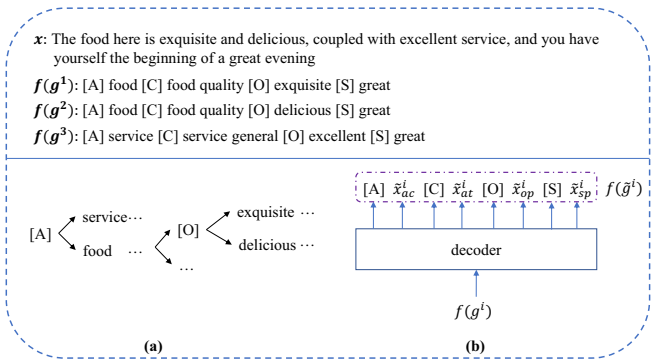

Figure 3. An example for demonstrating the schematic diagram of tuple generation. (a) The tuple generated by using beam search; (b) The generated tuple of the simplified strategy.

4.2.1 Preliminary Evaluation of Tuple Difficulty

Based on the selector training stage, we can intuitively conclude the accuracy of single tuple predictions. For the sample where the prediction is completely correct, it is easily learned by the model. However, it is worth noting that in the data used to train the model, there are different tuple outputs under the same input.Thus on the inference stage, we should use the beam search strategy instead of the greedy search strategy. Figure 3 provides a schematic diagram, the input x corresponds to three different tuples. When generating results based on the Seq2Seq model, the token following $[A]$ should most likely be "food" or "service" (indicating that these two tokens have the highest selection probabilities). To ensure that both words can be found, it is necessary to increase the beam width during beam search, As shown in Figure 3(a).

Therefore, for each input x, the set of sequences $\{f(g^1(x)), f(g^2(x)), \ldots, f(g^{|\mathcal{G}|}(x))\}$ is the $|\mathcal{G}|$ sequences that the model considers to be the most probable. By setting the beam search width to $|\mathcal{G}|$, we can extract from the model the $|\mathcal{G}|$ most likely generated outcomes for a given input x. If the sequences in $\{(x, f(g^1)), (x, f(g^2)), \ldots, (x, f(g^{|\mathcal{G}|}))\}$ are successfully generated by the selector, we naturally define them as "simple tuples", indicating that these tuples are ones the model can predict accurately. Conversely, sequences that the model fails to generate are defined as

"difficult tuples".

Process simplification In beam search, as the beam width is increased, the number of candidate sequences considered at each step grows, leading to a corresponding increase in the computational requirements and time taken for beam search. Therefore, we have approximated it. Specifically, as depicted in Figure 3(b), we fed the sequence $f(g^i)$ into the decoder of the Selector, while the sentence x is input into the encoder. Subsequently, the decoder generates the sequence $f(\tilde{g}^i)$ in parallel. We then calculate the number of token mismatches between $f(g^i)$ and $f(\tilde{g}^i)$, denoted as $error$. Tuples with $error$ value less than or equal to 1 are defined as "simple tuples", whereas those with $error$ value greater than 1 are defined as "difficult tuples".

4.2.2 Tuple Scheduler

After obtaining the preliminary evaluation of tuples, the next step is to determine the distribution of the output target and refer to the order of the tuples, which is the tuple scheduler. The tuple scheduler takes into account both position information and perplexity measurement and provides the tuple permutation.

Based on the preliminary evaluation, the tuples of each sample x are divided into simple tuples G_s and difficult tuples G_d. As analyzed in the previous pilot analysis, for the model, the easier it is to generate a tuple that is more consistent with the position of the input text. Here, we assess the position of the tuples based on the location of the sentiment elements (aspect terms and opinion terms) within the sentence. let $w_s = \{w_s^1, \cdots, w_s^{n_s}\}$ represents the score of tuples in G_s, we set:

$$w_s^j = \frac{1}{lp_a^j + p_o^j/l}, j \in \{1, 2, \cdots, n_s\}, \tag{4}$$

where n_s is the number of tuples in G_s, j is the j-th tuple, l is the length of x, p_a^j is the position of aspect in x for the j-th tuple and p_o^j is the position of opinion in x for the j-th tuple. For the difficult tuples G_d, we calculate the perplexity of each one and obtain the $w_d = \{w_d^1, \cdots, w_d^{n_d}\}$:

$$w_d^j = PPL(f(g_d^j)|x; \theta), j \in \{1, 2, \cdots, n_d\}, \tag{5}$$

where n_d is the number of tuples in G_d, g_d^j is the j-th tuple in G_d, and w_d^j represents the score of g_d^j. Finally, we obtain the simple tuple order $G_{ordered}^s$ and difficult tuple order $G_{ordered}^d$ in terms of w_s and w_d. The final tuple permutation is denoted as $\mathcal{G}_{ordered} = G_{ordered}^s \oplus G_{ordered}^d$.

4.3 Fine-tuning with Tuple Orders

After obtaining the ordered tuple sequence $\mathcal{G}_{ordered}$, our objective is to enable the model to leverage the knowledge gained from generating simple tuples to then attempt the generation of difficult tuples. Therefore, we concatenate each tuple in $\mathcal{G}_{ordered}$ after applying the function f using the special token $[SSEP]$ to form the sequence $\tilde{y}$, and continue to train the T5 model on $(x, \tilde{y})$ by the minimizing Eq. 6:

$$\mathcal{L}(x, \tilde{y}) = -\sum_{i=1}^{|N|} \log p_\theta(\tilde{y}_i|x, \tilde{y}_{<i}), \tag{6}$$

where N is the length of $\tilde{y}$.

5 Experiments

5.1 Tasks and Dataset

We conduct our methods on 14 datasets over 5 tasks, including quadruplet tasks, ASQP and ACOS, triplet tasks, ASTE and TASD, and a pair task, AOPE. For a fair comparison, we apply the same data splits as previous works.

For the ASQP task, we adopt two datasets in the restaurant domain based on SemEval tasks [16, 17, 18], Rest15 (R15) and Rest16 (R16) aligned and completed by zhang et al.[23] subsequently. For the ACOS task, we apply Restaurant-ACOS (Rest) and Laptop-ACOS (Lap) constructed by Cai et al. [3]. Although the ACOS and ASQP tasks share the same output format, the dataset used for the ACOS task places a greater emphasis on the analysis of implicit aspects and opinions compared to that of the ASQP task, which helps to measure our methods comprehensively. For the ASTE tasks, we adopt the datasets provided by Xu et al.[25], which is the revised variant of Peng et al. [15], the revised dataset addresses missing triplets with overlapping opinions. We adopt the dataset provided by Wan et al. [22] and Fan et al. [5] for TASD and AOPE tasks, respectively.

5.2 Implement Details

We employ the T5-Base model [19] from Huggingface Transformers library[1] as the pre-trained model of stage 3. To maintain consistency with the model used during the fine-tuning stage, our Selector is also trained using the same pre-trained model. The structure of the T5 encoder and decoder is similar to that of the Transformer [21].

For all ABSA tasks, we use a fixed batch size of 16 and a fixed learning rate $1e^{-4}$ to train the selector and model of Stage 3 with a single Nvidia A100 GPU. In the training process of the selector, we have kept the learning rate and batch size settings the same as previously established to maintain consistency with the fine-tuning process in the subsequent third phase to better understand tuple difficulty. Regarding the number of training epochs for the selector, we have uniformly set it to 5 rounds across all datasets for all tasks. For the training of the model in Stage 3, we followed the methodology of previous studies[7, 8, 29] by conducting training over 20 epochs and choosing the model from the final epoch for testing. During the inference, we utilize the greedy search decoding to generate the output sequence.

5.3 Evaluation Metrics

For all ABSA tasks, a predicted sentiment tuple is considered as correct if and only if all its elements are exactly the same as the gold tuple. We use F1 scores as the main evaluation metrics [3, 7, 23]. All reported F1 scores are averaged over 5 runs with different random seeds.

5.4 Compared method

We compare our methods with the following two types of previous strong baseline methods:

Discriminative methods generally utilize BERT as the language encoder and predict sentiment tuples. **TAS-BERT** [22], based on extraction jointly detects the sentiment tuples. **GTS** [24] introduces a grid tagging scheme for the AOPE task and extends it to the ASTE

[1] https://github.com/huggingface/transformers

Table 3. Main results on 14 datasets of ASQP, ACOS, TASD , ASTE and AOPE tasks. F1 scores are reported; the best results are in bold, while the second best are underlined. All the reported F1($F1, \%$) scores are the average of five runs.

Baseline Method	ACOS		ASQP		ASTE				TASD		AOPE			
	Lap	Rest	R15	R16	L14	R14	R15	R16	R15	R16	L14	R14	R15	R16
TAS-BERT	27.31	33.53	34.78	43.71	-	-	-	-	57.51	65.89	-	-	-	-
GTS	-	-	-	-	55.42	68.81	58.6	67.58	-	-	65.67	75.53	67.53	74.62
EMC-GCN	-	-	-	-	58.81	71.78	61.93	68.33	-	-	-	-	-	-
Extract-Classify	35.8	44.61	36.42	43.77	-	-	-	-	-	-	-	-	-	-
GAS	-	-	45.98	56.04	58.19	70.52	60.23	69.05	60.63	68.31	68.08	74.12	67.19	74.54
Paraphrase	43.51	59.18	46.93	57.93	60.55	70.89	62.82	71.70	63.06	70.87	-	-	-	-
BARTABSA	-	-	-	-	57.59	72.46	60.11	69.98	-	-	66.11	**77.68**	67.98	77.38
DLO	43.64	59.99	48.18	59.79	61.46	72.39	64.26	73.03	62.95	71.79	-	-	-	-
SVP(heuristic)	43.83	59.38	49.02	59.56	62.09	72.61	65.29	73.27	61.98	71.57	-	-	-	-
MVP	43.92	**61.54**	**51.04**	**60.39**	63.33	**74.05**	65.89	73.48	**64.53**	**72.76**	-	-	-	-
SLGM Order	43.67	59.96	48.71	59.22	63.54	73.52	65.24	72.04	62.5	71.15	71.75	74.05	68.78	77.45
MVP Order	43.94	59.41	48.9	59.26	63.21	73.25	65.43	72.65	62.5	71.15	71.89	73.92	68.57	77.34
Default Order	44.1	59.82	48.37	59.35	62.47	72.25	65.3	72.39	63.34	70.44	70.18	73.87	69.25	77.13
Our	**44.21**	60.65	50.34	60.12	**64.21**	73.92	**66.34**	**73.66**	63.54	71.87	**72.42**	74.46	**69.86**	**78.73**

task. **EMC-GCN** [4] propose an enhanced multi-channel graph convolutional network model to address the ASTE task by utilizing the relations between words. **Extract-Classify** [3] decomposes the quad extraction task into two steps.

Generative methods usually concatenate multiple tuples together into a sequence and then use the Seq2Seq model process. **GAS** [29] propose a model of various ABSA tasks as a generation process. **BARTABSA** redefines every ABSA subtask target as a sequence mixed by pointer indexes and sentiment class indexes, which converts all ABSA subtasks into a unified generative formulation. **Paraphrase** [23] designs semantic templates filled with fixed-order elements of tuples as generation targets. **DLO/ILO** [8] propose a simple but effective method to identify the most proper order, and further combine multiple proper templates as data augmentation to improve the ASQP task. **MVP** [7] introduces element order prompts to guide the language model to generate multiple sentiment tuples, each with a different template order, and then selects the most reasonable tuples by voting. When the process is simplified to utilize a single fixed heuristic template template, it transitions to **SVP (heuristic)**(Single-View Prompting) [7]. For comparison, we include three strategies from pilot experiments—**SLGM Order**, **MVP Order**, and **Default Order** into our baseline.

As a fair comparison, all results of these supervised methods are obtained from the base pre-trained model, either BERT, BART, or T5.

5.5 Experimental Results

5.5.1 Overall Results

Experimental results of various approaches are reported in Table 3. The best F1 score are marked in bold, and the second-best F1 score are marked in underlined. We observed that compared to baseline methods, our approach demonstrated robust performance. Although it did not surpass the MVP method on certain datasets, it achieved comprehensive performance improvements over the SVP method. Moreover, unlike the MVP method, which aggregates different template orders after fixing the tuple order, our method significantly outperforms the MVP scheme in terms of both training and inference speed, due we only consider one template. Furthermore, upon comparing our proposed ordering method with the three best-performing ordering methods from the pilot experiments—namely, SLGM Order, MVP Order, and Default Order—we observed comprehensive improvements. This further validates the rationality and effectiveness of our proposed approach. In the case of the R14 dataset for the AOPE task, the majority of generative methods exhibited relatively weaker performance, with the exception of BARTABSA. Our analysis suggests that it is difficult for the model to generate the token of this dataset on the decoder, while BARTABSA selects the desired token on the encoder by a pointer network.

Table 4. F1 scores of ablation study on ASTE dataset

Model	L14	R14	R15	R16
Our Approach	64.21	73.92	66.34	73.66
w/o discrete evaluation	62.84	71.71	62.61	71.02
w/o selector	63.54	73.52	65.24	72.04

5.5.2 Ablation Study

Taking the ASTE task as an example, we conducted ablation studies to analyze the impact of each strategy within our method. The results of ablation experiments are presented in Table4. The term "w/o discrete evaluation" indicates that we refrained from employing a discrete evaluation approach for assessing tuples. Instead, we chose to sort them based on their perplexity within the selector. "w/o selector" indicates that we did not utilize the selector for evaluating tuples, and instead, adopted the SLGM Order for constructing the target sequences. The performance significantly declined after omitting the discrete evaluation method. We ascribe this decline to two primary factors. Firstly, perplexity is an inadequate metric for assessing the complexity of tuples, as a lower perplexity value does not inherently signify that a tuple is straightforward or foreseeable. This limitation arises from the fact that perplexity evaluates the entire sequence, whereas the criteria for various ABSA tasks are stringent—even a single word error within the sequence can compromise the correctness of the tuple. Secondly, perplexity-based tuple evaluation neglects the positional information of tuples within sentences. Previous pilot experiments have demonstrated that sorting tuples based on their sentence positions can enhance model learning effectiveness. Therefore, using perplexity as an evaluation tool is not only impractical but also disregards the inherent sentence structure, elucidating why the "w/o discrete evaluation" approach yields inferior results compared to the "w/o Selector" method. Furthermore, we observed a slight decline in model performance when the selector was absent. This occurred even though the model utilized positional information for tuples within sentences. The absence of a difficulty

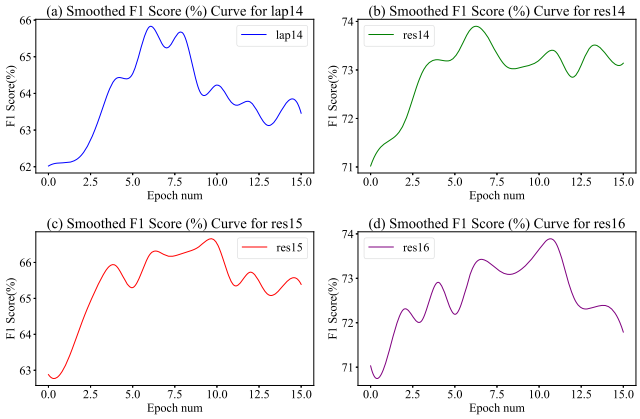

Figure 4. The impact of train epoch nums for Selector on the ASTE dataset (F1-score, %)

assessment mechanism led to the placement of complex and unpredictable tuples at the beginning of the sequence, thereby compromising overall performance.

5.5.3 *The impact of training epoch number for Selector*

In our approach, a critical hyperparameter for training the selector is the number of training epochs. To this end, we conducted experiments on the ASTE dataset to analyze the impact of the number of training epochs for the Selector. As illustrated in Figure 4, it is observed that the best number of training epochs varies across different ASTE datasets. Nonetheless, the F1 score curves for all four datasets exhibit a trend of initially increasing and then slightly decreasing. We ascribe this phenomenon to the constrained knowledge gained during the initial stages of selector training, leading the system to classify the majority of tuples as difficult and subsequently sort them based on perplexity. This method does not fully leverage the positional information of tuples in sentences, and relying solely on perplexity to evaluate tuple difficulty is unreasonable. In extreme cases, resembling the effect of random ordering in prior experiments. As the training epoch increases, overfitting results in most tuples being classified as easy, which in extreme cases approaches the ordering used in our prior experiment with the SLGM Order and the MVP Order. This experiment also validates the effectiveness of our proposed selector, demonstrating that after an appropriate number of training epochs, the selector can effectively evaluate tuples.

5.5.4 *Performance on Multi-Tuple data*

To assess the effectiveness of our model in handling Multi-Tuple data, we used the ASTE task as a case study, dividing the data in the test dataset into Single-Tuple and Multi-Tuple to further analyze the impact of our method on these two types of data. The experimental results are shown in Figure 5. These results indicate that our method significantly enhances the model's performance on Multi-Tuple data. Taking the exact match metric as an example, compared to the SLGM Order, our approach achieves improvements of 2.1%, 0.56%, 3.98%, and 2.20% on the L14, R4, R15, and R16 datasets, respectively, for Multi-Tuple data. Additionally, the performance on Single-Tuple data is similar to that achieved with the SLGM Order. Additionally, the findings indirectly validate that sorting tuples from simple to difficult is an effective approach for Multi-Tuple data, thereby confirming the rationality and effectiveness of our method.

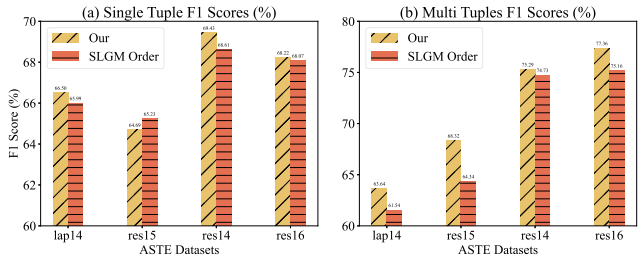

Figure 5. F1-score (%) for Single-Tuple Sentence and Multi-Tuple Sentence on ASTE Datasets compared with SLGM Order

5.6 *Time Complexity Analysis*

For a simple, we assume the length of the input sentence is denoted by l, and there are n tuples. After undergoing the transformation by function f in Figure 1, suppose the target sequence, constructed from n tuples, has a length $t = \sum_{i=1}^{n} t_i$, where t_i represents the length of the i tuple after transformation. Let m represent the number of training rounds in Stage 1, and g denote the training rounds in Stage 3 as illustrated in Figure 2. When the sequence is not too long, we can ignore the consumption of attention, as the time complexity of a Seq2Seq model is linearly related to the sequence length. For the selector training in Stage 1, its time complexity is approximately $O(nml+mt)$. In Stage 2, the time complexity of the discrete evaluation method has been optimized in Section 4.2.1, resembling speculative decoding, resulting in a time complexity of $O(nl+t)$. For the fine-tuning stage of Stage 3, its time complexity is approximately $O(gl+gt+g(n-1))$. Although the selector training and discrete evaluation introduce additional time, this overhead is mitigated by: (1) Only needing to be done once, with results saved for later training stages, (2) The number of training epochs for the selector is usually not large, making the time overhead acceptable compared to the improvements they bring.

6 Conclusion

In this work, we delve into the influence of tuple order on the performance of various ABSA tasks. Inspired by findings from pilot experiments and the proven effectiveness of the simple-prioritized principle in the domain of curriculum learning, we introduced an innovative tuple-order learning method. It is based on a discrete evaluation pattern to train a selector by reconstructing the training data and effectively evaluating the difficulty of each tuple. In addition, we also incorporate the positional information of tuples in sentences and the perplexity of tuple prediction to refine the tuple permutation, thereby enhancing the performance of various ABSA tasks. Specifically, considering the time efficiency of the model, we have designed an approximate evaluation strategy for assessiong the difficulty of single tuple by selector. By fine-tuning a pre-trained model with the target sequence constructed in the aforementioned tuple order, our model achieved significantly competitive performance.

Acknowledgements

We thank the anonymous reviewers for their valuable suggestions to improve the quality of this work. This work was partially supported by the National Natural Science Foundation of China 62176174 and Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] X. Bao, Z. Wang, X. Jiang, R. Xiao, and S. Li. Aspect-based sentiment analysis with opinion tree generation. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4044–4050, 2022.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 41–48, 2009.

[3] H. Cai, R. Xia, and J. Yu. Aspect-category-opinion-sentiment quadruple extraction with implicit aspects and opinions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 340–350, 2021.

[4] H. Chen, Z. Zhai, F. Feng, R. Li, and X. Wang. Enhanced multi-channel graph convolutional network for aspect sentiment triplet extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2974–2985, 2022.

[5] 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 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, 2019.

[6] H. Fei, Y. Ren, Y. Zhang, and D. Ji. Nonautoregressive encoder–decoder neural framework for end-to-end aspect-based sentiment triplet extraction. *IEEE Transactions on Neural Networks and Learning Systems*, 34 (9):5544–5556, 2023.

[7] Z. Gou, Q. Guo, and Y. Yang. MvP: Multi-view prompting improves aspect sentiment tuple prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4380–4397, 2023.

[8] M. Hu, Y. Wu, H. Gao, Y. Bai, and S. Zhao. Improving aspect sentiment quad prediction via template-order data augmentation. In *Proceedings of the 2022 Conferen/ce on Empirical Methods in Natural Language Processing*, pages 7889–7900, 2022.

[9] R. Li, H. Chen, F. Feng, Z. Ma, X. Wang, and E. H. Hovy. Dual graph convolutional networks for aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6319–6329, 2021.

[10] P. Liu, S. Joty, and H. Meng. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, 2015.

[11] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, 2022.

[12] Y. Mao, Y. Shen, J. Yang, X. Zhu, and L. Cai. Seq2Path: Generating sentiment tuples as paths of a tree. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2215–2225, 2022.

[13] S. Mensah, K. Sun, and N. Aletras. An empirical study on leveraging position embeddings for target-oriented opinion words extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9174–9179, 2021.

[14] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C. N. dos Santos, B. Xiang, and S. Soatto. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[15] 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 *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8600–8607, 2020.

[16] 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 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, 2014.

[17] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos. SemEval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, 2015.

[18] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. AL-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, V. Hoste, M. Apidianaki, X. Tannier, N. Loukachevitch, E. Kotelnikov, N. Bel, S. M. Jiménez-Zafra, and G. Eryiğit. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, 2016.

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[20] Y. Tian, G. Chen, and Y. Song. Aspect-based sentiment analysis with type-aware graph convolutional networks and layer ensemble. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2910–2922. Association for Computational Linguistics, 2021.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[22] H. Wan, Y. Yang, J. Du, Y. Liu, K. Qi, and J. Z. Pan. Target-aspect-sentiment joint detection for aspect-based sentiment analysis. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9122–9129, 2020.

[23] WenxuanZhang, Y. Deng, X. Li, Y. Yuan, L. Bing, and W. Lam. Aspect sentiment quad prediction as paraphrase generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9209–9219, 2021.

[24] Z. Wu, C. Ying, F. Zhao, Z. Fan, X. Dai, and R. Xia. Grid tagging scheme for aspect-oriented fine-grained opinion extraction. *CoRR*, abs/2010.04640, 2020.

[25] L. Xu, H. Li, W. Lu, and L. Bing. Position-aware tagging for aspect sentiment triplet extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2339–2349, 2020.

[26] H. Yan, J. Dai, T. Ji, X. Qiu, and Z. Zhang. A unified generative framework for aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2416–2429, 2021.

[27] H. Yan, J. Dai, T. Ji, X. Qiu, and Z. Zhang. A unified generative framework for aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2416–2429, 2021.

[28] Y. Yin, C. Wang, and M. Zhang. Pod: Positional dependency-based word embedding for aspect term extraction. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1714–1719, 2020.

[29] W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam. Towards generative aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 504–510, 2021.

[30] W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam. A survey on aspect-based sentiment analysis: Tasks, methods, and challenges. *IEEE Trans. Knowl. Data Eng.*, 35(11):11019–11038, 2023.

[31] S. Zhou and T. Qian. On the strength of sequence labeling and generative models for aspect sentiment triplet extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12038–12050, 2023.

[32] X. Zhou, X. Wan, and J. Xiao. Representation learning for aspect category detection in online reviews. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 417–424, 2015.

[33] T. Zhu, J. Ren, Z. Yu, M. Wu, G. Zhang, X. Qu, W. Chen, Z. Wang, B. Huai, and M. Zhang. Mirror: A universal framework for various information extraction tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8861–8876, 2023.