Error Comparison Optimization for Large Language Models on Aspect-Based Sentiment Analysis

Qianlong Wang[1], Keyang Ding[1], Hengxin Gao[1], Hui Wang[2], Ruifeng Xu[1,2,3]*
[1]Harbin Institute of Technology, Shenzhen, China
[2]Peng Cheng Laboratory, China
[3]Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies
qlwang15@outlook.com, keyang.ding@stu.hit.edu.cn, gaohengxin12@foxmail.com
wangh06@pcl.ac.cn, xuruifeng@hit.edu.cn

Abstract

Supervised fine-tuning (SFT) has enabled large language models (LLMs) to exhibit promising performance on various tasks. However, this fine-tuning process only compares current predictions and labels on each sample, yet fails to perceive and understand its error outputs from different degrees, which may potentially produce a large percentage of serious errors. This poses a problem for aspect-based sentiment analysis (ABSA), in that these serious errors bring a greater negative impact than slight ones. Humans tend to compare mistakes to understand the varying degrees of mistakes, thus avoiding major bad decisions. Inspired by this, we propose a simple yet effective framework, which could understand the degree of different errors by learning from comparative error pairs. It utilizes the SFT model to yield multiple outputs on each sample and selects slight and severe errors based on the acceptable scores. Together with the labels, we construct two comparative error pairs and exploit their calibration losses to optimize parameters. We conduct comprehensive experiments on ABSA datasets to demonstrate the effectiveness of our framework over baselines.

1 Introduction

Aspect-based sentiment analysis (ABSA) aims to extract aspects mentioned in reviews and identify their corresponding sentiment polarities (Pontiki et al., 2014). For example, given a review "*The Mountain Lion OS is not hard to figure out if you are familiar with Microsoft Windows.*", ABSA needs to extract two aspects "*Mountain Lion OS*" and "*Microsoft Windows*", and predict their sentiments as *positive* and *neutral*, respectively.

With the advent of large language models (LLMs) (Touvron et al., 2023; Meta, 2024), researchers have increasingly converged on the use of LLMs for solving ABSA. Based on the different paradigms, existing solutions can be broadly classified into two categories. One is the *in-context learning (ICL) paradigms*-based solutions (Wu et al., 2023; Xu et al., 2024b), which generate a prediction for each test sample by conditioning on demonstrations, without any parameter updates (Dong et al., 2022). This paradigm limits LLMs, which cannot fully learn the specific task knowledge from the given demonstrations (Peng et al., 2023). The other category is the *supervised fine-tuning (SFT) paradigms*-based solutions, which can enable LLMs to learn the relevant knowledge and adapt to a specific task by fine-tuning parameters (Zhang et al., 2023). Consequently, this paradigm can achieve better results on ABSA. In the SFT paradigm, LLMs are commonly optimized to minimize the difference between the current predicted outputs and ground-truth labels.

ABSA is a task that specifically requires LLMs

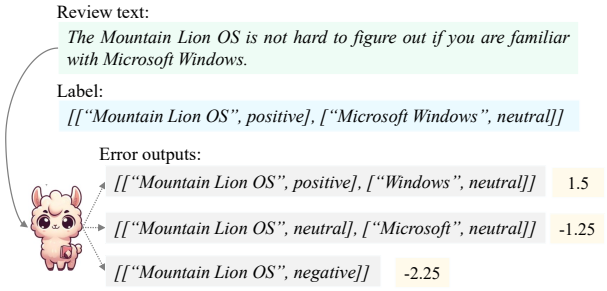

Figure 1: Error output examples. The score (calculated from Algorithm 1) denotes an acceptable degree of error.

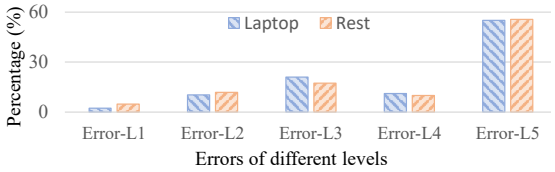

Figure 2: After SFT on LLaMA2-13B, the percentage of different levels of error outputs on both Laptop and Rest test sets. The higher the level, the more serious the error. For error levels, refer to Section Error Level Definition.

*Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18630–18646
July 27 - August 1, 2025

to perceive and understand errors at different degrees during fine-tuning. This is because the error outputs from different degrees in this task have different negative impacts, thus the model should reduce the outputs of severe errors. For example, for the aspect "*Mountain Lion OS*" in Figure 1, misjudging *positive* sentiment as *negative* may be more severe than misjudging *positive* as *neutral*. The reason is that the former has a greater sentiment polarity deviation, *i.e.*, more likely to have a negative impact and cause unpleasant consequences. Besides, for the aspect "*Microsoft Windows*", missing an entire aspect is more unacceptable than missing its boundary. This is because boundary errors, while also affecting the accuracy of the results, could still capture some of the important information in some cases, such as (*e.g., "Windows"*). Thus, LLMs should perceive and understand these near-labeled but problematic errors at different degrees when optimizing. However, SFT is merely a method of minimizing the discrepancy between the current predicted output and label, inherently incapable of understanding errors with different degrees. This not only limits performance improvements but also produces a high percentage of skewed severity errors (see Error-L3/L4/L5 in Figure 2).

Humans often exploit and compare mistakes in the learning process to improve and adapt their strategies. By comparing mistakes, it is possible to be aware of and understand mistakes at different degrees, thus learning how to make better decisions in the future. This process is known as "*error-driven learning*" in cognitive science (Herd et al., 2021). Applying this concept to machine learning (Hoppe et al., 2022), error-driven learning refers to comparing an acceptable and unacceptable output and leveraging their disparity to regulate the model, leading to distinguishing and understanding.

Inspired by this, we propose a simple yet effective framework that learns from comparative error pairs. By comparing them, this framework can perceive and understand the error outputs at different degrees, thus effectively reducing serious errors and approaching the expected results. Specifically, we first fine-tune LLMs with just one epoch for learning task knowledge and preventing over-fitting to obtain label-like outputs. Then, we conduct beam search decoding for each sample in the training set using LLMs to produce different and diverse error outputs. According to the acceptable scores derived from error level definitions, slight and serious errors are selected from all outputs. Together with labels, we construct two sets of comparative data. Finally, we introduce the calibration loss based on these comparative data, which expects LLMs to be able to calibrate severe errors to slight errors and to calibrate slight errors to labels. During the calibration process, LLMs perceive and understand the differences between errors by comparing. As a result, this offers LLMs targeted optimization directions to effectively achieve the desired output, thus improving performance and reducing the output of serious errors.

Our contribution can be summarized as threefolded: (1) To the best of our knowledge, we are the first work to define the level of error outputs on the ABSA task, and first introduce the error-driven learning concept and improve the perception and understanding of LLMs to different errors. (2) We propose a simple yet effective error comparison optimization framework that facilitates understanding the difference between errors by comparing error pairs. (3) We conduct comprehensive experiments on the popular ABSA datasets to show the effectiveness of our framework over baselines.

2 Related Work

Aspect-Based Sentiment Analysis. ABSA aims to determine user's attitudes toward specific aspects in the review. To solve this task, researchers have proposed various joint models based on neural networks and attention mechanisms (Li et al., 2019; Chen and Qian, 2020; Gong et al., 2020; Luo et al., 2020; Liang et al., 2021; Yu et al., 2021). The motivation behind them is to define ABSA as a single sequence labeling task with a unified tagging scheme (called *sequence labeling schema*). By unified tagging, these models could efficiently capture the relationships between aspects and sentiments, leading to accurate sentiment analysis.

In addition to the *sequence labeling schema*, several studies have recently embraced the *language generation schema* to address ABSA (Ling et al., 2022; Gao et al., 2022; Wang et al., 2023, 2024; Scaria et al., 2024; Hosseini-Asl et al., 2022). Under this schema, generative language models learn ABSA as language generation, where the inputs and outputs are represented as a serialized text. Compared with the sequence labeling schema, which tends to use an additional linear layer to produce the tag sequence, this schema does not require deliberate training of task-specific layers. Moreover, thanks to the booming availability of

LLMs, this schema is gradually taking over as the dominant solution.

Large Language Models. With the advent of GPT-3 (Brown et al., 2020), LLMs (Touvron et al., 2023) break into the limelight and draw enormous attention. They typically feature a vast array of model parameters and undergo training on immensely large volumes of raw data. Consequently, LLMs like ChatGPT (OpenAI, 2023) achieve substantial performance improvements on various NLP tasks. The related studies typically follow one of two paradigms: *in-context learning* (ICL) and *supervised fine-tuning* (SFT). In the ICL paradigm, LLMs can directly adapt to new tasks with few examples and instructions. Despite the promising results obtained within a training-free framework, researchers found that many factors have significant and unpredictable impacts on performance, such as example selection (Rubin et al., 2022), example ordering (Liu et al., 2022), and so on. In contrast, the SFT paradigm involves training LLMs on an annotated dataset specific to the task (Dong et al., 2023). This paradigm typically yields superior performance compared to ICL (Hu et al., 2021). This is because by being exposed to many examples in training, LLMs can develop a deeper understanding of the task nuances and improve generalization capabilities.

3 Error Level Definition

We define the error level to distinguish different error outcomes. The rationale behind this definition is grounded in the two critical dimensions of ABSA—aspect boundary and sentiment polarity—both of which are essential to an accurate outcome. Specific definitions are presented below:

- **Error-L1**: *Aspect boundary expansion and sentiment correctness.* Specifically, although the sentiment is correctly predicted, the model incorrectly expands the aspect words to neighboring words, *i.e.*, including additional words beyond the annotated aspect.
- **Error-L2**: *Aspect (left, right, or both) boundary incomplete and sentiment correctness.* This error occurs when the extracted aspect words do not fully cover the full expression of the annotated aspect, usually capturing only a portion of the aspect.
- **Error-L3**: *Aspect boundary correctness and sentiment polarity shift.* Although the aspect is correctly extracted, the predicted sentiment is shifted (and not completely reversed) from the annotated label, such as predicting the *positive* polarity as *neutral* polarity.
- **Error-L4**: *Aspect boundary correctness and sentiment polarity reversal.* This is similar to Error-L3, however, the sentiment polarity is completely reversed (*e.g.*, *positive* is misclassified as *negative*), which is usually considered a more serious error because the reversal can lead to fundamental misunderstandings.
- **Error-L5**: *Under-prediction and over-prediction.* The former indicates that the model fails to recognize an annotated aspect in the review text and its sentiment. The latter denotes that non-existent aspects are identified. This error is the most serious because under-prediction can lead to ignoring important aspects while over-prediction may misinterpret non-aspects.

Type	Example
Correct output	[*"Mountain Lion OS", positive*]
Error-L1	[*"The Mountain Lion OS is", positive*]
Error-L2	[*"OS", positive*]
Error-L3	[*"Mountain Lion OS", neutral*]
Error-L4	[*"Mountain Lion OS", negative*]
Error-L5	[] or [*"Mountain", neutral*]

Table 1: Examples of different error levels. For the sake of clarity and presentation, we take the aspect-sentiment pair [*"Mountain Lion OS", positive*] as an example.

We present some examples in Table 1 to better understand these errors at different levels.

4 Our Framework

4.1 Overview

As shown in Figure 3, our framework consists of three steps: (1) *supervised fine-tuning*, which fine-tunes LLMs to learn task knowledge for yielding many error outputs with similar labels and different levels; (2) *comparative error pair collection*, which conducts beam decoding for each sample and picks out slight and severe errors (along with labels) to form comparative error pairs; (3) *error comparison optimization*, which uses comparative pairs for optimization.

4.2 Supervised Fine-Tuning

In this work, we expect LLMs to learn not only the label information but also the comparisons between different errors when fine-tuning so that it improves

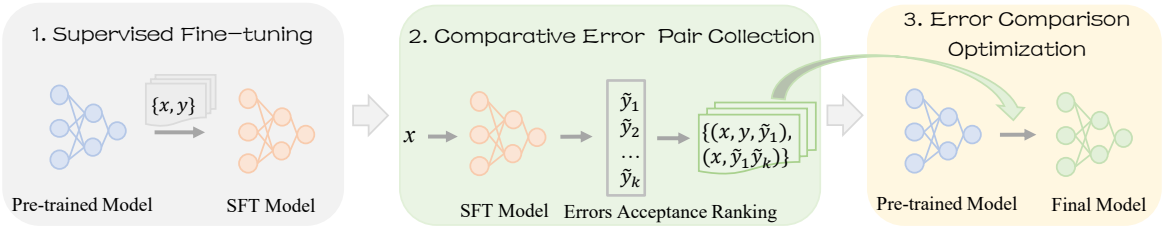

Figure 3: Overview of our framework. It consists of three steps: supervised fine-tuning, comparative error pair collection, and error comparison optimization. Here, y is the label, $\widetilde{y}_1$ is the slight error, and $\widetilde{y}_k$ is the serious error.

performance and reduces the output of serious errors. However, there are only labels on the existing training samples without errors at different degrees. If LLMs are used to infer directly on the training samples, the error outcomes whose distribution may be label-independent are not necessarily related to ABSA. Thus, to obtain error outputs with diverse and different degrees on samples, we first use SFT to fine-tune the LLM. Here, we consider the ABSA task where the review text x is input and the output is expected to conform to the probability distribution of the label y. Formulaically, SFT involves fine-tuning LLMs to minimize the following negative log-likelihood loss:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \mathcal{M}_\theta(y|x)\right] \quad (1)$$

here $\mathcal{M}_\theta$ denotes a pre-trained LLM parameterized by θ. After fine-tuning with merely one epoch to prevent over-fitting, we can obtain the SFT model $\mathcal{M}_{sft}$, which has learned the task knowledge.

4.3 Comparative Error Pair Collection

We can utilize $\mathcal{M}_{sft}$ to yield incorrect outcomes on the training samples as their error outputs. Here, we employ beam search as the decoding strategy and increase the randomness and diversity of outcomes via diversity penalty. Formally, the decoding process for a given review text x from the training set could be expressed as:

$$(y'_1, y'_2, ..., y'_k) = \mathcal{M}_{sft}(x) \quad (2)$$

After filtering the same outcomes as labels, the remaining ones are bound to contain multiple errors since imposing a penalty on similar decoding paths.

A straightforward way to collect pairs is to select two outputs randomly. However, the error degrees of the two outputs selected may be too close to form a meaningful comparison. To alleviate this problem, we compute an acceptable score for each output based on the error level definition. The core idea is that an output with a higher error level or more error types corresponds to a lower acceptable score, indicating that it is more unacceptable. The specific calculation can be referred to Algorithm 1 in Appendix. Based on acceptable scores, we rank these outputs (the ranking is $\widetilde{y}_1, \widetilde{y}_2, ..., \widetilde{y}_k$) and pick the slight error $\widetilde{y}_1$ (with the highest score) and serious error $\widetilde{y}_k$ (with the lowest score) to form comparison pairs. In addition, we construct another comparative pair based on the label y and $\widetilde{y}_1$ to learn more effectively from the slight error. Here, we capitalize on the reliable assumptions that (1) the label could be treated as a *no-error* error, and (2) comparing the slight error with the label may improve the understanding of the error degree.

4.4 Error Comparison Optimization

Conventional SFT only compares current predictions and labels for optimization, yet fails to compare errors at different degrees, which causes the model to treat non-labeled outputs equally. We expect LLMs to be able to perceive error outputs at different degrees during fine-tuning to improve performance and decrease output severity errors during inference.

To enable LLMs to have the capabilities we expected, we devise a calibration objective on the ABSA task based on the comparative error pairs constructed above:

$$\mathcal{L}(o_1, o_2) = max(0, \gamma - \log p(o_1|x) + \log p(o_2|x)) \quad (3)$$

$$\mathcal{L}_{\text{calibration}} = \mathcal{L}(y, \widetilde{y}_1) + \mathcal{L}(\widetilde{y}_1, \widetilde{y}_k) \quad (4)$$

where γ is a margin hyper-parameter; $\log p(o|x)$ denotes the probability of generating the output o based on the input x. This loss encourages LLMs to fulfill the following expectations: calibrate severe error output $\widetilde{y}_k$ to slight one $\widetilde{y}_1$, and calibrate slight error output $\widetilde{y}_1$ to no-error label y. This progressive calibration process achieves the correct output by comparing and refining errors, with the benefit of facilitating the understanding of errors at different degrees. Consequently, it allows LLMs to

		Laptop	Rest	Device	Service
Train	#s	3,043	3,875	2,556	1,490
	#t	2,305	4,312	1,502	1,841
Test	#s	800	1,472	1,279	747
	#t	634	1,738	731	887

Table 2: Dataset statistics. #s and #t are the number of samples and aspect-sentiment tuples, respectively.

acquire more task knowledge than merely comparing predictions and labels. Moreover, LLMs could mitigate severe error outputs by updating internal parameters.

Finally, we sum $\mathcal{L}_{\text{SFT}}$ (Eq. 1) and $\mathcal{L}_{\text{calibration}}$ (Eq. 4) as the final loss in achieving the fine-tuning of LLM. By exploiting SFT loss and calibration loss, our framework implements an effective learning process that fosters corrective optimization of the current prediction and understanding of the comparative differences between error outputs. It is worth noting that although our framework and DPO involve comparing pairs, they have key differences. Please refer to Section Differences with DPO in Appendix for detailed differences.

5 Experiments

5.1 Experimental Settings

Datasets. To evaluate the proposed framework, we conduct experiments on four ABSA datasets. **Laptop** is from SemEval2014 ABSA challenge (Pontiki et al., 2014), containing user reviews from the laptop domain. **Rest** refers to a combination of the restaurant datasets from SemEval ABSA challenge 2014, 2015, and 2016 (Pontiki et al., 2014, 2015, 2016). **Device** is the union set of all the digital device reviews collected by Toprak et al.. **Service** contains reviews from web services, which is introduced by Hu and Liu. Each sample in four datasets contains a review with one or multiple aspect-sentiment tuples. The statistics of datasets are shown in Table 2.

Implementation Details. We choose two LLaMA with different scales (`LLaMA3-8B` and `LLaMA2-13B`)[1] for conducting experiments. To reduce memory consumption and accelerate fine-tuning, we use the LoRA technique (Hu et al., 2021). Adaptations are applied with a rank of 8 and a scaling factor of 32 across all linear layers. We optimize the adapter parameters using AdamW. The epoch, batch size, learning rate, dropout rate, and hyper-parameter γ are set to 3, 4, 3e-5, 0.1, and 1.0, respectively. Besides, the beam number of decoding is 5. Experiments are conducted on GPU A6000. Evaluation metrics are precision (**P**), recall (**R**), and **F1** based on the exact match of the aspect and its sentiment polarity.

[1](1):https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct; (2):https://huggingface.co/meta-llama/Llama-2-13b-chat

5.2 Baseline Methods

To show the effectiveness of our framework, we compare it with several representative baselines: **BERT** (Devlin et al., 2019), **UGF** (Yan et al., 2021), **MRCOOL** (Yang and Zhao, 2022), **MSM** (He et al., 2022), **ICL** (Dong et al., 2022), **SFT** (Ouyang et al., 2022), **PPO** (Schulman et al., 2017), **DPO** (Rafailov et al., 2024), and **CPO** (Xu et al., 2024a). For ICL, we use four settings (**0-shot, 4-shot, 8-shot**, and **16-shot**). Detailed descriptions of these methods can be found in Section Baseline Details in the Appendix.

5.3 Main Results

Table 3 presents our main experiment results. From this table, we have the following observations:

(1) Our framework presents consistent superiority over all peer baselines. Compared to the ICL method using 16-shot examples, improvements ranging from 15.42% to 31.59% have been achieved. In addition, the obvious advantages over the SFT method are observed, with 1.72%∼3.30% gains. It emphasizes the usefulness of implementing LLMs to understand error outputs at different degrees when fine-tuning.

(2) The fine-tuning methods are considerably superior to the ICL methods. This superiority is likely due to the benefit of fine-tuning parameters to acquire task-specific knowledge. Additionally, we observe distinct behaviors among different fine-tuning methods. For example, DPO generally performs better and has more stability than PPO which is sensitive to hyper-parameters. It is worth noting that our framework exceeds PPO and DPO. This observation highlights the effectiveness of constructing comparative error pairs and utilizing progressive calibration optimization.

(3) Among datasets, most methods yield the best results on the Rest and the worst ones on the Device dataset. This is because most aspects (*e.g.*, *food names*) in Rest are expressed more regularly and have clear sentiment expressions while aspects in the Device are expressed in a variety of ways (*e.g.*, *"3x optical zoom"*). Nevertheless, our framework

	Laptop			Rest			Device			Service		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
smaller (Non-LLM) baselines												
BERT	57.51	60.04	58.75	59.59	68.62	63.79	48.16	57.19	52.31	56.92	61.67	59.20
UGF	59.65	62.38	60.98	70.44	61.26	65.53	49.92	58.53	53.88	57.24	62.01	59.52
MRCOOL	61.46	64.29	62.84	71.71	62.04	66.52	50.18	59.54	54.46	57.88	62.65	60.17
MSM	62.62	64.98	63.77	72.44	62.56	67.13	50.37	59.82	54.69	58.66	63.32	60.90
LLaMA2-13B-based												
in-context learning paradigm												
0-shot	26.98	33.27	29.83	42.20	43.60	42.89	20.78	38.58	27.26	42.30	42.67	42.49
4-shot	30.60	35.85	33.03	42.96	45.29	44.10	22.61	39.86	29.00	45.50	46.36	45.93
8-shot	29.72	38.58	33.59	45.16	48.88	46.95	23.41	44.69	30.82	46.20	46.57	46.38
16-shot	27.03	35.64	30.74	45.35	49.13	47.16	23.05	40.49	29.37	47.09	46.92	47.01
supervised fine-tuning paradigm												
SFT	53.13	63.41	57.79	62.93	67.58	64.69	53.13	51.02	52.05	57.24	62.69	59.85
PPO	50.98	63.25	56.42	60.54	68.23	64.16	45.28	59.78	51.53	58.95	61.45	60.18
DPO	51.69	67.41	58.37	62.62	69.50	65.88	50.06	54.81	52.33	57.35	64.14	60.56
CPO	57.02	60.04	58.49	66.89	65.80	66.34	54.43	49.88	52.06	57.83	62.50	60.08
Ours	56.31	64.66	**60.20**	66.85	67.91	**67.38**	60.81	50.05	**54.88**	57.93	67.75	**62.43**
LLaMA3-8B-based												
in-context learning paradigm												
0-shot	35.21	40.16	37.54	45.64	47.78	46.68	27.62	43.33	33.91	42.06	48.76	45.18
4-shot	35.71	42.79	38.95	49.97	53.29	51.58	27.47	43.96	32.91	45.01	50.42	47.57
8-shot	35.47	44.11	39.33	49.48	56.07	52.57	28.44	48.65	35.96	47.29	51.64	49.37
16-shot	32.48	44.11	37.44	44.96	53.08	48.69	25.76	46.74	33.29	47.40	55.59	51.18
supervised fine-tuning paradigm												
SFT	61.24	70.98	65.73	65.39	71.54	68.34	50.05	70.27	58.23	63.93	67.88	65.84
PPO	64.37	69.40	66.78	68.31	70.34	69.31	51.21	65.56	57.49	65.55	66.29	65.91
DPO	65.50	69.50	67.44	68.06	71.43	69.71	50.02	70.93	58.59	65.29	66.40	64.84
CPO	65.80	67.98	66.87	67.20	72.46	69.73	45.47	70.72	55.35	65.66	66.87	66.26
Ours	69.14	68.92	**69.03**	69.25	73.36	**71.24**	54.28	67.67	**60.20**	67.26	67.86	**67.56**

Table 3: Main results (%). The best score across all methods is bolded, and the second-best one is underlined. We also provide results from smaller (non-LLM) models for comparison with the presented LLM-based results. However, they are fundamentally different: the former often involves complex architectures or additional unlabeled data, while the latter typically focuses on modifying training objectives or leveraging pair-based fine-tuning.

	Laptop			Rest			Device			Service		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Ours	69.14	68.92	**69.03**	69.25	73.36	**71.24**	54.28	67.67	**60.20**	67.26	67.86	**67.56**
w/o $\mathcal{L}(\widetilde{y}_1, \widetilde{y}_k)$	64.38	69.65	67.38	67.32	72.76	69.94	52.18	68.63	58.30	64.33	68.49	66.34
w/o $\mathcal{L}(y, \widetilde{y}_1)$	66.16	69.71	67.89	67.98	72.47	70.16	49.84	70.63	58.41	66.40	67.32	66.86
w/o score	65.37	66.92	66.13	66.54	72.33	69.32	49.00	69.99	57.67	63.80	67.47	65.58

Table 4: Ablation study (%) based on LLaMA3-8B.

still achieves remarkable improvements across different datasets. This demonstrates the potential benefits of constructing error pairs and performing comparison optimization in different scenarios.

5.4 Ablation Study

We conduct ablation experiments to verify the performance gain of each component. (1) *w/o* $\mathcal{L}(\widetilde{y}_1, \widetilde{y}_k)$ (in Eq. 4): removing the comparative pairs of slight and serious errors. (2) *w/o* $\mathcal{L}(y, \widetilde{y}_1)$ (in Eq. 4): discards the comparative pairs of slight errors and labels. (3) *w/o* score: randomly select acceptable and serious errors from outputs. Additional details of these ablation variants can be found in Appendix B.4. We can come to the following conclusions from Table 4. First, removing any of these components would have a noticeable negative impact, revealing their individual importance. Second, the pair formed by slight and serious errors is somewhat more important than the ones formed by slight errors and labels. Third, randomly selected error pairs do not lead to noticeable improvements and even have a slight decrease compared with SFT. This indicates that pairs selected based on acceptable scores calculated from the error level definitions can form more meaningful comparisons.

5.5 Generalizability Analysis

Aspect Sentiment Triplet Extraction. To demonstrate the generalizability of the proposed

	ASET-Lap14			ASET-Res14			ASET-Res15			ASET-Res16		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
in-context learning paradigm												
0-shot	9.20	4.05	5.62	11.34	5.53	7.43	5.16	4.74	4.94	15.04	6.61	9.18
4-shot	40.52	39.77	40.14	52.11	50.90	51.50	43.62	46.59	45.06	52.38	57.58	54.86
8-shot	41.32	40.33	40.82	53.58	52.01	52.91	47.14	51.13	49.06	55.88	62.84	59.15
16-shot	43.01	41.98	42.49	56.50	56.33	56.42	48.31	53.19	50.63	53.75	59.92	56.66
supervised fine-tuning paradigm												
SFT	56.83	58.19	57.50	67.24	71.63	69.37	58.00	64.18	60.39	63.75	73.09	68.11
DPO	58.31	58.74	58.53	67.49	70.82	69.12	59.96	63.91	61.87	64.66	73.34	68.73
CPO	56.38	56.90	56.64	67.29	72.03	69.58	59.48	65.96	62.60	65.35	74.12	69.46
Ours	59.22	61.19	**60.19**	69.03	73.11	**71.01**	60.67	67.56	**63.92**	66.63	76.01	**71.00**

Table 5: Generalizability results (%) on Aspect Sentiment Triplet Extraction task on LLaMA3-8B. The best score is bolded, and the second-best one is underlined. Refer to Table 11 in the Appendix for the results on LLaMA2-13B.

	ACOS-Lap			ACOS-Rest			ASQP-Rest15			ASQP-Rest16		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
in-context learning paradigm												
0-shot	0.00	0.00	0.00	0.10	0.10	0.10	0.67	1.00	0.81	0.27	0.37	0.31
4-shot	22.13	20.75	21.42	35.02	32.42	33.67	30.38	28.93	29.63	37.06	34.79	35.89
8-shot	24.57	23.51	24.03	36.21	35.69	35.95	31.08	31.32	31.20	38.23	40.05	39.11
16-shot	24.82	24.37	24.95	40.24	39.41	39.82	31.81	32.45	32.12	35.80	37.41	36.59
supervised fine-tuning paradigm												
SFT	44.58	45.04	44.81	60.50	61.86	61.17	48.88	52.57	50.66	54.17	59.70	56.80
DPO	44.25	44.44	44.34	58.77	59.84	59.30	48.41	49.81	49.10	54.42	58.44	56.36
CPO	45.21	45.21	45.21	61.78	61.91	61.85	50.36	53.46	51.86	55.11	59.32	57.14
Ours	46.51	46.70	**46.60**	62.83	62.69	**62.76**	51.44	54.52	**52.93**	56.45	60.67	**58.48**

Table 6: Generalizability results (%) on Aspect Sentiment Quad Prediction task when using LLaMA3-8B. The best score is bolded, and the second-best one is underlined. Refer to Table 12 in Appendix for the results on LLaMA2-13B.

framework to other ABSA tasks, we extend it to the Aspect Sentiment Triplet Extraction (ASTE) task (Peng et al., 2020). It aims to extract an aspect term, the corresponding opinion term, and the associated sentiment. Here, we can adapt the error level of the aspect term (*i.e., boundary expansion* and *boundary incomplete*) to the opinion term because both are words or phrases. The experiments are carried out on four ASTE datasets, with the detailed information provided in Appendix B.1. According to the experimental results in Table 5, our framework consistently outperforms all baselines. It suggests that comparing errors at varying degrees is potentially useful when fine-tuning on ASTE. Moreover, it further highlights the generalizability of our framework to other ABSA tasks.

Aspect Sentiment Quad Prediction. To further show the adaptability of the proposed framework to other ABSA tasks, we extend it to a representative and challenging task, Aspect Sentiment Quad Prediction (ASQP) (Zhang et al., 2021). ASQP aims to predict four elements: aspect term, aspect category, opinion term, and sentiment polarity. In addition to applying the aspect term's error level to the opinion term, we also classify the aspect category error as non-matching, since it is a predefined label. The experiments are performed on four ASQP datasets, with dataset details given in Appendix B.2. According to the experimental results in Table 6, our framework performs better than baselines. This indirectly highlights the advantage of enabling LLMs to understand errors during fine-tuning, even for complex and challenging ABSA tasks. Moreover, it further demonstrates the extensibility of the idea of error comparison optimization to other ABSA tasks, including more complex quadruplet extraction.

Souce	en	zh	en	de	en	cs
Target	zh	en	de	en	cs	en
ICL	8.26	11.95	13.03	20.14	11.87	25.04
SFT	16.09	20.32	21.05	25.33	20.09	35.35
DPO	18.03	20.23	20.81	24.90	20.23	35.65
Ours	**20.92**	**21.73**	**22.94**	**26.54**	**21.48**	**37.57**

Table 7: Generalizability results (BLEU score %) on Machine Translation task based on LLaMA3-8B. Here, the ICL method exploits the 16-shot in-context example.

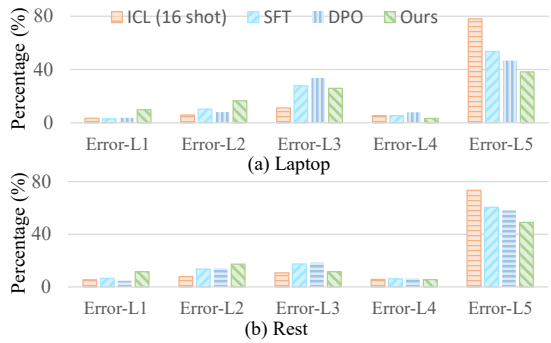

Figure 4: The proportion of errors at different levels outputted by different methods (based on LLaMA3-8B).

	Laptop		Rest	
	2-option	3-option	2-option	3-option
ICL	47.87	35.10	42.89	35.22
SFT	65.01	53.97	69.01	60.05
DPO	68.40	54.77	68.02	61.95
Ours	**74.93**	**59.11**	**75.69**	**67.56**

Table 8: Accuracy (%) of understanding error degrees. Here, the ICL exploits the 16-shot in-context example.

Machine Translations. To confirm the generalizability of our framework to non-ABSA tasks, we apply it to the Machine Translation task. We conduct three language pairs, considering both from English (en) and to English directions: Chinese (zh), German (de), and Czech (cs). Here, we collect human-written datasets from WMT'17 to WMT'20 as the parallel training data and use dataset WMT'22 as the test data. Considering the cost of fine-tuning, we use only 3,000 parallel data for each direction. *To clarify, we do not explicitly define error levels in translation. For simplicity, we directly calculate acceptable scores based on the BLEU algorithm.* Among all the decoded outputs, the highest score corresponds to slight errors, while the lowest score corresponds to severe errors. Table 7 shows experimental results. We can observe that: (1) The results in xx→en are higher than those in en→xx, which may be attributed to a large amount of English pre-training data. (2) Our framework achieves superior results, which shows the effectiveness of the error comparison optimization concept for machine translation, further confirming its generalizability.

5.6 Discussion

Percentage of Different Levels of Error Outputs. Comparing error pairs enables LLMs to understand errors at different degrees, thus decreasing the output of serious errors. To test this statement, we present the percentage of different levels of errors outputted by different methods in Figure 4. We can see that: (1) Due to the inability to learn nuanced task knowledge, the ICL method frequently occurs with under-prediction and over-prediction (*i.e.*, Error-L5). (2) Compared to the ICL method, fine-tuning methods remarkably reduce this proportion on Error-L5, yet increase the sentiment polarity shift error (*i.e.*, Error-L3). We conjecture that fine-tuning fails to equip LLMs to distinguish between *neutral* and *positive* (or *negative*) sentiments effectively. (3) Compared to SFT and DPO, our framework notably reduces the proportion of serious errors, especially in Error-L3 and Error-L5, despite a slight increase in acceptable errors.

Analysis on the Understanding Error Degrees. To further examine the superiority of the proposed framework in understanding and distinguishing error outputs at different degrees, we here design a *puzzle-like experiment*. Specifically, for each test sample, we manually construct different error outputs based on the label and error level definitions. We randomly select two or three error outputs as candidate options, then shuffle their order and ask the methods (based on LLaMA3-8B) to identify the acceptable one. Here, the error output constructed based on the low level is treated as the answer. The experimental results are shown in Table 8. Based on the results, we can draw the following conclusions: (1) Among all methods, the ICL method has the worst accuracy, approaching the results of random guessing. This suggests that the ICL paradigm fails to realize the goal of perceiving and understanding different error outcomes. (2) Our framework achieves remarkable superiority, which shows that error comparison optimization enables LLMs to differentiate errors with different degrees.

6 Conclusion

In this paper, we expect LLMs to understand errors at different degrees during fine-tuning for mitigating the output of severe errors on ABSA. To this end, we construct comparative error pairs and utilize calibration objects for optimization. Experimental results show that our framework exceeds baselines and achieves the desired performance. Moreover, we verify its effectiveness and generalization through analysis experiments.

Limitations

In this section, we list two limitations to understand this work more comprehensively:

1. *To make fine-tuning and inference more effective, we wrap the text-label pair of the original dataset into input-output instruction via the prompt templates.* Refer to Section Prompt Templates in the Appendix for specific information on templates. In general, a hint-rich and elaborated template should be exploited because it can prompt LLMs with richer relevant external knowledge, thus inducing the generation of desired answers. However, in this work, we did not design some hint-rich and elaborated templates for tasks involved in the experiment primarily due to two reasons: (1) the manual design process is time-consuming and cumbersome, and (2) it is not the main focus of this research.

2. *The motivation of our framework is to use comparative error pairs for optimizing to allow the model to perceive error outputs at different degrees.* In this work, we exploit a rule-based algorithm to compute an acceptable score for each output, thereby identifying the slight error (with the highest score) and serious error (with the lowest score) to form comparison pairs. Although our framework is generalizable, which can be applied to other ABSA tasks (*e.g.*, Aspect Sentiment Triplet Extraction and Aspect Sentiment Quad Prediction) and non-ABSA tasks (*e.g.*, Machine Translation), this rule-based algorithm needs to be simply modified or even replaced when generalization is performed. For example, when our framework is applied to machine translation tasks, this algorithm needs to be replaced with the BLEU algorithm to compute acceptable scores.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China 62176076, Natural Science Foundation of Guang Dong 2023A1515012922, the Shenzhen Foundational Research Funding JCYJ20220818102415032, the Major Key Project of PCL2023A09, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies 2022B1212010005 and CIPSC-SMP-ZHIPU Large Model Cross-Disciplinary Fund ZPCG20241119405.

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. In *Advances in NeurIPS*, volume 33, pages 1877–1901.

Hongjie Cai, Rui Xia, and Jianfei Yu. 2021. Aspect-category-opinion-sentiment quadruple extraction with implicit aspects and opinions. In *Proceedings of ACL*, pages 340–350.

Zhuang Chen and Tieyun Qian. 2020. Relation-aware collaborative learning for unified aspect-based sentiment analysis. In *Proceedings of ACL*, pages 3685–3694.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Tianhao Gao, Jun Fang, Hanyu Liu, Zhiyuan Liu, Chao Liu, Pengzhang Liu, Yongjun Bao, and Weipeng Yan. 2022. Lego-absa: A prompt-based task assemblable unified generative framework for multi-task aspect-based sentiment analysis. In *Proceedings of COLING*, pages 7002–7012.

Chenggong Gong, Jianfei Yu, and Rui Xia. 2020. Unified feature and instance-based domain adaptation for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 7035–7045.

Kai He, Rui Mao, Tieliang Gong, Chen Li, and Erik Cambria. 2022. Meta-based self-training and re-weighting for aspect-based sentiment analysis. *IEEE Transactions on Affective Computing*, 14(3):1731–1742.

Seth Herd, Kai Krueger, Ananta Nair, Jessica Mollick, and Randall O'Reilly. 2021. Neural mechanisms of human decision-making. *Cognitive, Affective, & Behavioral Neuroscience*, 21(1):35–57.

Dorothée B Hoppe, Petra Hendriks, Michael Ramscar, and Jacolien van Rij. 2022. An exploration of error-driven learning in simple two-layer networks from a discriminative learning perspective. *Behavior Research Methods*, 54(5):2221–2251.

Ehsan Hosseini-Asl, Wenhao Liu, and Caiming Xiong. 2022. A generative language model for few-shot aspect-based sentiment analysis. In *Findings of NAACL*, pages 770–787.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *ICLR*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD*, pages 168–177.

Zheng Li, Xin Li, Ying Wei, Lidong Bing, Yu Zhang, and Qiang Yang. 2019. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning. In *Proceedings of EMNLP*, pages 4590–4600.

Yunlong Liang, Fandong Meng, Jinchao Zhang, Yufeng Chen, Jinan Xu, and Jie Zhou. 2021. An iterative multi-knowledge transfer network for aspect-based sentiment analysis. In *Findings of EMNLP*, pages 1768–1780.

Yan Ling, Jianfei Yu, and Rui Xia. 2022. Vision-language pre-training for multimodal aspect-based sentiment analysis. In *Proceedings of ACL*, pages 2149–2159.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *arXiv preprint arXiv:2101.06804*, pages 100–114.

Huaishao Luo, Lei Ji, Tianrui Li, Daxin Jiang, and Nan Duan. 2020. Grace: Gradient harmonized and cascaded labeling for aspect-based sentiment analysis. In *Findings of EMNLP*, pages 54–64.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

OpenAI. 2023. Introducing chatgpt plus. https://www.openai.com/blog/introducing-chatgpt-plus. Archived from the original on March 20, 2023. Retrieved March 20, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in NeurIPS*, 35:27730–27744.

Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2020. Knowing what, how and why: A near complete solution for aspect-based sentiment analysis. In *Proceedings of AAAI*, volume 34, pages 8600–8607.

Hao Peng, Xiaozhi Wang, Jianhui Chen, Weikai Li, Yunjia Qi, Zimu Wang, Zhili Wu, Kaisheng Zeng, Bin Xu, Lei Hou, et al. 2023. When does in-context learning fall short and why? a study on specification-heavy tasks. *arXiv preprint arXiv:2311.08993*.

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

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammed Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of SemEval*, pages 19–30.

Maria Pontiki, H Papageorgiou, I Androutsopoulos, D Galanis, M Pontiki, and S Manandhar. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of SemEval*, pages 486–495.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. In *Advances in NeurIPS*, volume 36, pages 53728–53741.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of NAACL*, pages 2655–2671.

Kevin Scaria, Himanshu Gupta, Siddharth Goyal, Saurabh Sawant, Swaroop Mishra, and Chitta Baral. 2024. Instructabsa: Instruction learning for aspect based sentiment analysis. In *Proceedings of NAACL*, pages 720–736.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of ACL*, pages 575–584.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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 AAAI*, 05, pages 9122–9129.

Qianlong Wang, Hongling Xu, Keyang Ding, Bin Liang, and Ruifeng Xu. 2024. In-context example retrieval from multi-perspectives for few-shot aspect-based sentiment analysis. In *Proceedings of LREC-COLING*, pages 8975–8985.

Zengzhi Wang, Qiming Xie, and Rui Xia. 2023. A simple yet effective framework for few-shot aspect-based sentiment analysis. In *Proceedings of SIGIR*, pages 1765–1770.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of ACL*, pages 1423–1436.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024a. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. In *ICML*.

Hongling Xu, Qianlong Wang, Yice Zhang, Min Yang, Xi Zeng, Bing Qin, and Ruifeng Xu. 2024b. Improving in-context learning with prediction feedback for sentiment analysis. *arXiv preprint arXiv:2406.02911*.

Lu Xu, Hao Li, Wei Lu, and Lidong Bing. 2020. Position-aware tagging for aspect sentiment triplet extraction. In *Proceedings of EMNLP*, pages 2339–2349.

Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021. A unified generative framework for aspect-based sentiment analysis. In *Proceedings of ACL*, pages 2416–2429.

Yifei Yang and Hai Zhao. 2022. Aspect-based sentiment analysis as machine reading comprehension. In *Proceedings of COLING*, pages 2461–2471.

Guoxin Yu, Jiwei Li, Ling Luo, Yuxian Meng, Xiang Ao, and Qing He. 2021. Self question-answering: Aspect-based sentiment analysis by role flipped machine reading comprehension. In *Findings of EMNLP*, pages 1331–1342.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Wenxuan Zhang, Yang Deng, Xin Li, Yifei Yuan, Lidong Bing, and Wai Lam. 2021. Aspect sentiment quad prediction as paraphrase generation. In *Proceedings of EMNLP*, pages 9209–9219.

Appendix

A Framework Details

Algorithm 1 Calculation of acceptable scores (*pseudo-code*)

Input: an incorrect output y'; the label y;
Output: an acceptable score s for y' (the larger the score, the more acceptable the output);
Converting list y' and y into set;
$s = |y' \cap y|$; *# initializing acceptance score*
for e' in y' **do** *# e' is an aspect-sentiment pair in output*
    if $e' \in$ Error-L1 **then** *# aspect boundary expansion*
        $s = s - 0.25$
    else if $e' \in$ Error-L2 **then** *# aspect boundary incomplete*
        $s = s - 0.5$
    else if $e' \in$ Error-L3 **then** *# sentiment polarity shift*
        $s = s - 0.75$
    else if $e' \in$ Error-L4 **then** *# sentiment polarity reversal*
        $s = s - 1$
    else if $e' \in$ Error-L5 **then** *# over-prediction*
        $s = s - 1.25$
    end if
end for
for e in y **do** *# e is an aspect-sentiment pair in label*
    if e is *under-prediction* **then**
        $s = s - 1.25$
    end if
end for

A.1 Algorithm Description

The algorithm aims to calculate an "*acceptable score*" for each predicted output y' by comparing it with the ground-truth label y. The core idea is that an output with a higher error level or more error types should have a lower acceptable score, indicating that it is less acceptable. The error levels are classified into five types (Error-L1 to Error-L5), each with an associated penalty. The penalty is subtracted from the initial score (which is based on the overlap between y' and y) to reflect the severity of errors. After calculating acceptable scores, they are then used as a ranking metric. The slightest error (with the highest score) and the most serious error (with the lowest score) are identified for further comparison.

Here, the penalty is intended to reflect the severity of the errors, and the difference between error levels is primarily qualitative rather than quantitative. The reason behind choosing 0.25 as the initial penalty value stems from empirical observations during experimentation. The initial acceptable scores are typically 1 or 2, and subtracting 0.25 (or a multiple thereof) from the score when an error is detected helps ensure that the penalty is significant enough to differentiate errors at different levels. For instance, when encountering the most serious error (Error-L5), the score would be reduced by 1.25, which reasonably reflects the severity of the error. This is found to be effective in distinguishing between different levels of error severity. *It is worth noting that other values could also be chosen (such as 0.5, 1, etc.), but as long as the penalty increases incrementally across error levels (with higher levels leading to a larger reduction), the final ranking of outputs will not change.*

A.2 Supplementary Details of Calibration Loss

In this work, we construct two comparative error pairs—the serious error versus the slight error, and the slight error versus the label. A finer-grained progressive correction approach, such as transforming errors step-by-step across all levels, could potentially lead to more targeted optimization. However, it may have a negative impact on computational efficiency. Performing each error-level transition for each sample during backpropagation would significantly increase computational overhead.

A.3 Differences with DPO

In this work, we expect the proposed framework to achieve perception and understanding of error outputs by comparing pairs. Although the way of comparing pairs in our framework bears a resemblance to direct preference optimization (DPO) (Rafailov et al., 2024), they are fundamentally different. The detail differences are:

1. *Motivation different.* DPO directly compares a pair of preference data. By contrast, our framework is based on the error learning-driven idea, which learns progressively from error comparison by comparing serious errors with slight errors, and slight errors with no-error labels.
2. *Dataset requirement.* DPO necessitates a preference dataset. In contrast, our framework only requires conventional fine-tuning data to obtain the SFT model that produces the comparison error outcomes.
3. *Loss Optimization*. The loss of DPO is designed to minimize the discrepancy between the optimal policy and reference policy, requiring more resources. In contrast, our loss aims to compare error pairs and calibrate them,

which does not involve an additional reference policy.

B Experimental Details

B.1 Aspect Sentiment Triplet Extraction Dataset

The Aspect Sentiment Triplet Extraction (ASTE) datasets originally include four public datasets: Lap14 (Pontiki et al., 2014), Res14 (Pontiki et al., 2014), Res15 (Pontiki et al., 2015), and Res16 (Pontiki et al., 2016). Peng et al. introduced the ASTE-Data-v1 by annotating three key elements for each instance: aspect terms, opinion terms, and sentiment polarity, thereby enabling the extraction of aspect-opinion-sentiment triplets. However, (Xu et al., 2020) pointed out that not all triplets in ASTE-Data-v1 were fully annotated, leading to incomplete triplet coverage. To address this issue, they refined the dataset and released the updated version, ASTE-Data-v2, which provides more comprehensive and accurate triplet annotations.

In this paper, we compare the proposed framework with existing baselines on the ASTE-Data-v2, evaluating its performance and generalization capabilities. The detailed statistics of datasets are provided in Table 9.

		Lap14	Rest14	Rest15	Rest16
Train	#S	906	1,266	605	857
	#A	1,280	2,051	862	1,198
	#O	1,254	2,061	935	1,300
	#T	1,460	2,388	1,013	1,394
Dev	#S	219	310	148	210
	#A	295	500	213	296
	#O	302	497	236	319
	#T	346	577	249	339
Test	#S	328	492	322	326
	#A	463	848	432	452
	#O	466	844	460	474
	#T	543	994	485	514

Table 9: Daraser statistics of ASET-Data-v2 (Xu et al., 2020). #S, #A, #O, and #T represent the number of sentences, aspect terms, opinion terms, and triplets, respectively.

B.2 Aspect Sentiment Quad Prediction Dataset

The Aspect Sentiment Quad Prediction (ASQP) Datasets consist of four datasets: ACOS-Lap and ACOS-Rest (Cai et al., 2021), ASQP-Rest15 and ASQP-Rest16 (Zhang et al., 2021).

Among these, ACOS-Lap and ACOS-Rest were constructed to extract all quadruples of aspect-category-opinion-sentiment from review sentences. Here, ACOS-Lap is a new and much larger dataset focused on laptop reviews, surpassing the size of the original SemEval Laptop dataset. ACOS-Rest is an extension of the SemEval Restaurant dataset. ASQP-Rest15 and ASQP-Rest16 were developed by Zhang et al., based on the SemEval Shared Challenges. The annotations for the opinion terms and aspect categories are derived from previous work (Peng et al., 2020) and (Wan et al., 2020), respectively. Additionally, sentences without explicit aspect terms were included.

We use these datasets to evaluate the generalization and effectiveness of the proposed methods on the challenging quadruplet extraction task. The detailed statistics of these datasets are provided in Table 10.

	Train		Dev		Test	
	#S	#Q	#S	#Q	#S	#Q
ACOS-Lap	2,934	4,172	326	440	816	1,161
ACOS-Rest	1,530	2,484	171	261	583	916
ASQP-Rest15	834	1,354	209	347	537	795
ASQP-Rest16	1,264	1,989	316	507	544	799

Table 10: Daraser statistics of four ASQP datasets (Cai et al., 2021; Zhang et al., 2021). #S and #Q represent the number of sentences and quads, respectively.

B.3 Baseline Details

To make a comprehensive study, we select some representative baselines to demonstrate the effectiveness of the proposed framework. Below are brief introductions to each baseline.

Smaller (Non-LLM) baseline models are as follows:

- **BERT** (Devlin et al., 2019): A widely used baseline model that employs BERT-base-uncased as an encoder and utilizes a linear layer for classification, where label information combines aspect boundaries and sentiment categories.
- **UGF** (Unified Generative Framework) (Yan et al., 2021) redefines the task target as a sequence blending pointer indexes with sentiment class indexes, converting ABSA into a unified generative formulation. This approach leverages the sequence-to-sequence BART model in an end-to-end framework.
- **MRCOOL** (MRC-PrOmpt mOdeL framework) (Yang and Zhao, 2022) elicits mul-

	ASET-Lap14			ASET-Res14			ASET-Res15			ASET-Res16		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
in-context learning paradigm												
0-shot	7.69	0.92	1.64	9.87	0.80	1.48	21.27	2.06	3.75	4.34	0.38	0.71
4-shot	33.25	26.88	29.73	47.52	34.80	40.18	43.03	35.67	39.00	52.47	47.47	49.86
8-shot	32.72	29.65	31.11	51.85	43.66	47.40	41.36	42.47	41.91	49.88	43.38	46.40
16-shot	40.12	34.80	37.27	50.51	44.46	47.29	41.44	41.44	41.44	51.25	47.85	49.49
supervised fine-tuning paradigm												
SFT	52.46	51.95	52.20	67.11	65.69	66.39	56.58	60.20	58.34	59.95	67.06	63.31
DPO	50.82	51.01	50.91	65.90	66.70	66.30	56.47	60.20	58.28	60.52	67.70	63.91
CPO	52.26	53.03	52.65	65.36	65.89	65.63	57.08	59.79	58.40	60.20	68.28	63.99
Ours	54.03	54.28	**54.15**	68.03	67.46	**67.74**	60.20	59.72	**59.96**	62.74	68.27	**65.38**

Table 11: Generalizability results (%) on Aspect Sentiment Triplet Extraction task when using `LLaMA2-13B`. The best score across all methods is bolded, and the second-best one is underlined.

	ACOS-Lap			ACOS-Rest			ASQP-Rest15			ASQP-Rest16		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
in-context learning paradigm												
0-shot	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.52	0.12	0.24
4-shot	18.03	12.83	14.99	33.59	23.03	27.33	26.01	18.49	21.61	28.90	21.52	24.67
8-shot	20.78	15.41	17.70	33.04	24.78	28.32	27.78	22.26	24.72	32.50	27.90	30.03
16-shot	22.58	17.91	19.98	32.49	25.43	28.53	24.96	20.00	22.20	31.77	26.28	28.76
supervised fine-tuning paradigm												
SFT	42.81	42.11	42.46	55.95	56.44	56.19	41.76	44.02	42.86	52.06	56.82	54.33
DPO	42.09	40.82	41.45	57.04	56.55	56.79	40.02	40.88	40.44	51.60	54.44	53.98
CPO	44.59	41.60	43.04	59.09	57.09	58.07	42.96	43.01	42.99	54.28	55.44	54.86
Ours	46.08	42.72	**44.33**	61.12	57.89	**59.46**	44.51	44.68	**44.58**	55.23	56.12	**55.67**

Table 12: Generalizability results (%) on Aspect Sentiment Quad Prediction task when using `LLaMA2-13B`. The best score across all methods is bolded, and the second-best one is underlined.

tiple sentimental aspects using a machine-read comprehension model. It classifies corresponding sentiment polarities through a prompt-learning approach, employing the BERT-base-uncased model with standard parameters.

- **MSM** (Meta-based Self-training Method) (He et al., 2022) employs a meta-weighter in conjunction with self-training. A teacher model generates in-domain pseudo-labels on unlabeled data, which a student model learns from, while the meta-weighter assigns subtask-specific weights to instances, balancing class labels and reducing noise during training. MSM utilizes BERT-large with 1024 hidden dimensions.

LLM-based baseline models are as follows:

- **ICL** (In-Context Learning) (Dong et al., 2022) would retrieve a few input-output pairs as demonstrations from the training set and obtain predictions from LLMs without training. We conducted experiments using four different shot settings (**0-shot, 4-shot, 8-shot**, and **16-shot**). Here, we use BM25 as a retriever owing to its simplicity and effectiveness (Wang et al., 2024).
- **SFT** (Supervised Fine-Tuning) (Ouyang et al., 2022) involves further training a pre-trained LLM on a labeled dataset with specific tasks by minimizing the loss between its predictions and the annotated labels.
- **PPO** (Proximal Policy Optimization) (Schulman et al., 2017) is a policy gradient method for reinforcement learning. For simplicity, we first calculate the acceptable scores of outputs according to the error level definition and then normalize them as the reward.
- **DPO** (Direct Preference Optimization) (Rafailov et al., 2024) uses a supervised approach to fine-tune LLMs based on preference data. Here, we construct preference data based on slight and severe errors for each sample.
- **CPO** (Contrastive Preference Optimization) (Xu et al., 2024a) is a variant of DPO, which omits the policy model and uses both model-

generated and reference data to guide LLM.

B.4 Supplementary Details of Ablation Variants

In this paper, we evaluate several stripped-down variants of our framework to assess the performance gains brought by each component. Below is a detailed explanation of each ablation variant:

- *w/o* $\mathcal{L}(\widetilde{y}_1, \widetilde{y}_k)$: This variant removes the comparative pairs of slight and severe errors. We introduce this variant to validate the effectiveness of the calibration loss between severe errors and slight errors. This helps to demonstrate the potential benefits of error comparison optimization, particularly in guiding the model to mitigate more severe errors.
- *w/o* $\mathcal{L}(y, \widetilde{y}_1)$: This variant discards the comparative pairs between slight errors and the no-error labels. We introduce this variant to assess the importance of calibrating the slight error output to the no-error label. This helps to make the model lean towards more accurate predictions.
- *w/o* score: In this case, we randomly select two outputs as slight and severe errors and perform calibration between them. Here, the two outputs chosen may be too similar, which makes the comparison less meaningful. More critically, without using an acceptable score to rank these errors, it is possible that the model could misidentify an acceptable error as a severe one or vice versa, leading to counterproductive calibration. The reason for including this ablation variant is to validate that selecting error pairs based on scores derived from error level definitions results in more meaningful comparisons. Using a score ensures that the selected errors have distinct characteristics, making the calibration process more effective and meaningful.

C Supplementary Discussion

C.1 Effect of Beam Number on Performance

The beam number k of decoding is an important hyper-parameter in our framework. Here, we study the effect of varying values $\{0, 3, 5, 7, 10, 12\}$ on performance to explore its sensitivity. Figure 5 depicts the experimental results. We draw the following observation from this figure: (1) Initially, the performance improves as k increases. This implies that larger beams may yield diverse error outputs at different degrees, forming more meaningful comparative error pairs. (2) After k exceeds 5, the performance reaches a peak and then fluctuates. The potential reason for this is that although more outputs are produced, the errors may not be more diverse in terms of degrees.

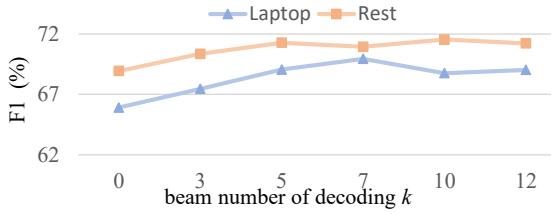

Figure 5: Performance at different beam numbers. When $k = 0$, our framework degrades to an SFT method.

	Case 1	
	The *[Mountain Lion OS]*$_{positive}$ is not hard to figure out if you are familiar with *[Microsoft Windows]*$_{neutral}$.	
ICL:	[[*"OS", positive*](✗), [*"Microsoft Windows", positive*](✗)]	L2&L3
SFT:	[[*"Mountain Lion OS", positive*](✓)]	L5
DPO:	[[*"Mountain Lion OS", positive*](✓)]	L5
Ours:	[[*"Mountain Lion OS", positive*](✓), [*"Microsoft Windows", neutral*](✓)]	/
	Case 2	
	Chatting with *[Acer support]*$_{neutral}$, I was advised the problem was corrupted *[operating system files]*$_{neutral}$.	
ICL:	[[*"operating system files", negative*](✗)]	L3&L5
SFT:	[[*"operating system", negative*](✗)]	L5
DPO:	[[*"operating system", negative*](✗)]	L5
Ours:	[[*"Acer support", neutral*](✓), [*"operating system files", negative*](✗)]	L3

Table 13: Case study. The aspect in the review text is marked with the symbol `[]`, while its corresponding sentiment label is shown in the subscript. ✓and ✗denote the correct and incorrect predictions, respectively. The rightmost column indicates the error level of the incorrect prediction.

C.2 Case Study

To understand the superiority of our framework intuitively, we show the prediction differences between the different methods on two test cases in Table 13. We can observe that: (1) In Case 1, ICL (with 16-shot) performs the worst, with not only a boundary incomplete (Error-L2) but also a sentiment shift (Error-L3) in predictions. Although SFT and DPO correctly predict an aspect-sentiment pair,

another label pair is missing (under-prediction). In contrast, our framework correctly predicts two label pairs, showing that comparing errors at different degrees can improve prediction accuracy. (2) Although the predictions of all methods are not completely correct in Case 2, the output of our framework is the most acceptable (lowest error level). We attribute this to the motivation that understanding error outputs does mitigate the occurrence of severe errors.

D Prompt Templates

Below we present prompts used in this work to understand the details of the experiments better.

Prompt template for the main task (Aspect-Based Sentiment Analysis):

```
Task Description:
Perform Aspect-Based Sentiment Analysis (ABSA) on customer reviews. ABSA involves identifying specific features or attributes of a product or service (aspects) discussed in review text, and analyzing the sentiment expressed towards these aspects.

Instructions:
1. Identify Aspects: Identify all the distinct features or attributes mentioned in the review. Aspects should be extracted as noun phrase spans.
2. Analyze Sentiment: Assess the sentiment for each identified aspect based on the language used in relation to that aspect. Sentiment should be classified as:
  - Positive (POS): Expresses satisfaction or positive emotions.
  - Negative (NEG): Expresses dissatisfaction or negative emotions.
  - Neutral (NEU): Neither explicitly positive nor negative, or the context does not provide enough information for a clear sentiment.
3. Label the Aspects and Sentiments: Format your findings as a list of lists, where each inner list contains an aspect followed by its sentiment polarity code.

Examples for Guidance:
{"input example"}

Task for Completion:
Review: "{input review text}"

Output Format: Please provide your analysis in the following JSON format:
{
    "Output": "[
        ['Aspect1', 'Sentiment1'],
        ...
    ]"
}
```

Prompt template for the generalization analysis on Aspect Sentiment Triplet Extraction:

```
Task Description:
Perform Aspect Sentiment Triplet Extraction (ASTE) on customer reviews. ASTE involves identifying specific attributes of a product (aspects) discussed in the review, extracting any opinion terms used to describe them, and analyzing the sentiment expressed towards these aspects.

Instructions:
1. Identify Aspects: Identify all the distinct features or attributes mentioned in the review. Aspects should be extracted as noun phrase spans. If the aspect is implicit (not explicitly mentioned in the review), output `null` for the aspect.
2. Extract Opinions: Identify the explicit opinion terms associated with each aspect. Opinions should be extracted as noun phrase spans. If the opinion term is implicit (not explicitly mentioned in the review), output `null` for the opinion term.
3. Analyze Sentiment: Assess the sentiment for each identified aspect based on the language used in relation to that aspect. Sentiment should be classified as:
  - Positive (POS): Expresses satisfaction or positive emotions.
  - Negative (NEG): Expresses dissatisfaction or negative emotions.
  - Neutral (NEU): Neither explicitly positive nor negative, or the context does not provide enough information for a clear sentiment.
4. Label the Aspects, Sentiments, and Opinion Terms: Format your findings as a list of lists, where each inner list contains:
  - Aspect
  - Opinion
  - Sentiment polarity code

Examples for Guidance:
{"input example"}

Task for Completion:
Review: "{input review text}"

Output Format: Please provide your analysis in the following JSON format:
{
    "Output": "[
       ['Aspect1', 'Opinion1', 'Sentiment1'],
          ...
    ]"
}
```

Prompt template for the generalization analysis on Aspect Sentiment Quad Prediction:

```
Task Description:
Perform Aspect Sentiment Quad Prediction (ASQP) on customer reviews. ASQP aims to predict all quads (aspect term, opinion term, aspect category, sentiment polarity) for a given review.

Instructions:
1. Identify Aspects: Identify all the distinct features or attributes mentioned in the review. Aspects should be extracted as noun phrase spans. If the aspect is implicit
```

```
(not explicitly mentioned in the review), output `null` for the aspect.
2. Extract Opinion Terms: Identify the specific words or phrases that convey the sentiment towards each aspect. Opinions should be extracted as noun phrase spans. If the opinion term is implicit (not explicitly mentioned in the review), output `null` for the opinion term.
3. Determine Aspect Categories: Map each identified aspect to a predefined category. It consists of an entity label and attribute label, with possible values like:
{"input predefined category labels"}
4. Analyze Sentiment: Assess the sentiment for each aspect based on the language used in relation to it. Sentiment should be classified as:
  - Positive (POS): Expresses satisfaction or positive emotions.
  - Negative (NEG): Expresses dissatisfaction or negative emotions.
  - Neutral (NEU): Neither explicitly positive nor negative, or the context does not provide enough information for a clear sentiment.
5. Label the Quads: Format your findings as a list of quads, where each quad contains:
  - Aspect term
  - Opinion term
  - Aspect category
  - Sentiment polarity

Examples for Guidance:
{"input example"}

Task for Completion:
Review: "{input review text}"

Output Format: Please provide your analysis in the following JSON format:
{
    "Output": "[
            ['Aspect1', 'Opinion1', 'Aspect category1', 'Sentiment1'],
        ...
    ]"
}
```

Prompt template for the generalization analysis on Machine Translations:

```
Machine translation is the automated process of converting text from {"source"} language to {"target"} language using advanced algorithms and linguistic models.
The key to translation is accurately conveying the original meaning, context, and nuances from one language to another language.

Examples for Guidance:
{"input example"}

Task for Completion:
Review the following translation: "{source input}"

Output Format: Please provide your analysis in the following JSON format:
{
    "Output": "Your translation here"
}
```

Prompt template for analysis of error understanding (*Puzzle-like Experiment*):

```
Some examples for understanding the ABSA task:
{"input example"}

Task Description:
Given a review text, you are provided with {"two (or three)"} options, each representing extracted aspects and their associated sentiment analysis results.
If an option is an empty list, it means no aspects are present in the sentence.
Your task is to select the most appropriate option that best represents the aspects and sentiment expressed in the review.

Instructions:
1. Review the Options: Each option is a list of lists, where each inner list contains an aspect followed by its sentiment polarity code (POS for Positive, NEG for Negative, NEU for Neutral).
2. Select the Best Option: Choose the option that most accurately captures the aspects discussed in the review and the sentiment expressed towards those aspects.
3. Output the Result: Provide the selected option as a JSON object.

Review: "{input review text}"
Options:
{"input options"}

Output Format: Please provide your analysis in the following JSON format:
{
    "Output": "{A | B (or | C)}"
}
```