Trigger^3: Refining Query Correction via Adaptive Model Selector

Kepu Zhang,[1] Zhongxiang Sun,[1] Xiao Zhang,[1,*] Xiaoxue Zang,[2] Kai Zheng,[2] Yang Song,[2] Jun Xu[1]

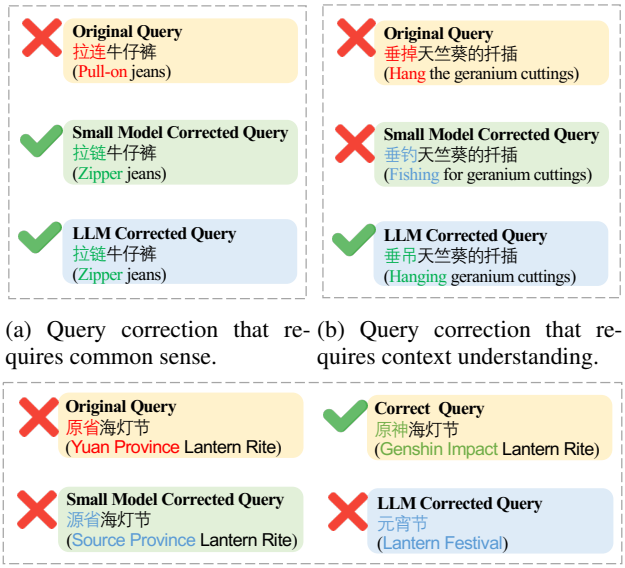
[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2] Kuaishou Technology Co., Ltd.
{kepuzhang, sunzhongxiang, zhangx89}@ruc.edu.cn
{zangxiaoxue,zhengkai,yangsong}@kuaishou.com, junxu@ruc.edu.cn

Abstract

In search scenarios, user experience can be hindered by erroneous queries due to typos, voice errors, or knowledge gaps. Therefore, query correction is crucial for search engines. Current correction models, usually small models trained on specific data, often struggle with queries beyond their training scope or those requiring contextual understanding. While the advent of Large Language Models (LLMs) offers a potential solution, they are still limited by their pre-training data and inference cost, particularly for complex queries, making them not always effective for query correction. To tackle these, we propose Trigger3, a large-small model collaboration framework that integrates the traditional correction model and LLM for query correction, capable of adaptively choosing the appropriate correction method based on the query and the correction results from the traditional correction model and LLM. Trigger3 first employs a correction trigger to filter out correct queries. Incorrect queries are then corrected by the traditional correction model. If this fails, an LLM trigger is activated to call the LLM for correction. Finally, for queries that no model can correct, a fallback trigger decides to return the original query. Extensive experiments demonstrate Trigger3 outperforms correction baselines while maintaining efficiency.

1 Introduction

In online search scenarios, users may input incorrect queries due to insufficient knowledge, voice input, etc., resulting in errors such as typos, missing characters, homophones, and similar shapes (Ye et al. 2023; Pande et al. 2022). If we do not correct the queries and use the original queries for searching, the results may significantly deviate from the user's needs. Therefore, to improve the user's search experience, search engines must implement query correction services that automatically detect and correct errors in queries.



(a) Query correction that requires common sense.

(b) Query correction that requires context understanding.

(c) Query correction that requires specific domain knowledge.

Figure 1: Examples of query correction, where the red characters are the original errors, the blue characters are the results of corrected but incorrect, and the green characters are the correct result. The small model is traditional correction model GECToR and the LLM is Qwen1.5-7B-Chat.

In the field of query correction, the existing mainstream correction models can be divided into Seq2Seq and Seq2Edit methods. The Seq2Seq model (Shao et al. 2024; Xue et al. 2021) treats the correction task as a machine translation task, that is, translating the incorrect query into the correct query; the Seq2Edit model (Zhang et al. 2022) treats the correction task as a sequence labeling task, correcting errors by marking insertions, deletions, etc. In this paper, *we refer to these two types of traditional correction models as small models.* Nowadays, Large Language Models (LLMs) have demonstrated robust semantic comprehension in numerous tasks (Brown et al. 2020; Ouyang et al. 2022), making them a viable option for query correction. When using the small

*Corresponding author: Xiao Zhang (zhangx89@ruc.edu.cn).
Work partially done at Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education. Work done when Kepu Zhang and Zhongxiang Sun were interns at Kuaishou.

model and LLM for query correction, we anticipate the following three observations:

- Some queries are related to grammatical errors, which can be corrected based on common sense, where common sense refers to the knowledge that is easily included in the small model or LLM pre-training data, a capability possessed by both small and large models (Ding et al. 2024). For instance, as shown in Figure 1 (a), a user mistakenly inputs "pull-on" instead of "zipper" in the query due to a grammatical error. "pull-on" is not common in Chinese, while the correct "zipper" is very common, thus both models can correct it. Therefore, *both small models and LLMs are capable of correcting errors in queries that can be addressed with common sense*.
- Some queries necessitate a comprehensive understanding of query context, which may pose challenges for small models. For example, as depicted in Figure 1 (b), a user incorrectly inputs "Hanging geranium cuttings" as "Hang the geranium cuttings". GECToR corrects it to "Fishing for geranium cuttings". The words "fishing", "hanging", and "hang" are all grammatically correct in Chinese with similar pronunciations but vastly different meanings. Therefore, *small models cannot correct errors in queries that require strong contextual semantic understanding, while LLMs can*.
- As user queries may cover various aspects, there are certain queries that even the LLM might struggle to handle. These could be queries related to real-time news or specific domains. For example, As depicted in Figure 1 (c), within the gaming field, a user incorrectly inputs "Genshin Impact Lantern Rite" as "Yuan Province Lantern Rite". The small model corrects it to "Source Province Lantern Rite", while the LLM corrects it to "Lantern Festival". Both the small model and LLM, lacking knowledge in this specific domain, provide incorrect corrections. We observe that the corrected queries by the models might completely deviate from the user's original input. Using these deviated results as the final queries can severely affect the user search experience. Therefore, *neither small models nor LLMs can correct errors in queries related to specific domains or real-time news*.

From these observations, we can learn that neither small models nor LLMs are universally effective in query correction tasks. Moreover, in terms of correction costs, the expenditure for small models is typically less than that for LLMs (Ramírez, Birch, and Titov 2024). Therefore, the crucial issues when relying on small models and LLMs for query correction tasks are: *when to employ either model and which one to choose for query correction, the small model or the LLM?* This is essentially a model selection problem for large-small model collaboration tasks, aimed at improving model performance and efficiency to enhance the trustworthiness (Liu et al. 2023) and controllability (Shen et al. 2024) of LLM-powered systems.

To address the aforementioned issues, in this paper, we propose a novel model selector framework for query correction, named $\mathrm{Trigger}^3$, to adaptively integrate the small model and LLM for query correction. $\mathrm{Trigger}^3$ mainly consists of three parts: Correction Trigger (CT), LLM Trigger (LT) and Fallback Trigger (FT).

For when to employ models for correction: The CT selects incorrect queries for subsequent correction. The FT conducts a review after the correction by both models, returning the original query for those that are difficult for both models to correct. For which model to choose: The LT selects queries that are difficult for the small model to correct but can be corrected by the LLM to the LLM for correction. In cases both models can correct, the small model's corrections are taken as final queries. Through the three modules, we not only leverage the correction capabilities of both models but also consider their limits, leading to enhanced correction performance and efficiency.

To validate the effectiveness and efficiency and of the proposed $\mathrm{Trigger}^3$ framework, we conduct experiments on two query correction datasets, using three small models and two LLMs. The results consistently demonstrate that $\mathrm{Trigger}^3$ achieves optimal performance and high efficiency. We summarize our contributions as follows:

- We propose $\mathrm{Trigger}^3$, a novel large-small model collaboration framework that adaptively completes query correction by considering feedback from both the small model and LLM, which is model-agnostic.
- We explore the combination of the small models and LLMs in the field of query correction, providing solutions for applying LLMs in query correction and how small models and LLMs can better collaborate.
- We conduct extensive experiments on both commercial and public datasets, showing that $\mathrm{Trigger}^3$ achieves superior performance while maintaining high efficiency.

2 Related Work

2.1 Query Correction in Search Engines

With the rise of neural networks, the current query correction models are mainly divided into two types: Seq2Edit and Seq2Seq. Seq2Edit models (Zhang et al. 2022; Awasthi et al. 2019; Liang et al. 2020) treat correction as a sequence tagging problem, completing the correction through editing operations such as insertion and deletion. Seq2Seq models (Shao et al. 2024; Zhang et al. 2021; Zhao and Wang 2020) view the correction task as a translation task, translating the incorrect query into the correct one. They can achieve decent correction performance to a certain extent, but due to insufficient knowledge or weaker semantic understanding, they struggle to handle some queries.

Recently, some work has explored the application of Large Language Models (LLMs) in the correction field. By designing prompts and conducting a comprehensive evaluation of ChatGPT's performance on the correction task through in-context learning, (Fang et al. 2023; Li et al. 2023; Davis et al. 2024; Coyne and Sakaguchi 2023) find that LLMs tend to over-correct, and there is still a significant gap between LLM and small models trained on specific correction datasets. (Fan et al. 2023) confirms that fine-tuning can enhance LLM's ability in text correction.

2.2 Model Selection of Language Models

Model selection has long been a fundamental problem in machine learning (Ding, Tarokh, and Yang 2018; Zhang, Liao, and Liao 2019; Zhang and Liao 2020). Considering the high cost of LLMs, recent work has explored how to balance performance and efficiency. Their methods are mainly divided into two categories. The first category selects small and large models through a routing approach, mainly by predicting the accuracy of the small model's responses (Lu et al. 2023; Ding et al. 2024) to determine the invocation of the large model. The second category adopts a cascading approach to decide whether to invoke the larger model after the execution of the smaller one. (Madaan et al. 2023) uses few-shot learning within the small model to verify its answers. (Yue et al. 2023) judges based on the consistency of multiple answer samples obtained by the small model. In code-driven QA tasks, (Zhang et al. 2023) introduces an automatic code executor to decide based on the generated code execution. Most recently, (Ramírez, Birch, and Titov 2024) makes decisions based on the uncertainty of the small model's output.

Unlike the above methods, we consider the specificity of query correction, which does not necessarily require an answer. Firstly, if the query is already correct, there's no need for correction. Secondly, both small and large models may not always provide accurate corrections. Hence, we designed the CT and FT to address these considerations.

3 Trigger^3: The Proposed Framework

3.1 Task Formalization

In the query correction task, we are given a set of data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{|D|}$, where $|D|$ indicates the total number of data, each of these data samples contains: x_i represents the i-th original query, y_i represents the correct query corresponding to x_i. The goal of the query correction task is to learn the function from the original query to the target query. Here query x_i and y_i may not be the same length.

3.2 General Framework

The large-small model collaboration framework of Trigger^3 is shown in Figure 2. The input is the original query, and after interacting with the adaptive model selector, the small model and LLM, the output is the final corrected query.

The adaptive model selector consists of 1) **Correction Trigger (CT)** f_{CT} that decides whether the original query needs to be corrected, 2) **LLM Trigger (LT)** f_{LT} that analyzes if the LLM is needed for query correction, and 3) **Fallback Trigger (FT)** f_{FT} that checks whether the original query needs to be returned. As shown in Algorithm 1, a query x is corrected following the process below:

The query will first go through the CT (line 3), which will determine whether it needs to be corrected based on its correctness. If the CT determines that the query needs to be corrected, it passes the query to the small model. This model is designed to handle common and simple errors and is more efficient compared to the LLM. We denote it as f_{small}. f_{small} takes original query x as input and outputs its corrected query y_{small}:

$$y_{\mathrm{small}} = f_{\mathrm{small}}(x; \theta_{\mathrm{small}}), \tag{1}$$

Algorithm 1: Process flow of Trigger^3.

```
1  Input: Original query x and Trigger^3's models.
2  Output: Final corrected query y_final.
3  p_CT ← f_CT(x)                                  ▷ Correction Trigger
4  if p_CT = 1 then
5      y_small ← f_small(x)
6      p_LT ← f_LT(x, y_small)                     ▷ LLM Trigger
7      if p_LT = 1 then
8          y_LLM ← f_LLM(x, y_small)
9          y_c = y_LLM
10     else
11         y_c = y_small
12     p_FT ← f_FT(x, y_c)                         ▷ Fallback Trigger
13     if p_FT = 1 then
14         y_final = x
15     else
16         y_final = y_c
17 else
18     y_final = x
```

where θ_{small} is the learnable parameters in small model. After being corrected by the small model, the query corrected by the small model and the original query will go through the LT (line 6) to determine whether the LLM is needed for correction.

If the LT determines that the query cannot be corrected by the small model, but can be corrected by the LLM, the query is passed to the LLM. This model is more powerful and can handle more complex errors, but it is more resource-intensive. We denote it as f_{LLM}. f_{LLM} takes (x, y_{small}) as input and outputs the its corrected query y_{LLM}:

$$y_{\mathrm{LLM}} = f_{\mathrm{LLM}}(x, y_{\mathrm{small}}; \theta_{\mathrm{LLM}}), \tag{2}$$

where θ_{LLM} is parameters in f_{LLM}.

Finally, the FT (line 12) will determine whether to return the original query as the final query output based on the corrected query and the original query. That is, the final corrected query may use the corrections from the small model, the LLM, or it may remain the original query:

$$y_{\mathrm{final}} = x \text{ or } y_{\mathrm{small}} \text{ or } y_{\mathrm{LLM}} \tag{3}$$

3.3 The First Trigger: Correction Trigger

To improve efficiency, we first judge the correctness of the original query. If the query itself is correct, there is no need to use the small model and the LLM for correction.

We use the Correction Trigger (CT) to achieve the above goal. Given the initial query x, CT is a scoring function that indicates the probability of the query being incorrect:

$$p_{\mathrm{CT}} = P(\text{Incorrect}|x) = f_{\mathrm{CT}}(x; \theta_{\mathrm{CT}}), \tag{4}$$

where $P(\text{Incorrect}|x)$ is the probability that the query x is incorrect. If p_{CT} is above a certain threshold, we can conclude that the query is incorrect and correction is needed.

We use the representation of the [CLS] token in BERT (Devlin et al. 2019) to get the score p_{CT}.

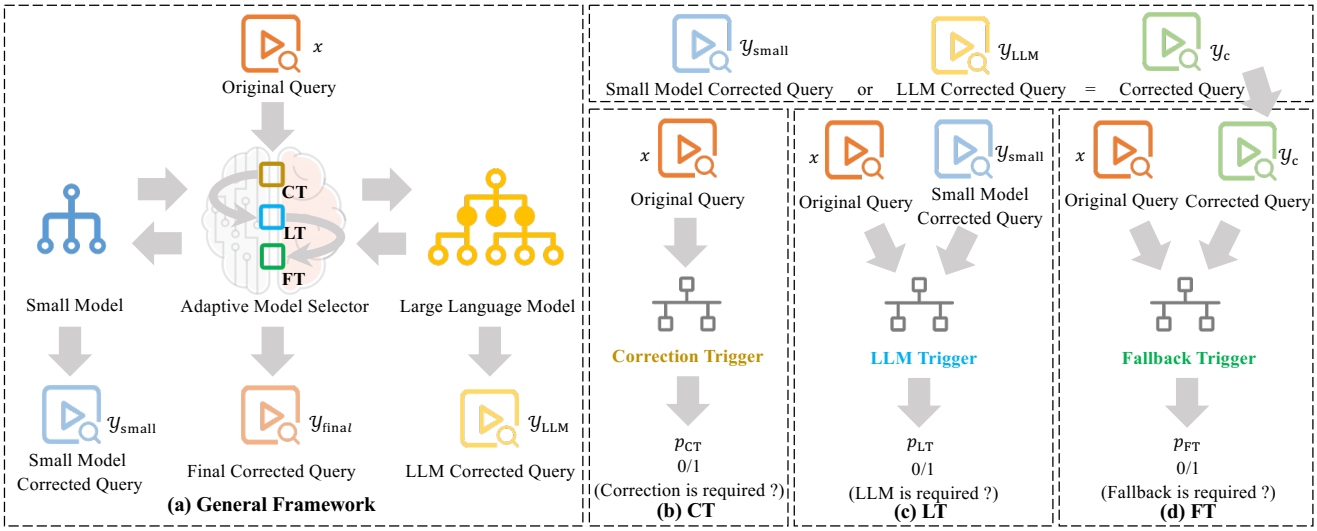


Figure 2: The architecture of the proposed framework Trigger^3. (a) The general framework of Trigger^3. (b) The Illustration of Correction Trigger (CT). (c) The Illustration of LLM Trigger (LT). (d) The Illustration of Fallback Trigger (FT).

3.4 The Second Trigger: LLM Trigger

After the small model's correction, we use a LLM Trigger (LT) to decide whether to invoke the Large Language Model (LLM). Considering that the LLM may not be able to solve the problem either, we hope to use LT to identify the queries that the small model cannot correct but the LLM can. Given the pair of the original query and the query preliminarily rewritten by the small model (x, y_{small}), LT is a scoring function that indicates the probability of calling LLM:

$$\begin{aligned} p_{\text{LT}} &= P(\text{Invoke LLM}|x, y_{\text{small}}) \\ &= f_{\text{LT}}(x, y_{\text{small}}; \theta_{\text{LT}}), \end{aligned} \tag{5}$$

where y_{small} is the output of the small model. We use the [SEP] token to separate x and y_{small}, and take the representation of the [CLS] token to get the score p_{LT}.

3.5 The Third Trigger: Fallback Trigger

Considering that both small and large models may not be able to correct some queries such as real-time news queries or domain-specific queries, which, if modified, may seriously damage the user search experience, as shown in Figure 1 (b), it is better to use the original query. This operation is inspired by the research about LLM's refusal to answer (Chen et al. 2024) and LLM security (Zheng et al. 2024; Sun et al. 2023).

After the small model or LLM correction, we can review the rewrite and choose whether to return the original query based on the corrected query and the original query. Given the original query and corrected query, p_{FT} is used to indicate the probability of returning the original query:

$$\begin{aligned} p_{\text{FT}} &= P(\text{Return } x|x, y_{\text{c}}) \\ &= f_{\text{FT}}(x, y_{\text{c}}; \theta_{\text{FT}}), \end{aligned} \tag{6}$$

where y_{c} is either y_{small} or y_{LLM}, which can be known according to Algorithm 1. We use the [SEP] token to separate x and y_{c}, and take the representation of the [CLS] token to get the score p_{FT}.

3.6 Model Training in Trigger^3

In Trigger^3, for the three modules, we use the widely used binary cross-entropy loss (Devlin et al. 2019) as the objective function:

$$\begin{aligned} \mathcal{L}_{\text{XT}} = -\frac{1}{|\mathcal{D}_{\text{XT}}|} \sum_{\mathcal{D}_{\text{XT}}} & y_{\text{XT}} \log(p_{\text{XT}}) \\ & + (1 - y_{\text{XT}}) \log(1 - p_{\text{XT}}), \end{aligned} \tag{7}$$

where $\text{XT} \in \{\text{CT}, \text{LT}, \text{FT}\}$, y_{XT} is the label and p_{XT} is the prediction score.

For $\mathcal{D}_{\text{CT}}$, we take the wrong query in the training dataset as the positive sample and the correct query as the negative sample.

Before introducing the dataset construction for $\mathcal{D}_{\text{LT}}$ and $\mathcal{D}_{\text{FT}}$, we first introduce a few character-edit-based indicators that will be used later: True positive (TP) indicates whether the model has correct edits, False positive (FP) indicates whether the model's edits have changed the correct characters into the wrong ones, and False negative (FN) indicates whether the model's edits have missed any necessary changes for the correct query. For the small model's editing indicators, we represent them as TP_{S}, FP_{S}, FN_{S}. For the LLM, we represent them as TP_{L}, FP_{L}, FN_{L}.

For $\mathcal{D}_{\text{LT}}$, we use the queries that **small model can't correct, but LLM can** as the positive samples. Specifically, a query is determined to be a positive sample for LT as long as it meets any of the following three points: 1) The small model does not have correct edits, but the LLM does. 2) The small model has incorrect edits, but the LLM does not. 3) The small model has missed necessary edits, but the LLM does not, i.e., the LLM has completely corrected this query.

Train	Avg len	#Query	Error Rate
Commercial	9.43	1,444,213	97.8%
QQ	9.81	111,703	79.1%
Valid	**Avg len**	**#Query**	**Error Rate**
Commercial	9.41	14,737	97.8%
QQ	9.78	12,412	75.1%
Test	**Avg len**	**#Query**	**Error Rate**
Commercial	9.43	14,737	97.8%
QQ	9.79	13,791	74.7%

Table 1: Statistics of the used query correction datasets. **Avg len** is the average length of the original query, **#Query** denotes the number of the queries and **Error Rate** denotes the percentage of the incorrect queries.

This can be represented as

$$\begin{aligned}(\mathrm{TP_S} < 0 \quad &\text{and} \quad \mathrm{TP_L} > 0)\\ \text{or } (\mathrm{FP_S} > 0 \quad &\text{and} \quad \mathrm{FP_L} < 0)\\ \text{or } (\mathrm{FN_S} > 0 \quad &\text{and} \quad \mathrm{FN_L} < 0).\end{aligned}$$

Negative samples are then sampled in the same quantity as positive samples, excluding all positive samples from the training dataset.

For $\mathcal{D}_{\mathrm{FT}}$, we use the queries that **both small model and LLM cannot correct** as the positive samples. Specifically, a query is determined to be a positive sample for FT if the editing of the rewritten query does not have a correct edit. We consider that both the small model and LLM do not have a correct edit, specifically represented as

$$\mathrm{TP_S} < 0 \quad \text{and} \quad \mathrm{TP_L} < 0.$$

Negative samples are then sampled in the training set, excluding all positive samples, with the same number of positive samples. The training details are in Section 4.1.

4 Experiments

4.1 Experimental Settings

Dataset We conduct query correction experiments on the following two datasets:

Commercial is based on the user search logs from a popular short video platform in 2024. The construction process of Commercial dataset is as follows: 50% of the data is obtained by rejecting samples from online correction logs with a rewriting confidence greater than 0.99. The remaining 50% of the data is generated from high-quality online queries through methods such as homophone substitution, near-sound character replacement, adjacent character transposition, and random character addition or deletion.

QQ is a publicly available search-related dataset, due to the lack of publicly available query correction datasets, we modify it as a query correction dataset. Following (Ye et al. 2023), we first use a language model to filter the queries, selecting those with a high probability of being correct. We then perform similar operations like Commercial dataset on these queries to construct a query correction dataset.

The statistics and the construction process of the datasets are shown in Table 1.

Metrics Following (Xu et al. 2022), we use the widely used metrics character-level and word-level precision (P)/recall (R)/F-measure ($\mathrm{F}_{0.5}$) from ChERRANT scorer (Zhang et al. 2022) to evaluate the correction performance.

Baselines In order to verify the validity of $\mathrm{Trigger}^3$, we consider the following correction model as the small model: GECToR, BART, mT5, which are short for GECToR-Chinese (Zhang et al. 2022), BART-Large (Shao et al. 2024) and mT5-Base (Xue et al. 2021). We consider the following LLM: Qwen1.5-7B-Chat (Bai et al. 2023) and Baichuan2-7B-Chat (Yang et al. 2023). We improve LLM's correction performance by fine-tuning it and applying it for direct correction (Single) and using small model rewrites as part of LLM prompts for corrections (Cascading).

We further combine the small model and LLM and compare $\mathrm{Trigger}^3$ to the following framework: Random-Routing, Routing (Lu et al. 2023; Šakota, Peyrard, and West 2024), HybridLLM (Ding et al. 2024), Random-Cascading and Margin Sampling (Ramírez, Birch, and Titov 2024). Specifically, we compare the correction performance of GECToR, BART, mT5, LLM itself and with $\mathrm{Trigger}^3$. Then, using these small models and LLMs, we further compare $\mathrm{Trigger}^3$ with the above frameworks.

Implementation Details Our code implementation is based on Huggingface Transformers (Wolf et al. 2020) in Pytorch. The fine tuning cost of LLM is much higher than that of small models. Therefore, following (Fan et al. 2023), for the fine tuning of LLM, we only used 1,000 pieces of data from the training dataset, while for the training of small models, we used all available training datasets. For the fine tuning of LLMs, we use LoRA (Hu et al. 2021) for efficient fine tuning. We utilize the Adam (Kingma and Ba 2014) optimizer, setting the initial learning rate to 5e-5, the batch size to 16, and applying a cosine learning rate schedule for 3 epochs. For the auxiliary models used in $\mathrm{Trigger}^3$ and all frameworks, we select ten thousand queries from the training dataset to fine-tune BERT (Devlin et al. 2019). All experiments are performed on NVIDIA V100 32GB GPUs. The source code, datasets, more experimental results and details can be found in the following link:

Code — https://github.com/ke-01/Trigger3.

4.2 Main Results

We investigate the correction performance of our proposed $\mathrm{Trigger}^3$. As shown in Table 2, which presents the correction performance on two datasets, we can draw the following conclusions:

• **Overall Performance**. $\mathrm{Trigger}^3$ surpasses all base small models, LLMs and frameworks in $\mathrm{F}_{0.5}$ while ensuring no decrease in recall rate. This demonstrates the effectiveness of our proposed $\mathrm{Trigger}^3$ in integrating the small model and LLM, taking into account the feedback from both when deciding whether to call the LLM and returning the original query strategy for queries that neither model corrects well.

• **Cascading vs. Routing**. We find that the cascade framework performs better overall in correction than the routing

Category	Model	Commercial						QQ					
		Character-level			Word-level			Character-level			Word-level		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Individual	**GECToR** (Small Model)	59.59	**76.30**	62.32	58.68	68.71	60.44	39.96	46.10	41.05	44.59	43.69	44.41
	Single (LLM)	45.47	42.87	44.92	45.57	40.96	44.56	41.57	40.50	41.35	43.93	37.78	42.55
	Cascading (LLM)	72.43	67.13	71.30	72.34	64.35	70.59	51.84	47.00	50.79	54.66	44.72	52.34
Combination	Random Routing	53.16	59.29	54.28	52.54	54.56	52.93	40.32	42.97	40.82	44.00	40.53	43.26
	Meta Routing	59.08	70.25	61.02	58.70	64.71	59.81	43.38	46.77	44.02	46.20	43.24	45.58
	HybridLLM	59.63	71.82	61.72	59.10	65.76	60.32	43.57	46.51	44.13	46.60	43.13	45.86
	Random Cascading	64.61	71.60	65.90	64.35	66.33	64.73	45.07	46.30	45.31	48.96	43.89	47.85
	Margin Sampling	66.56	71.41	67.48	66.29	66.54	66.34	47.25	46.57	47.11	51.24	44.70	49.79
	Trigger^3 (Ours)	**74.66**†	74.33	**74.60**†	**74.79**†	**71.33**†	**74.07**†	**60.09**†	**48.69**†	**57.40**†	**63.45**†	**46.96**†	**59.29**†
Individual	**BART** (Small Model)	73.52	71.99	73.21	73.91	71.54	73.42	59.83	60.51	59.97	62.26	62.11	62.23
	Single (LLM)	45.47	42.87	44.92	45.57	40.96	44.56	41.57	40.50	41.35	43.93	37.78	42.55
	Cascading (LLM)	72.73	62.55	70.43	73.05	61.79	70.48	55.73	52.41	55.03	58.57	51.32	56.96
Combination	Random Routing	59.64	57.24	59.14	60.14	56.16	59.30	50.55	50.13	50.47	53.56	49.64	52.73
	Meta Routing	68.68	65.77	68.08	69.06	64.97	68.20	60.78	60.52	60.73	63.87	60.72	63.21
	HybridLLM	71.12	68.64	70.61	71.66	68.08	70.91	61.82	60.44	61.54	64.92	60.65	64.02
	Random Cascading	73.23	67.43	71.99	73.73	66.83	72.24	57.52	56.03	57.21	60.11	56.21	59.29
	Margin Sampling	72.67	66.52	71.35	72.95	65.88	71.41	58.73	58.46	58.67	61.51	59.16	61.03
	Trigger^3 (Ours)	**76.57**†	**72.07**	**75.63**†	**76.86**†	**71.57**	**75.74**†	**66.31**†	**61.43**†	**65.27**†	**68.59**†	**62.17**	**67.20**†
Individual	**mT5** (Small Model)	67.42	59.04	65.56	68.44	58.02	66.06	54.71	52.01	54.15	56.61	51.02	55.40
	Single (LLM)	45.47	42.87	44.92	45.57	40.96	44.56	41.57	40.50	41.35	43.93	37.78	42.55
	Cascade (LLM)	66.18	54.90	63.57	66.90	53.79	63.79	50.48	49.02	50.18	52.86	46.56	51.46
Combination	Random Routing	56.05	51.00	54.96	56.66	49.60	55.09	47.78	45.86	47.38	50.32	44.06	48.93
	Meta Routing	64.08	57.62	62.67	64.98	56.51	63.09	57.84	52.14	56.60	60.87	50.71	58.53
	HybridLLM	66.71	58.44	64.87	67.52	57.21	65.17	59.14	52.22	57.61	62.00	50.90	59.41
	Random Cascading	66.96	57.17	64.74	67.94	56.10	65.19	52.29	50.01	51.82	54.27	48.08	52.91
	Margin Sampling	67.51	59.06	65.63	68.54	58.04	66.14	55.26	51.95	54.57	57.05	50.74	55.66
	Trigger^3 (Ours)	**69.64**†	**59.13**	**67.25**†	**70.44**†	**58.10**	**67.57**†	**61.36**†	**52.72**†	**59.41**†	**64.43**†	**51.29**	**61.29**†

Table 2: Performance comparisons between Trigger^3 and the baselines when the LLM is Qwen1.5-7B-Chat. Single: directly using LLM for correction. Cascading: using smaller model rewrites as part of LLM prompts. The LLMs use 1,000 data for fine tuning, while the small model use full training data for training. The boldface indicates the best performance, and the underline indicates the second performance. '†' indicates that the improvements are significant (t-tests, p-value < 0.05).

framework. This is mainly because, in the correction task, without the preliminary rewriting from the small model, direct correction by the LLM may result in over-correction, leading to poorer correction performance. This suggests that in the query correction task, the preliminary rewriting by the small model can serve as an implicit feature to help improve the LLM's correction performance.

• **Comparison of Different Small Models**. For different small models, we note that combining with the LLM improves the performance of Seq2Edit more significantly. This is mainly because the types of errors that Seq2Edit and Seq2Seq can correct are more complementary. This also reflects to some extent that the errors Seq2Seq and LLM can solve may be more alike. However, as the errors that the LLM and Seq2Seq small model can correct are different, this can also enhance the base model's correction performance.

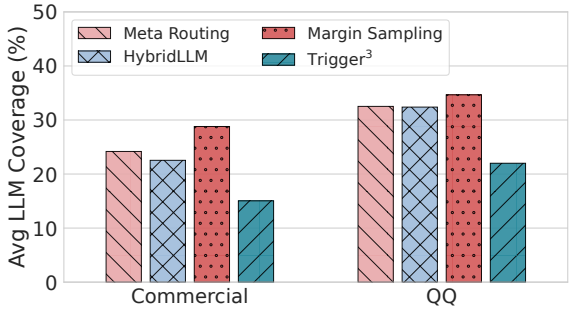


Figure 3: Average LLM Coverage of Trigger^3 and the three frameworks when the LLM is Qwen1.5-7B-Chat. The lower the bar, the better.

4.3 Ablation Study

Trigger^3 consists of three main components: CT (Correction Trigger), LT (LLM Trigger), and FT (Fallback Trigger). To explore the impact of different components on the correction performance, we conduct ablation experiments by adding these three components one by one. Although CT is the first module that the query goes through during inference, it does not carry out correction and therefore, cannot demonstrate the effect on correction performance. Hence, we add it last. The base models are the small model and the LLM in a cascade manner. The ablation results on Commercial and QQ datasets are shown in Table 3, and we provide detailed discussions for each module below:

+LT: This represents adding the LLM trigger to the base model and integrating LLM. It decides whether to call LLM for specific queries and only calls LLM when necessary. We can observe that adding LT consistently improves performance, reflecting the effectiveness of LT in integrating small

Model	Commercial		QQ	
	Char-$F_{0.5}$	Word-$F_{0.5}$	Char-$F_{0.5}$	Word-$F_{0.5}$
GECToR	62.32	60.44	41.05	44.41
LLM	71.30	70.59	42.29	45.48
+LT	73.33	72.83	55.91	57.91
+FT	74.17	73.66	56.63	58.49
+CT	**74.60**	**74.07**	**57.40**	**59.29**
BART	73.21	73.42	59.97	62.23
LLM	70.43	70.48	52.79	55.38
+LT	74.90	75.03	64.52	66.40
+FT	75.21	75.33	64.67	66.58
+CT	**75.63**	**75.74**	**65.27**	**67.20**
mT5	65.56	66.06	54.15	55.40
LLM	63.57	63.79	50.18	51.46
+LT	66.69	67.07	57.22	59.03
+FT	67.11	67.44	58.28	60.10
+CT	**67.25**	**67.57**	**59.41**	**61.29**

Table 3: Ablation studies of Trigger^3 on Commercial and QQ datasets when the LLM is Qwen1.5-7B-Chat. The boldface indicates the best performance.

models and LLMs.

+FT: This represents adding the fallback trigger, which reviews the correction results. It decides whether to return the original query based on the original and corrected queries. If neither of the models can correct the query, we return the original query. Adding FT improves correction performance on both datasets and all three small models, demonstrating its effectiveness.

+CT: This represents adding the correction trigger, which judges the correctness of the input query. For queries that are correct, there is no need for models to correct. Adding CT also improves correction performance. We attribute this improvement to its similar function to FT. Queries that are already correct do not need correction, and having the small model and LLM correct them may actually decrease correction performance.

4.4 Efficiency Analysis

In the process of deploying the model, considering the possibility of parallel pipeline execution, the portion of the query processed by LLMs often becomes a bottleneck for efficiency. At the same time, a widely recognized basic assumption from previous research (Ramírez, Birch, and Titov 2024; Lu et al. 2023) in the field of efficient inference is that smaller models are more inference-efficient than larger models. Based on this concept, similar to (Ding et al. 2024), we use the proportion of queries addressed by LLM as an indicator to evaluate efficiency, termed as LLM coverage:

$$\text{LLM coverage} = \frac{\text{The number of queries corrected by LLM}}{\text{The total number of queries}}$$

The average LLM coverage (the mean of the LLM coverage across three small models within the framework) of Trigger^3 and the other three frameworks on two datasets can be found in Figure 3. In conjunction with Table 2, we

Small	Framework	Commercial		QQ	
		$\mathbf{F}_{0.5}$	**LC**	$\mathbf{F}_{0.5}$	**LC**
GECToR	Meta Routing	61.02	38.22	44.02	39.13
	HybridLLM	61.72	33.77	44.13	39.82
	Margin	67.48	45.63	47.11	64.46
	Trigger^3	**74.60**	**32.09**	**57.85**	**31.24**
BART	Meta Routing	68.08	16.88	60.73	32.11
	HybridLLM	70.61	12.57	61.54	31.02
	Margin	71.35	28.69	58.67	33.55
	Trigger^3	**75.63**	**3.84**	**65.27**	**18.09**
mT5	Meta Routing	62.67	17.45	56.60	26.34
	HybridLLM	64.87	21.26	57.61	26.33
	Margin	65.63	12.09	54.57	**5.97**
	Trigger^3	**67.25**	**9.19**	**59.41**	16.66

Table 4: Efficiency comparisons between Trigger^3 and other frameworks. The boldface indicates optimal performance and optimal efficiency. **LC** is short for LLM Coverage, which denotes the proportion of queries solved by LLM. $\mathbf{F}_{0.5}$ is Char-$F_{0.5}$. Margin is short for Margin Sampling.

can find that Trigger^3 maintains high efficiency while improving correction performance, mainly due to the following two reasons: 1) Trigger^3 considers excluding the queries that are correct themselves before making corrections and uses CT to filter out the correct queries. 2) Before Trigger^3 hands over the queries to LLM for correction, it considers that only the queries that LLM can correct are handed over to LLM for processing.

The proportion of queries handled by LLM on three different small models for each framework can be found in Table 4. Take a concrete example, if the dataset is Commercial, the LLM is Qwen1.5-7B-Chat, and the small model is GECToR, the LLM coverage is 32.09. For about 67.91% of queries, only a small model is enough. The proportion of queries corrected by other LLMs and small models combinations can be similarly obtained from the examples above. We find that Trigger^3 not only maintains high correction performance but also ensures efficiency.

5 Conclusion

In this paper, we propose a large-small model collaboration framework, Trigger^3, to adaptively perform query correction. Specifically, Trigger^3 uses three triggers to integrate the small model and LLM for query correction. First, before performing query correction, it judges the correctness of the query and selects the incorrect query to be corrected by the small model. Second, after the small model correction, it selects the queries that the small model cannot correct but the LLM can, and hands them over to LLM for correction. Finally, after the LLM correction, it reviews and selects the queries that neither the LLM nor the small model can correct, and returns the original query as output. The superiority and efficiency of Trigger^3's correction performance are validated through extensive experiments.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 62376275, 92470205, 62377044), Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. Supported by fund for building world-class universities (disciplines) of Renmin University of China. Supported by Public Computing Cloud, Renmin University of China. Supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (23XNKJ13). Supported by Kuaishou Technology.

References

Awasthi, A.; Sarawagi, S.; Goyal, R.; Ghosh, S.; and Piratla, V. 2019. Parallel Iterative Edit Models for Local Sequence Transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4260–4270.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; Hui, B.; Ji, L.; Li, M.; Lin, J.; Lin, R.; Liu, D.; Liu, G.; Lu, C.; Lu, K.; Ma, J.; Men, R.; Ren, X.; Ren, X.; Tan, C.; Tan, S.; Tu, J.; Wang, P.; Wang, S.; Wang, W.; Wu, S.; Xu, B.; Xu, J.; Yang, A.; Yang, H.; Yang, J.; Yang, S.; Yao, Y.; Yu, B.; Yuan, H.; Yuan, Z.; Zhang, J.; Zhang, X.; Zhang, Y.; Zhang, Z.; Zhou, C.; Zhou, J.; Zhou, X.; and Zhu, T. 2023. Qwen Technical Report. *arXiv preprint arXiv:2309.16609*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, J.; Lin, H.; Han, X.; and Sun, L. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17754–17762.

Coyne, S.; and Sakaguchi, K. 2023. An Analysis of GPT-3's Performance in Grammatical Error Correction. *arXiv preprint arXiv:2303.14342*.

Davis, C.; Caines, A.; Andersen, Ø.; Taslimipoor, S.; Yannakoudakis, H.; Yuan, Z.; Bryant, C.; Rei, M.; and Buttery, P. 2024. Prompting open-source and commercial language models for grammatical error correction of English learner text. *arXiv preprint arXiv:2401.07702*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Ding, D.; Mallick, A.; Wang, C.; Sim, R.; Mukherjee, S.; Rühle, V.; Lakshmanan, L. V. S.; and Awadallah, A. H. 2024. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *The Twelfth International Conference on Learning Representations*.

Ding, J.; Tarokh, V.; and Yang, Y. 2018. Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6): 16–34.

Fan, Y.; Jiang, F.; Li, P.; and Li, H. 2023. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 69–80. Springer.

Fang, T.; Yang, S.; Lan, K.; Wong, D. F.; Hu, J.; Chao, L. S.; and Zhang, Y. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Y.; Huang, H.; Ma, S.; Jiang, Y.; Li, Y.; Zhou, F.; Zheng, H.-T.; and Zhou, Q. 2023. On the (in) effectiveness of large language models for chinese text correction. *arXiv preprint arXiv:2307.09007*.

Liang, D.; Zheng, C.; Guo, L.; Cui, X.; Xiong, X.; Rong, H.; and Dong, J. 2020. BERT enhanced neural machine translation and sequence tagging model for Chinese grammatical error diagnosis. In *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, 57–66.

Liu, Y.; Yao, Y.; Ton, J.-F.; Zhang, X.; Cheng, R. G. H.; Klochkov, Y.; Taufiq, M. F.; and Li, H. 2023. Trustworthy LLMs: A survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*.

Lu, K.; Yuan, H.; Lin, R.; Lin, J.; Yuan, Z.; Zhou, C.; and Zhou, J. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Madaan, A.; Aggarwal, P.; Anand, A.; Potharaju, S. P.; Mishra, S.; Zhou, P.; Gupta, A.; Rajagopal, D.; Kappaganthu, K.; Yang, Y.; et al. 2023. Automix: Automatically mixing language models. *arXiv preprint arXiv:2310.12963*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Pande, M.; Kakkar, V.; Bansal, M.; Kumar, S.; Sharma, C.; Malhotra, H.; and Mehta, P. 2022. Learning-to-Spell: Weak Supervision based Query Correction in E-Commerce Search with Small Strong Labels. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 3431–3440.

Ramírez, G.; Birch, A.; and Titov, I. 2024. Optimising Calls to Large Language Models with Uncertainty-Based Two-Tier Selection. *arXiv preprint arXiv:2405.02134*.

Šakota, M.; Peyrard, M.; and West, R. 2024. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 606–615.

Shao, Y.; Geng, Z.; Liu, Y.; Dai, J.; Yan, H.; Yang, F.; Li, Z.; Bao, H.; and Qiu, X. 2024. Cpt: A pre-trained unbalanced transformer for both chinese language understanding and generation. *Science China Information Sciences*, 67(5): 1–13.

Shen, C.; Zhang, X.; Shi, T.; Zhang, C.; Xie, G.; and Xu, J. 2024. A survey of controllable learning: Methods and applications in information retrieval. *arXiv preprint arXiv:2308.05374*.

Sun, H.; Zhang, Z.; Deng, J.; Cheng, J.; and Huang, M. 2023. Safety assessment of chinese large language models. *arXiv preprint arXiv:2304.10436*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 38–45.

Xu, L.; Wu, J.; Peng, J.; Fu, J.; and Cai, M. 2022. FCGEC: Fine-Grained Corpus for Chinese Grammatical Error Correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 1900–1918.

Xue, L.; Constant, N.; Roberts, A.; Kale, M.; Al-Rfou, R.; Siddhant, A.; Barua, A.; and Raffel, C. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 483–498.

Yang, A.; Xiao, B.; Wang, B.; Zhang, B.; Bian, C.; Yin, C.; Lv, C.; Pan, D.; Wang, D.; Yan, D.; et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Ye, D.; Tian, B.; Fan, J.; Liu, J.; Zhou, T.; Chen, X.; Li, M.; and Ma, J. 2023. Improving Query Correction Using Pretrain Language Model In Search Engines. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2999–3008.

Yue, M.; Zhao, J.; Zhang, M.; Du, L.; and Yao, Z. 2023. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*.

Zhang, J.; Krishna, R.; Awadallah, A. H.; and Wang, C. 2023. Ecoassistant: Using llm assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046*.

Zhang, X.; and Liao, S. 2020. Hypothesis sketching for online kernel selection in continuous kernel space. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2498–2504.

Zhang, X.; Liao, Y.; and Liao, S. 2019. A survey on online kernel selection for online kernel learning. *WIREs Data Mining and Knowledge Discovery*, 9(2): e1295.

Zhang, Y.; Li, Z.; Bao, Z.; Li, J.; Zhang, B.; Li, C.; Huang, F.; and Zhang, M. 2022. MuCGEC: a Multi-Reference Multi-Source Evaluation Dataset for Chinese Grammatical Error Correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3118–3130.

Zhang, Z.; Zhang, H.; Chen, K.; Guo, Y.; Hua, J.; Wang, Y.; and Zhou, M. 2021. Mengzi: Towards lightweight yet ingenious pre-trained models for chinese. *arXiv preprint arXiv:2110.06696*.

Zhao, Z.; and Wang, H. 2020. Maskgec: Improving neural grammatical error correction via dynamic masking. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1226–1233.

Zheng, C.; Yin, F.; Zhou, H.; Meng, F.; Zhou, J.; Chang, K.-W.; Huang, M.; and Peng, N. 2024. On prompt-driven safeguarding for large language models. In *Forty-first International Conference on Machine Learning*.