Dynamic Order Template Prediction for Generative Aspect-Based Sentiment Analysis

Yonghyun Jun and **Hwanhee Lee**[*]
Department of Artificial Intelligence, Chung-Ang University
`{zgold5670, hwanheelee}@cau.ac.kr`

Abstract

Aspect-based sentiment analysis (ABSA) assesses sentiments towards specific aspects within texts, resulting in detailed sentiment tuples. Previous ABSA models often used static templates to predict all the elements in the tuples, and these models often failed to accurately capture dependencies between elements. Multi-view prompting method improves the performance of ABSA by predicting tuples with various templates and then assembling the results. However, this method suffers from inefficiencies and out-of-distribution errors. In this paper, we propose a Dynamic Order Template (DOT) method for ABSA, which dynamically creates an order template that contains only the necessary views for each instance. Ensuring the diverse and relevant view generation, our proposed method improves F1 scores on ASQP and ACOS datasets while significantly reducing inference time.[*]

1 Introduction

Aspect-based sentiment analysis (ABSA) aims to identify the sentiment of aspects in a given text rather than simply classifying the overall sentiment of the entire text. ABSA research evolves to generate quadruples consisting of four elements: 1) Aspect (A), 2) Category (C) for the type of A, 3) Opinion (O) for A, and 4) Sentiment (S) for A. Many recent studies, such as T5-paraphrase, tackle this problem using generative models (Zhang et al., 2021b). These approaches usually get review sentences as input and output the span of quadruples in fixed order form, such as "C is S because A is O" (Zhang et al., 2021a). However, this static single-order template cannot express the dependence between elements as in Figure 1 due to the autoregressive nature of the transformer (Vaswani et al., 2017). Moreover, the model output can heavily depend on the order of generation of each element (Hu et al., 2022).

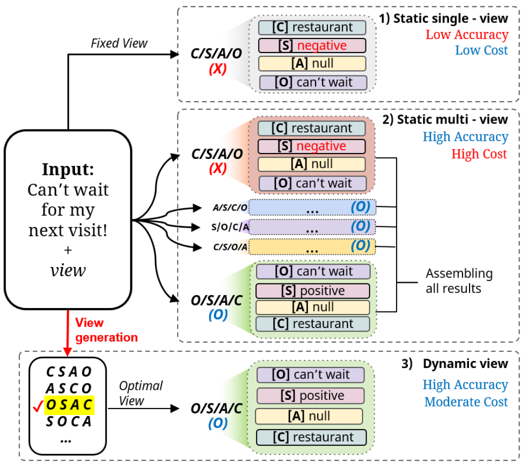


Figure 1: Comparison of three different generative ABSA methods. 1) static single-view, 2) static multi-view, and 3) dynamic-view prediction (ours).

Multi-view prompting (Gou et al., 2023) (MvP) deals with this issue by constructing order templates as a channel for "viewing" different perspectives in a sentence. As shown in Figure 1, MvP permutes all possible element orders and sorts them based on the entropy of the pre-trained model at the dataset level. Using this entropy, MvP samples top-k orders and adds these orders as a prompt template. During inference time, MvP conducts majority votes on generated sentiment tuples with various templates. Through this ensemble approach, MvP uses the intuition of solving problems from multiple views in human reasoning and decision (Stanovich and West, 2000), resulting in enhanced performance. However, we find that this static multi-view approach of MvP has several drawbacks: ***1) Inefficiency:*** Even for samples where the answer can be easily found and multiple views are not necessary, this method generates the same number of views, resulting in unnecessary computation that increases the inference

[*]Corresponding author.
[*]Our implementation is publicly available at `https://github.com/imsongpasimin/DOT`

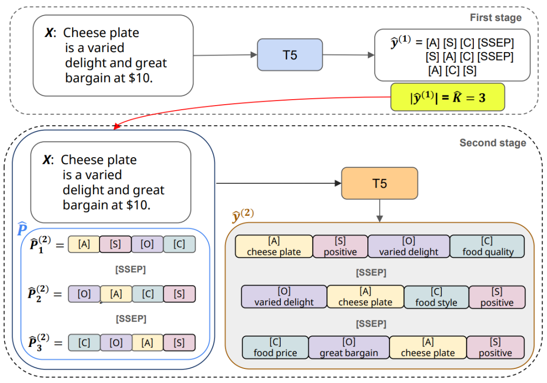

First stage
X: Cheese plate is a varied delight and great bargain at $10.
T5
$\hat{y}^{(1)}$ = [A] [S] [C] [SSEP] [S] [A] [C] [SSEP] [A] [C] [S]
$|\hat{y}^{(1)}| = \hat{K} = 3$
Second stage
X: Cheese plate is a varied delight and great bargain at $10.
$\hat{P}$
$\hat{P}_1^{(2)}$ = [A] [S] [O] [C]
[SSEP]
$\hat{P}_2^{(2)}$ = [O] [A] [C] [S]
[SSEP]
$\hat{P}_3^{(2)}$ = [C] [O] [A] [S]
T5
$\hat{y}^{(2)}$
[A] cheese plate [S] positive [O] varied delight [C] food quality
[SSEP]
[O] varied delight [A] cheese plate [C] food style [S] positive
[SSEP]
[C] food price [O] great bargain [A] cheese plate [S] positive

Figure 2: Overview of our proposed two stage method. We use two T5 models for each stage: one for generating the initial order template, the other for forming the final order template and generating sentiment tuples.

time. **2)** ***Limited Transferability***: MvP uses the number of views k as a hyperparameter, applying the same k value across all datasets during training and inference. However, since the optimal number of ensemble models varies according to the data domain, it requires manual adjustment of the k value for each dataset (Shahhosseini et al., 2022), which hinders the transferability to other datasets. To resolve the aforementioned shortcomings, we propose a Dynamic Order Template (DOT) method for ABSA that combines the advantages of both single-view and multi-view approaches. By prioritizing multiple views based on instance-level entropy, DOT aims to generate only the necessary number of views for each instance during inference. For an example that contains only one tuple, as in Figure 1, DOT dynamically creates only one view as an order template necessary to predict the tuple. After generating the views, DOT generates tuples using the views inside the order template. This phase operates in a multi-view manner, enabling us to retain the benefits of previous multi-view methods. Extensive experiments on four widely used sentiment quadruple prediction datasets, derived from ASQP (Pontiki et al., 2016; Zhang et al., 2022), ACOS (Cai et al., 2021a, 2023), and MEMD-ABSA (Cai et al., 2023), demonstrate that our method shows state-of-the-art performance with significantly lower inference time compared to the multi-view approach. Moreover, we show that our method is robust to domain shift compared to previous methods, resulting in higher transferability.

2 Method

Our proposed Dynamic Order Template (DOT) method is composed of two stages as in Figure 2. The first stage involves generating an initial order template (Sec 2.1) to predict the number of tuples. The second stage involves refining the initial template from stage 1 to produce the final order template and predicting the sentiment tuples based on it (Sec 2.2). For both stages, we map sentiment tuples (A, C, S, O) to marker tokens $[A]$, $[C]$, $[S]$, and $[O]$ respectively. Also, for the instances that contain multiple sentiment tuples, we indicate each tuple with the respective tokens and concatenate the targets with [SSEP] tokens. It is important to note that null or missing values in the datasets are encoded as the literal string "null". During template construction, our method maps this string directly to the designated marker tokens, eliminating the need for any special handling.

2.1 Stage 1: Generating Order Template

We assume that the number of sentiment tuples K_i in i^{th} instance present for each instance corresponds to the required number of views. Consequently, we are able to divide the complex sentiment tuple generation task into two relatively simpler subtasks: 1) predicting the number of sentiment tuples in the sentence, and 2) generating sentiment tuples with the exact number of first stage predictions. Building separate models specialized for each subtask and enabling collaboration between them was instrumental in achieving significant performance improvements. In this framework, each view is interpreted as the ordered scaffold for generating a single sentiment tuple, as depicted in Figure 2. This allows each prediction order to correspond one-to-one with a sentiment tuple in the second stage.

We observe that instead of directly using the value of K_i as the target, sampling views corresponding to K_i as the target for the model to generate leads to more accurate prediction of the number of tuples in the given instance (More details are in Appendix C). To establish the view sampling strategy, we start by ranking all possible views generated through permutations through each entropy score, following (Hu et al., 2022). Specifically, we calculate entropy of each view v in instance-level with vanilla T5 by calculating conditional generation probability as follows:

$$\mathcal{E}_{i,v} = -\sum P(v|x_i) \log P(v|x_i) \tag{1}$$

Here, $\mathcal{E}_{i,v}$ is the entropy of the total sequence when the i_{th} instance is input into the T5 model and v

is the output. At this time, we note that actually utilizing only A, C, S during the first stage notably facilitates the training process in the second stage. We provide a detailed analysis on excluding O in Appendix C.2. After computing the entropy, we sort the views by the entropy in ascending order to get the ranked set of views $P_i^{(1)}$. And then we sample the top K_i views for each sample and concatenate these views as an order template. Using the original i^{th} input sentence, we train the T5 model to generate the first-stage target $y_i^{(1)}$ as follows:

$$y_i^{(1)} = P_{i,1}^{(1)} \text{ [SSEP] } P_{i,2}^{(1)} \text{ [SSEP] } \dots P_{i,K_i}^{(1)},$$

where $P_{i,K_i}^{(1)}$ denotes K_i^{th} view in $P_i^{(1)}$. We set the loss function to train the T5 model as in Equation (2), where $|B|$ denotes the batch size of the model. The scaling factor is omitted for simplicity.

$$\mathcal{L}_1 = -\sum_{i=1}^{|B|}\sum_{t=1}^{T} \log p(\boldsymbol{y_{i,t}^{(1)}}|\boldsymbol{x_i}, \boldsymbol{y_{i,<t}^{(1)}}) \tag{2}$$

2.2 Stage 2: Sentiment Tuple Generation

In the second stage, the model is trained to generate the sentiment tuple of a given instance using the number of sentiment tuples (i.e. K_i). Different from the first stage, we need to generate all elements in sentiment quadruples including O in this stage. Hence, we re-rank all views to pick K_i views including O (i.e. (A, C, S, O)). Here, we adopt the same strategy as in the first stage, using entropy to form a ranked set of views, $P_i^{(2)}$. We then sample top K_i views from $P_i^{(2)}$ and add them as an order template prompt P_i to original input sentence. We design the second stage target $y_i^{(2)}$ by aligning each sentiment tuple with an order template, ensuring that the model learns to generate different tuples for different perspectives. Also, we place the corresponding elements next to each marker token within P_i as follows:

$$y_i^{(2)} = P_{i,1}^{(2)} \otimes tuple_1 \text{ [SSEP] } \dots P_{i,K_i}^{(2)} \otimes tuple_{K_i},$$

where $P_{i,K_i}^{(2)}$ represents K_i^{th} view in $P_i^{(2)}$ and $tuple_{K_i}$ is the K_i^{th} sentiment tuple for given instance. $\otimes$ denotes an interleaved combination between marker tokens and elements. Detailed examples for both stages are present in Appendix E. We design the loss function for training the T5 model in second stage as follows.

$$\mathcal{L}_2 = -\sum_{i=1}^{|B|}\sum_{t=1}^{T} \log p(\boldsymbol{y_{i,t}^{(2)}}|\boldsymbol{x_i}, \boldsymbol{P_i}, \boldsymbol{y_{i,<t}^{(2)}}) \tag{3}$$

2.3 Two-stage inference

During inference time, two stages are conducted sequentially. In the first stage, the model generates the initial order template, denoted as $\hat{y^{(1)}}$. In the second stage, we count the number of generated views from $\hat{y^{(1)}}$ to set $\hat{K}$. Using $\hat{K}$, we sample the top $\hat{K}$ views from the newly ranked set of views and constructs the final order template, referred to as $\hat{P}$. Finally, $\hat{P}$ is directly appended to the inference sentence, enabling the generation of different sentiment tuples for each view in $\hat{P}$. Throughout inference, we employ a constrained-decoding strategy (De Cao et al., 2020) to ensure that the output at each stage conforms to the required format. The overall two-stage process is described in Figure 2.

3 Experiment

3.1 Benchmark Datasets

We adopt two widely used ABSA datasets: ASQP and ACOS, where the task is to predict sentiment quadruples. For ASQP task, we use rest15 (R15) and rest16 (R16) datasets released from (Pontiki et al., 2016; Zhang et al., 2022). For ACOS task, we use laptop16(Lap) and rest16(Rest) datasets constructed by (Cai et al., 2021a; Pontiki et al., 2016). Also, we adopt additional ACOS benchmarks from MEMD datasets (Restaurant, Laptop, Books, Clothing, Hotel) (Xu et al., 2023) which use a different source from the previous datasets. We refer to the Restaurant and Laptop datasets in MEMD as M-Rest and M-Laptop, respectively, for the sake of clarity.

3.2 Baselines

We benchmark our approach against a suite of extraction- and generation-based baselines. For a subset of baselines, we confine evaluation to the four datasets—R15, R16, Laptop, and Restaurant—because these methods were originally optimized for ASQP or ACOS tasks. Other baselines are assessed on their full applicable dataset range. This selective restriction ensures a fair comparison. The complete list of baselines is as follows: *TAS-BERT* (Wan et al., 2020) jointly extracts and detects sentimental tuples. *Extract-Classify* (Cai et al., 2021b) divide the task into two stages: extraction and classification. *One-ASQP (large)* (Zhou et al., 2023) identify the aspect-opinion-sentiment (AOS) triplets simultaneously. *Seq2Path* (Mao et al., 2022) generates sentiment tuples as multiple paths of a

Methods	**ASQP**		**ACOS**		**MEMD**					**Avg**	**Time(s)**
	R15	**R16**	**Lap**	**Rest**	**M-Rest**	**M-Lap**	**Books**	**Clothing**	**Hotel**		
TAS-BERT	34.78	43.71	27.31	33.53	-	-	-	-	-	-	-
Extract-Classify	36.42	43.77	35.80	44.61	-	-	-	-	-	-	-
One-ASQP (large)	-	-	41.56	60.69	-	-	-	-	-	-	-
Seq2Path	-	-	42.97	58.41	-	-	-	-	-	-	-
AugABSA	50.01	60.88	-	-	-	-	-	-	-	-	-
SCRAP	49.93	**62.48**	-	-	-	-	-	-	-	-	-
Paraphrase	46.93	57.93	43.51	61.16	57.38	35.07	39.30	43.00	68.79	50.34	40.63
DLO	48.18	59.79	43.64	59.99	57.07	35.56	42.63	43.35	**70.27**	51.16	260.74
MvP	51.04	60.39	43.92	**61.54**	58.12	35.25	42.57	**43.94**	69.06	51.76	2161.81
GPT-4o	40.45	47.29	24.77	46.53	35.11	20.69	30.39	40.27	24.84	34.48	-
LLaMa-3.1-8b	37.52	47.60	40.07	54.06	38.10	31.16	28.62	32.21	44.62	39.33	-
Qwen-2.5-7b	29.93	39.34	12.48	33.56	25.63	24.13	17.77	18.09	38.03	26.66	-
Mistral-7b	44.14	51.96	39.02	53.02	41.28	26.80	26.54	21.81	40.35	38.32	-
DOT (Ours)	**51.91**	61.24	**44.92**	59.25	**58.25**	**39.02**	**43.02**	43.37	69.94	**52.28**	298.17

Table 1: F1 scores for ABSA on nine datasets. Best results are in bold, second-best underlined. Results are averaged over five seeds. Time denotes average inference time.

tree, and automatically selects a valid one. *AugABSA* (Wang et al., 2023) generates a original text based on augmented sentiment quadruples. *SCRAP* (Kim et al., 2024) optimizes the model to generate extract-then-assign reasonings and the corresponding sentiment quadruplets in sequence. *Paraphrase* (Zhang et al., 2021a) formulates a paraphrase generation process for ABSA with a single fixed order. *DLO* (Hu et al., 2022) augments data via the multiple order templates. *MvP* aggregates sentiment tuples generated from multiple orders of prompts via ensembling. Also, we benchmark popular LLMs such as GPT-4o (Hurst et al., 2024), LLaMa-3.1-8b (Dubey et al., 2024), Qwen-2.5-7b (Yang et al., 2025), and Mistral-7b (Jiang et al., 2023). Detailed setups for LLMs are in Appendix H.

3.3 Implementation Details

We utilize the pre-trained T5-base (Raffel et al., 2020) model as the backbone for the first stage. We also use the model trained in the first stage as the backbone for the second stage, allowing us to leverage a tuned initial point for the ABSA dataset to have the regularization effect inspired by (Fu et al., 2023). Additionally, we observe that the label of the datasets (i.e. sentiment tuples) irregularly contains stop words. Thus, we eliminate irregularities in tuples through stop-word filtering in the second stage. We provide a detailed analysis and filtering process in Appendix A.

3.4 Results

Performance Comparison We use the F1 score, which is a standard metric for ABSA, to measure the performance of the systems. Table 1 demonstrates that our DOT framework achieves state-of-the-art results, ranking first on five of the nine benchmarks and second on three others. To further elaborate, a significant factor contributing to the enhanced performance is our approach of dividing the complex sentiment quadruple generation task into two subtasks, each handled by dedicated models. As a result, our method achieves better performance compared to a single model handling multi-view processing alone. Crucially, the effectiveness of this two-stage approach depends on the accuracy of the first task, which we comprehensively presented in Appendix C. However, its performance is slightly lower on the Rest and Clothing datasets, which we analyze in Section F.

Inference time We also measure inference time using T5-base model for all baselines. We check inference time for each dataset, and average them. As in Table 1, we dramatically reduce inference time particularly compared to the multi-view methods such as MvP (Gou et al., 2023), by predicting solely the necessary number of views for each sample. On the other hand, single view inferences (Zhang et al., 2021a; Hu et al., 2022) and extraction approaches (Wan et al., 2020; Cai et al., 2021b; Zhou et al., 2023) can be more memory- and time-efficient(Zhou et al., 2023), they generally exhibit inferior performance compared to generative methods. We provide more details on the inference time in Appendix D.

Transferability To examine the transferability of each model, we conduct an in-depth experiment on cross-dataset evaluation. We group the datasets

Train	*SemEval*		*Yelp*	
Test	*SemEval*	*Yelp*	*Yelp*	*SemEval*
Paraphrase	52.38	38.52(-11.86)	57.38	44.88(-12.50)
MvP_3	55.62	34.42(-21.20)	57.27	41.72(-15.55)
MvP_9	56.89	35.02(-21.87)	56.98	42.52(-14.46)
MvP_{15}	**57.66**	35.21(-21.45)	58.12	41.94(-16.18)
DOT	57.47	**39.88**(-17.59)	**58.25**	**46.97**(-11.28)

Table 2: Cross-dataset evaluation results for validating the effect of domain shift.

Model Configuration	**Average F1**
Full Model	54.33
w/o filtering	53.31 (-1.02)
w/o stage division	52.73 (-1.60)
w/o entropy score	52.53 (-1.80)
w/o multi view	52.31 (-2.02)
w/o stage division, entropy score	50.04 (-4.29)
w/o filtering, stage division, entropy score	45.80 (-8.53)

Table 3: Ablation study for the proposed method, which shows the average F1 across ASQP and ACOS.

into two sources: SemEval (Pontiki et al., 2016) (R15, R16, Rest) and Yelp (M-Rest), and assess performance by training on one group and testing on the other in a zero-shot setting. For the MvP model, we vary the number of views (3, 9, and 15) to evaluate sensitivity in static multi-view methods, while T5-paraphrase uses a static single order. As shown in Table 2, our model significantly outperforms the baselines in cross-dataset evaluation. Although T5-paraphrase suffers a smaller performance drop, it still falls behind our method, and MvP shows notable degradation regardless of view count. These results demonstrate that our model effectively identifies optimal views even for out-of-domain datasets.

Performance Across Model Scales Beyond using T5-base as the sole backbone, we extend our experiments to examine how performance varies with model scale, specifically testing T5-small and T5-large. In this setup, we employ the same backbone model for both stages. As shown in Figure 3, performance consistently improves across all benchmarks as the backbone scale increases, supporting the scaling law (Kaplan et al., 2020) showing larger models yield better accuracy. Nonetheless, these gains do not fully compensate for the inefficiency introduced by loading larger models and performing multiple inference steps. Considering the trade-off between accuracy and computational cost, T5-base emerges as the most balanced choice.

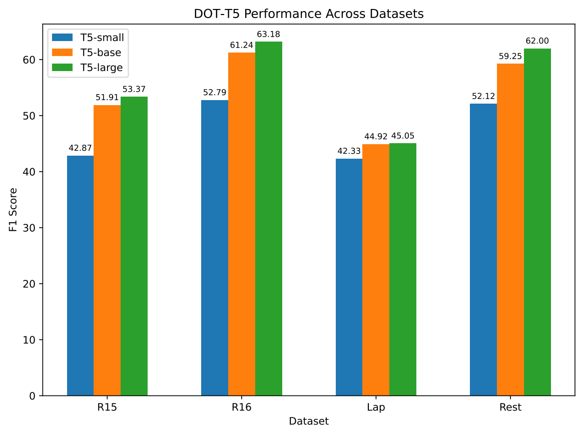


Figure 3: Bar chart illustrating F1 performance variations across T5-small, T5-base, and T5-large backbones on R15, R16, Laptop, and Rest benchmarks.

Ablation study To further investigate the effectiveness of each component of our framework, we conduct an ablation study and present the results in Table 3. Firstly, we record the results without stop-word filtering. Also, we unify the two stages into one, directly generating multiple order templates and tuples without including order prompting. Additionally, we evaluate the results of sampling the views randomly, checking whether the entropy score is valid. Lastly, we exclude the multi-view approach by training and testing our model using only the view with the lowest entropy for each instance as the order template. We perform an ablation study by excluding one or more of the four components of our method mentioned earlier. By observing the gaps between these variants with the original model, we verify the effectiveness of each component of our method. For a more comprehensive ablation study, please refer to Appendix C.

4 Conclusion

We propose Dynamic Order Template (DOT) method for generative ABSA, addressing inefficiencies and out-of-distribution errors. Experiments on nine datasets demonstrate that DOT achieves state-of-the-art performance with reduced inference time, effectively balancing the strengths of previous single and multi-view approaches for ABSA.

Limitation

Our DOT method is highly efficient and powerful, yet it still has several limitations. DOT method consists of two stages: view generation and tuple generation. We train separate models for each task, and these two models perform inference sequentially. This form is not end-to-end, so it is disadvantageous in terms of training time and memory.

Also, since we directly connect first stage and second stage, if any errors occur, the errors may propagate and magnify as it moves to the subsequent stage. It results in relatively large standard deviation for different seeds as reported in Table 10. However, by splitting the task of 'predicting the appropriate number of tuples' into two sub-tasks—'predicting the appropriate number of tuples' and 'accurately predicting the tuples'—it becomes significantly easier to achieve accurate results in both areas, thereby enhancing overall performance in our work.

Finally, we define the number of necessary views as the number of sentiment tuples for simplicity and efficiency. A more complex yet refined method for determining the necessary number of views could be further explored in future research.

Ethics Statement

This study utilizes the various datasets for aspect-based sentiment analysis, which are accessible online. Additionally, we have properly cited all the papers and sources referenced in our paper. We plan to release the pre-trained model and the code for training the proposed system.

Acknowledgement

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [RS-2021-II211341, Artificial Intelligence Graduate School Program (Chung-Ang University)].

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hongjie Cai, Nan Song, Zengzhi Wang, Qiming Xie, Qiankun Zhao, Ke Li, Siwei Wu, Shijie Liu, Jianfei Yu, and Rui Xia. 2023. Memd-absa: a multi-element multi-domain dataset for aspect-based sentiment analysis. *arXiv preprint arXiv:2306.16956*.

Hongjie Cai, Rui Xia, and Jianfei Yu. 2021a. 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.

Hongjie Cai, Rui Xia, and Jianfei Yu. 2021b. 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.

Siva Uday Sampreeth Chebolu, Franck Dernoncourt, Nedim Lipka, and Thamar Solorio. 2023. A review of datasets for aspect-based sentiment analysis. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 611–628.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Alabhya Farkiya, Prashant Saini, Shubham Sinha, and Sharmishta Desai. 2015. Natural language processing using nltk and wordnet. *Int. J. Comput. Sci. Inf. Technol*, 6(6):5465–5469.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12799–12807.

Zhibin Gou, Qingyan Guo, and Yujiu Yang. 2023. Mvp: Multi-view prompting improves aspect sentiment tuple prediction. *arXiv preprint arXiv:2305.12627*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Mengting Hu, Yike Wu, Hang Gao, Yinhao Bai, and Shiwan Zhao. 2022. Improving aspect sentiment quad prediction via template-order data augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7900.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Jieyong Kim, Ryang Heo, Yongsik Seo, SeongKu Kang, Jinyoung Yeo, and Dongha Lee. 2024. Self-consistent reasoning-based aspect-sentiment quad prediction with extract-then-assign strategy. *arXiv preprint arXiv:2403.00354*.

M Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yue Mao, Yi Shen, Jingchao Yang, Xiaoying Zhu, and Longjun Cai. 2022. Seq2path: Generating sentiment tuples as paths of a tree. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2215–2225.

Maria Pontiki, Dimitrios Galanis, Harris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Mohsen Shahhosseini, Guiping Hu, and Hieu Pham. 2022. Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Machine Learning with Applications*, 7:100251.

Marina Solnyshkina, Radif Zamaletdinov, Ludmila Gorodetskaya, and Azat Gabitov. 2017. Evaluating text complexity and flesch-kincaid grade level. *Journal of social studies education research*, 8(3):238–248.

Keith E. Stanovich and Richard F. West. 2000. Advancing the rationality debate. *Behavioral and Brain Sciences*, 23(5):701–717.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hai Wan, Yufei Yang, Jianfeng Du, Yanan Liu, Kunxun Qi, and Jeff Z Pan. 2020. Target-aspect-sentiment joint detection for aspect-based sentiment analysis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9122–9129.

An Wang, Junfeng Jiang, Youmi Ma, Ao Liu, and Naoaki Okazaki. 2023. Generative data augmentation for aspect sentiment quad prediction. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (* SEM 2023)*, pages 128–140.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Ting Xu, Huiyun Yang, Zhen Wu, Jiaze Chen, Fei Zhao, and Xinyu Dai. 2023. Measuring your aste models in the wild: A diversified multi-domain dataset for aspect sentiment triplet extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2837–2853.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Wenxuan Zhang, Yang Deng, Xin Li, Lidong Bing, and Wai Lam. 2021a. Aspect-based sentiment analysis in question answering forums. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4582–4591.

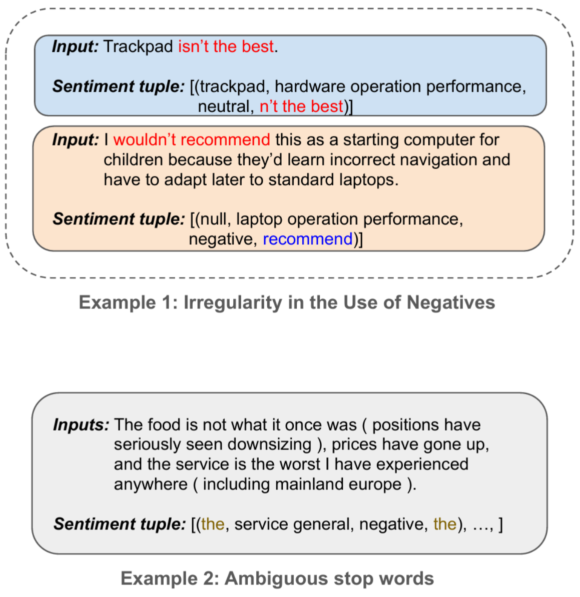
Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2021b. Towards generative aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 504–510.

Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2022. A survey on aspect-based sentiment analysis: Tasks, methods, and challenges. *IEEE Transactions on Knowledge and Data Engineering*.

Junxian Zhou, Haiqin Yang, Yuxuan He, Hao Mou, and JunBo Yang. 2023. A unified one-step solution for aspect sentiment quad prediction. *arXiv preprint arXiv:2306.04152*.

A Detailed Experimental Setup

We use AdamW optimizer (Loshchilov and Hutter, 2017) with a learning rate of 1e-4 for training two T5 models. We set the batch size to 16 for training and 24 for inference. We train the first stage model for 30 epochs, and train 40 epochs for the second stage.



Input: Trackpad isn't the best.

Sentiment tuple: [(trackpad, hardware operation performance, neutral, n't the best)]

Input: I wouldn't recommend this as a starting computer for children because they'd learn incorrect navigation and have to adapt later to standard laptops.

Sentiment tuple: [(null, laptop operation performance, negative, recommend)]

Example 1: Irregularity in the Use of Negatives

Inputs: The food is not what it once was (positions have seriously seen downsizing), prices have gone up, and the service is the worst I have experienced anywhere (including mainland europe).

Sentiment tuple: [(the, service general, negative, the), ...,]

Example 2: Ambiguous stop words

Figure 4: Two examples of irregularity of stop words. Note that these examples are the not all of the stop-word problems.

We also note that the dataset labels (i.e., the sentiment tuples) sporadically include stop words. For example, as in the first example of Figure 4, the inclusion of negations in the opinion terms is inconsistent. Also, as in the second example, element tuples sometimes contain ambiguous and meaningless stop words as elements. As a result, the fine-tuned model sometimes generates sentiment tuples containing stop words irregularly. It can yield critical performance degradation, even though they don't affect the meaning of the sentiment elements. To resolve the problem of stop words, we filter these stop words using nltk package(Farkiya et al., 2015) for both generated results and dataset labels. We use four RTX 4090 GPUs to train and evaluate all of the models.

B Case Study

We conduct a case study and analyze the properties of the outputs generated by the proposed method. As depicted in Figure 5, we classify the output results into three main cases.

The first case involves sentences that do not require multiple views for accurate prediction. For these sentences, our model succeeds in making efficient predictions using only a single view. We observe that this case is the most common type in our study, significantly contributing to the model's efficiency.

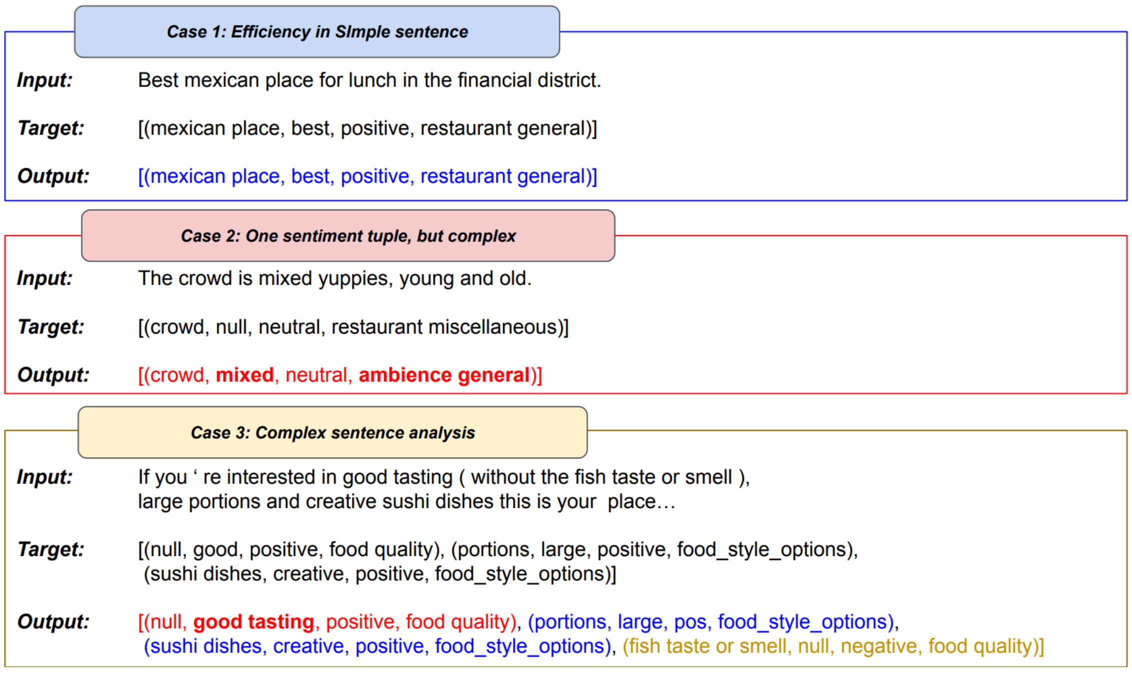
The second shows an example that predicts requires fewer views, but the example actually requires more views. Our analysis reveals that such cases frequently occurs with implicit O. As shown in Table 1, this suggests that our model's performance might lag behind other baselines on the ACOS Rest16 dataset, which contains many samples with implicit A and O. Additionally, the model struggles with predicting infrequent C in the training set. Incorporating the concept of self-information and defining the necessary number of views based on the 'amount of information in a sample' could effectively address this issue.

The final case involves cases with multiple sentiment tuples and longer lengths. We explain that errors in this scenario stem from two main reasons. Firstly, longer sentences include extended phrases that modify A or O. Including all these modifiers as elements often leads to errors, a common problem across different models that requires an alternative solution. Secondly, errors occur when the number of tuples is incorrectly predicted in the first stage. If the predicted number of tuples is insufficient, some target sentiment tuples might be overlooked. Conversely, overestimation leads to the extraction of irrelevant aspects, as depicted in the Figure 5. However, we optimize the first stage to reduce tuple count errors, which helped mitigate performance drops by minimizing incorrectly generated or overlooked tuples.

C Depth Analysis on First Stage

C.1 Accuracy on the Number of Views

We assess the accuracy of predicting the value of $\hat{K}$ and present the results in Table 4. We evaluate the output by comparing it to the number of labeled sentiment tuples using RMSE and accuracy. We carefully implement the first stage baselines to compare our method properly as follows: *Random*: We find that the number of sentiment tuples in the training dataset is mostly in the range of 1 to 6. For each inference, we randomly sample one of the 6 numbers and compare it with our first stage result. *Majority*: We also reveal that about



Case 1: Efficiency in SImple sentence

Input: Best mexican place for lunch in the financial district.

Target: [(mexican place, best, positive, restaurant general)]

Output: [(mexican place, best, positive, restaurant general)]

Case 2: One sentiment tuple, but complex

Input: The crowd is mixed yuppies, young and old.

Target: [(crowd, null, neutral, restaurant miscellaneous)]

Output: [(crowd, **mixed**, neutral, **ambience general**)]

Case 3: Complex sentence analysis

Input: If you ' re interested in good tasting (without the fish taste or smell), large portions and creative sushi dishes this is your place…

Target: [(null, good, positive, food quality), (portions, large, positive, food_style_options), (sushi dishes, creative, positive, food_style_options)]

Output: [(null, **good tasting**, positive, food quality), (portions, large, pos, food_style_options), (sushi dishes, creative, positive, food_style_options), (fish taste or smell, null, negative, food quality)]

Figure 5: Case study for the three main types of results. Blue one denotes correct, red one denotes incorrect, and the yellow one denotes irrelevant.

60 percent of labels consist of a single tuple. We construct a baseline that predicts only 1 for the number of tuples, to check whether our model has the ability to predict the number of sentiment tuples of a sentence. *Classification*: We adopt the RoBERTa model (Liu et al., 2019) to evaluate the results when treating the prediction of the number of views as a sequence classification task. We set the classes based on the number of sentiment tuples. As shown in Figure 6, the distribution of tuple counts is skewed towards the lower end, with instances containing more than seven tuples being nearly non-existent. Consequently, we limit the categories from 1 to 6 and clip instances with 7 or more tuples to 6. Additionally, to address label imbalance, we employ a weighted loss function, where the weights are set as the inverse of the frequency ratio for each category as in Equation (4). We use the same notation as in Section 2.1, and $\mathcal{I}\,()$ denotes the indicator function. This approach enables the model to effectively classify even the less represented classes.

$$\boldsymbol{W_c} = \frac{|D|}{\sum_D \mathcal{I}(min(|\boldsymbol{y}|, \boldsymbol{6}) = \boldsymbol{c})} \qquad (c \in [1, 6])$$
$$\mathcal{L}_{cls} = -\sum_{i=1}^{|B|} \boldsymbol{W_{k_i}} \log p(\boldsymbol{k_i}|\boldsymbol{x_i})\mathcal{I}\,(\boldsymbol{k_i} \leq \boldsymbol{6}) \tag{4}$$

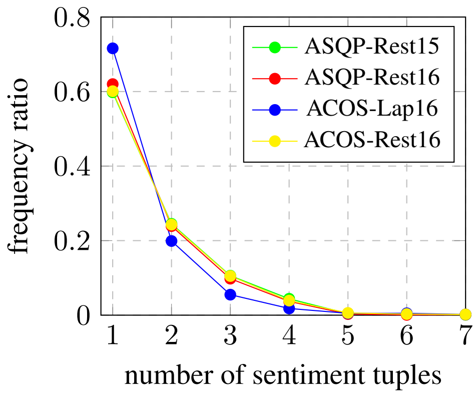


Figure 6: Distribution of the number of sentiment tuples. The sources are from training datasets of each task. We normalize each count by dividing it by the total number of data points. The number of tuples is clipped to 7.

C.2 Effect of Element Exclusions

We analyze the impact of excluding various marker tokens, including the [O] token representing opinions, to determine which token exclusions contribute to performance improvements. Additionally, we experiment with cases where no element exclusion is performed. In this section, we have also included the second stage results to provide a detailed comparison of the overall performance.

As in Table 4, our proposed method outperforms the other baselines and nearly predicts the actual distribution of sentiment tuples within a small mar-

Methods	First stage RMSE	First stage Acc.	Second stage F1 score
Random	2.80	18.89	-
Majority	0.99	63.39	-
Classification	0.83	61.90	-
DoT_{first}	0.54	77.83	**54.33**
exclude $[C]$	0.54	77.53	53.91
exclude $[A]$	**0.53**	77.77	53.71
exclude $[S]$	0.54	77.65	53.55
full elements	0.55	**78.22**	53.94

Table 4: First stage results for each main baseline and exclusion of specific tokens. We report average RMSE loss and accuracy for first stage, and F1 score for second stage.

gin of error. This result justifies the use of the output from the first stage in the second stage. The first stage results in Table 4 do not exhibit significant performance differences among various exclusion. However, for the second stage results, which serve as the final output of this task, we observe a significant performance difference. The performance in the second stage is generally higher when O is omitted because generating O correctly is the most difficult and crucial task in quadruple prediction (Chebolu et al., 2023). If O is not trained in the first stage and is reused in the second stage, the model appears to focus more on learning about O compared to other elements, which already have some level of information.

D Computing Inference Time

We compare inference times based on view methods across different dataset sizes. The dataset consisted of randomly sampled test data from laptop16, with 200, 400, 600, and 800 samples. The baselines were set as static single view (T5-paraphrase) and static multi view (MvP), with the number of views for the multi view fixed at 15. Figure 7 shows that we not only dramatically reduce inference time of utilizing multi views, but also reduce the rate of increase in inference time with respect to the number of datasets. On the other hand, in terms of single view, we significantly increase F1 performance while suppressing the increase in inference time and the rate of its increase. These results suggest that the efficiency of our method becomes more pronounced as the dataset size increases.

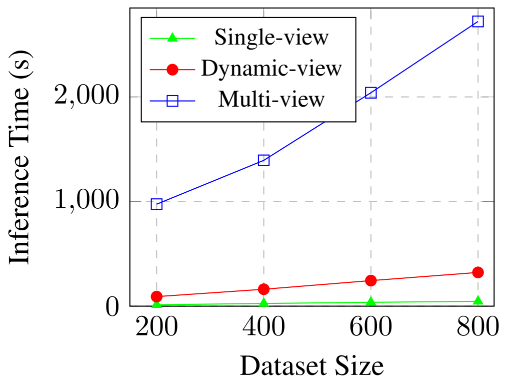


Figure 7: Inference time among dataset size for each model.

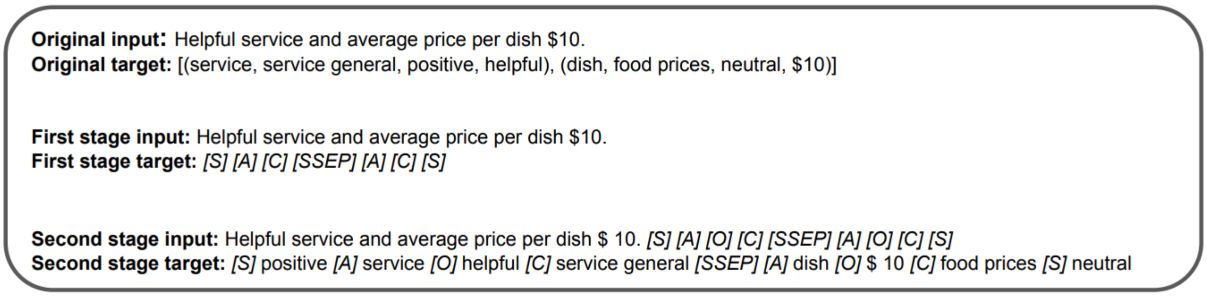
E Input and Target Examples for Each Stage

In Figure 8, we provide detailed examples for input and output pairs in each stage. The input sentences in the dataset are presented in a basic sentence structure, while the labels consist of lists of sentiment tuples. To preprocess this data, during the first stage, the original input sentence is kept unchanged, and the target is set as the initial order template, which consisted of a number of views corresponding to the number of sentiment tuples in the label. In the second stage, the input is processed by appending the final order template as a prompt to the original input sentence. The target is then constructed by adjusting the order of the elements within the sentiment tuples to align with the corresponding views in the order template.

F Analysis on Implicit Term

In Table 1, DOT suffers from predicting sentiment tuples in Rest and Clothing domains. We noted that the ACOS dataset contains a significant number of instances with implicit aspects or implicit opinions. Additionally, we discovered that the Rest and Clothing dataset are smaller in scale compared to other ACOS datasets. The scale of each dataset and the number of instances containing implicit terms are recorded in Table 5. Based on these observations, we hypothesized that the size of the dataset and the distribution of implicit terms contribute to the performance degradation observed in the Rest and Clothing datasets.

As shown in Table 6, it is evident that the F1 score for instances containing implicit terms in the Rest dataset is significantly lower compared to using the paraphrase method. Additionally, we



Original input: Helpful service and average price per dish $10.
Original target: [(service, service general, positive, helpful), (dish, food prices, neutral, $10)]

First stage input: Helpful service and average price per dish $10.
First stage target: *[S] [A] [C] [SSEP] [A] [C] [S]*

Second stage input: Helpful service and average price per dish $ 10. *[S] [A] [O] [C] [SSEP] [A] [O] [C] [S]*
Second stage target: *[S]* positive *[A]* service *[O]* helpful *[C]* service general *[SSEP] [A]* dish *[O]* $ 10 *[C]* food prices *[S]* neutral

Figure 8: Examples for input and target from original dataset for both first and second stage.

Datasets	**ASQP**		**ACOS**		**MEMD**				
	R15	**R16**	**Lap**	**Rest**	**M-Rest**	**M-Laptop**	**Books**	**Clothing**	**Hotel**
total samples	834	1264	2934	1530	3622	2863	2092	1674	2481
implicit samples	272	446	1826	822	1801	1751	1523	1083	1278
implicit sample %	32.6	35.3	62.2	53.7	49.7	61.2	72.8	64.7	51.5

Table 5: The size of each dataset and the number of samples containing implicit terms. For ease of comparison, We also provide the percentage of samples with implicit terms relative to the total number of samples. It is evident that the implicit term ratio in the ACOS dataset is higher compared to that in the ASQP dataset.

Methods	**Rest**	**M-Rest ¼**	**M-Rest ½**	**M-Rest full**
Paraphrase	**50.06**	**40.26**	47.77	49.09
DOT	44.84	35.74	**47.81**	**49.49**

Table 6: F1 scores only for samples containing implicit terms. We report the performance in Rest dataset and the performance trends across different dataset scales.

observed a performance degradation when training on a randomly selected quarter of the M-Rest dataset. However, as the amount of training data from the M-Rest dataset increased, the performance on implicit terms improved, eventually surpassing the F1 score of the paraphrase method in the full M-Rest dataset. This result demonstrates that the small size of the dataset with a high proportion of implicit terms is the primary cause of the performance degradation in the Rest and Clothing dataset. It also suggests that the performance is likely to improve as the dataset size increases.

G Additional Analysis

In this section, we conduct an in-depth analysis of various aspects of our model. For a comprehensive evaluation, we use Paraphrase and MvP as baselines, running identical experiments for comparison. We assess performance across multiple tasks using several benchmarks, including R15, R16, Lap, Rest, and M-Rest.

Different Backbone Model We conduct the experiment using different encoder-decoder based model, BART (Lewis, 2019) as backbone model. We utilize BART with the same hyperparameters and data processing techniques applied to the T5 model for three methods including ours. However, as in Table 7, we observe a noticeable decline in overall F1-scores for all models, primarily due to insufficient hyperparameter tuning compared to T5. Nevertheless, as shown in the results, our method still outperforms the baseline models with BART, suggesting that its effectiveness is not highly dependent on the choice of backbone model.

Methods	**ASQP**		**ACOS**		
	R15	**R16**	**Lap**	**Rest**	**M-Rest**
Paraphrase	31.77	38.15	30.98	36.65	35.74
MvP	33.48	41.01	32.57	**40.40**	40.30
DOT	**35.98**	**41.73**	**33.12**	39.61	**40.91**

Table 7: F1 score on benchmark datasets using BART as the backbone model.

Complex Sentences As mentioned in Appendix B, processing long and complex contexts is a well-known challenge, and our model performs similarly to others in this regard. We define complex sentences as those containing more than three sentiment tuples, exceeding 22 words in length, or having a Flesch-Kincaid Grade Level (Solnyshkina et al., 2017) of 9 or higher, representing the top 20% for each criterion. We sample these complex sentences and evaluate the F1 scores for these samples. As shown in the Table 8, performance degradation in complex sentences is a common issue across all models. We attribute the larger performance drop in our model compared to MvP to the fact that it uses fewer views, which limits its capacity to thoroughly analyze complex sentences. Nevertheless, our model's final Complex F1 score remains close to that of MvP and surpasses that of the Paraphrase model.

Methods	ASQP		ACOS		
	R15	R16	Lap	Rest	M-Rest
Paraphrase	44.94	55.73	37.15	**56.12**	54.37
MvP	46.71	**58.00**	37.68	56.06	**58.23**
DOT	**46.93**	57.70	**38.57**	54.16	57.72

Table 8: F1 scores evaluated on complex samples only.

Training Complexity Our method may appear complex due to the numerous components that require training. However, since our method involves simply training the T5 model twice without complex optimization procedures, the overall training time is not significantly longer than that of other models. As shown in the Table 9, even though our model uses 30 and 40 epochs for two stages of training—more than the 20 epochs used in MvP—the total training time remains much shorter than that of MvP. In terms of memory usage, only two T5 models are allocated in memory, so the memory consumption does not increase exponentially compared to existing models.

Methods	ASQP		ACOS		
	R15	R16	Lap	Rest	M-Rest
Paraphrase	212.83	314.18	652.31	349.79	815.48
MvP	3883.74	5008.84	11006.07	6169.02	14634.71
DOT	1161.73	1648.61	3310.63	1814.41	4157.93

Table 9: Training duration for each benchmark.

Standard Deviation We conduct experiments using five different random seeds and calculate the standard deviation of the outcomes. Results are reported in Table 10. Our findings indicate that our model exhibits a higher overall standard deviation compared to other baselines. This can be attributed to the structure of the method, where an error at one stage is likely to propagate and accumulate. However, it is important to note that the absolute value of the standard deviation is not significantly large. In fact, the higher variation suggests that the model may possess greater potential to achieve stronger performance.

Methods	ASQP		ACOS		
	R15	R16	Lap	Rest	M-Rest
Paraphrase	± 0.44	± 0.64	± 0.26	± 0.68	± 0.38
MvP	± 0.54	± 0.29	± 0.48	± 0.72	± 0.48
DOT	± 0.74	± 0.85	± 1.01	± 0.76	± 0.42

Table 10: Standard deviation of outcomes in Table 1.

H Detailed Setups for LLM Experiments

As in Table 1, we perform the ABSA task using the GPT-4o, LLaMa-3.1-8B, and Mistral-7B models, compairing the results with our DOT model. For the GPT model, we utilize in-context learning (Brown et al., 2020). We randomly sample 10 instances and combine them with instruction format, and add it as a prompt. For the other three open-source LLMs, we employ instruction tuning (Wei et al., 2021) with the training dataset for fine-tuning, using the same instructions as in GPT prompts. To ensure stable model training during fine-tuning, we utilize the LoRa (Hu et al., 2021). We present the specific prompts and framework in Figure 9.

```
According to the following sentiment elements definition:

- The 'aspect term' refers to a specific feature, attribute, or aspect of a product
    or service that a user may express an opinion about, the aspect term might be '
    null' for implicit aspect.
- The 'opinion term' refers to the sentiment or attitude expressed by a user towards
     a particular aspect or feature of a product or service, the aspect term might
    be 'null' for implicit opinion.
- The 'aspect category' refers to the category that aspect belongs to, and the
    available categories includes: {dataset specific categories}.
- The 'sentiment polarity' refers to the degree of positivity, negativity or
    neutrality expressed in the opinion towards a particular aspect or feature of a
    product or service, and the available polarities inlcudes: 'positive', 'negative
    ' and 'neutral'.

Recognize all sentiment elements with their corresponding aspect terms, aspect
    categories, opinion terms and sentiment polarity in the following text with the
    format of [('aspect term', 'aspect category', 'sentiment polarity', 'opinion
    term'), ...]:
```

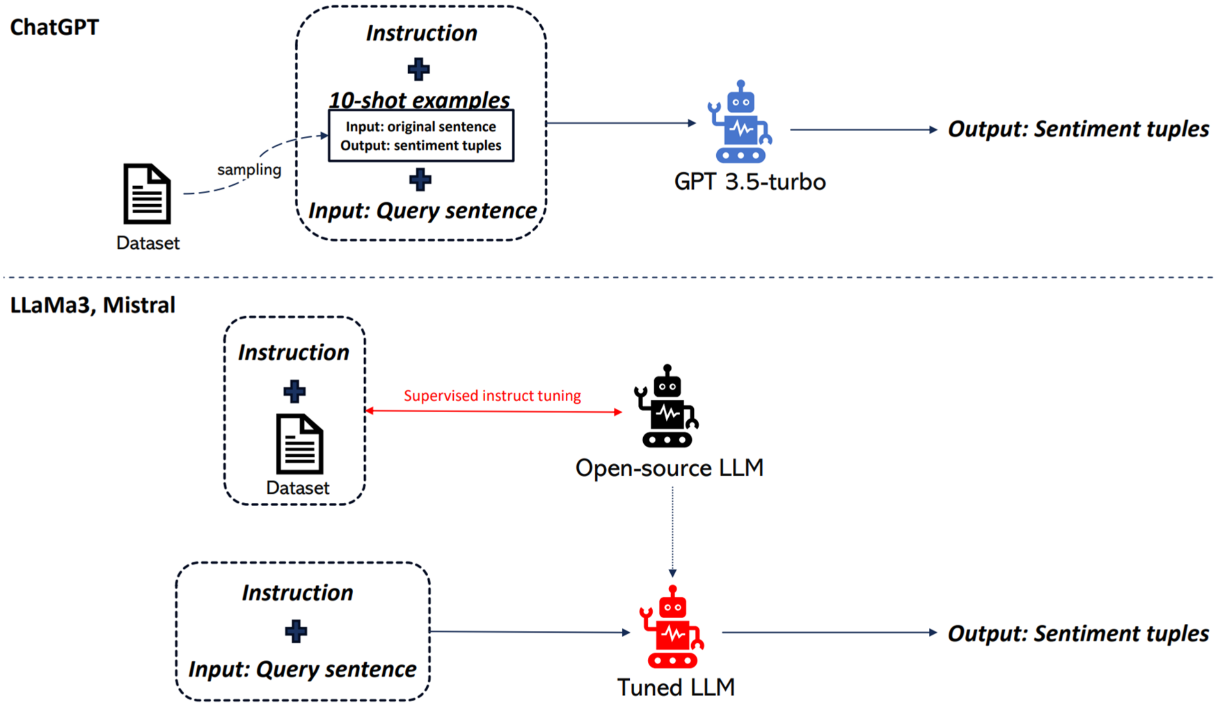


Figure 9: Instruction format for two LLM frameworks. We utilize in-context learning for GPT-3.5-turbo inference, and instruction-tuning for LLaMa-3.1 and Mistral inference respectively.