

From Historical Queries to Actionable Rubrics: Category-Aware Rubric Generation for Local-Life AI Search

Jiale Mao^{1,*}, Fuwei Zhang^{1,*}, Dongbo Xi³, Zhen Chen³
Yifan Yang³, Peng Yan³, Fuzhen Zhuang^{1,†}, Zhao Zhang^{2,†}

¹Institute of Artificial Intelligence, Beihang University

²School of Computer Science and Engineering, Beihang University

³Meituan

zy2542210@buaa.edu.cn, zhangfuwei@buaa.edu.cn, xidongbo@meituan.com, chenzhen06@meituan.com

yifan.yang@meituan.com, yanpeng04@meituan.com, zhuangfuzhen@buaa.edu.cn, zhao_zhang@buaa.edu.cn

Abstract

Rubric generation improves the interpretability and actionability of large language model evaluation, but in online-to-offline (O2O) local-life AI search, user queries often reflect more complex needs, such as consumption intent, location constraints, and scenario preferences. Directly generating rubrics from a single query tends to produce generic criteria that fail to satisfy AI search evaluation needs, while leveraging candidate responses, preference samples, or external evidence at scale remains challenging. To address these limitations, we propose Category-Aware Rubric Generation (CaRG), a category-aware evaluation dimension candidate pool framework. Through embedding-based clustering, representative selection, and iterative LLM-based consolidation, CaRG constructs reusable candidate pools of both general and category-specific evaluation dimensions from representative historical queries. Subsequently, these candidate pools are leveraged to generate query-level rubrics with concrete positive and negative scoring criteria for target queries. Experimental results demonstrate that CaRG consistently outperforms Direct-Rubric and shows more stable overall quality than existing query-only baselines, while achieving the strongest overall agreement with production golden rubric rankings. Our repository is available at <https://github.com/PlussLe/CaRG>.

1 Introduction

As the capabilities of large language models continue to improve, an increasing body of work has explored their applications across various domains

*Jiale Mao and Fuwei Zhang were student interns at Meituan, and this work was completed during their internships. Zhao Zhang and Fuzhen Zhuang, faculty members at Beihang University, provided remote academic guidance only and did not access or handle any physical business materials or on-site operational assets.

†Corresponding authors: Fuzhen Zhuang and Zhao Zhang

(Zhang et al., 2026a, 2025, 2026b; Guo et al., 2026; Feng et al., 2024; Wang et al., 2024a); meanwhile, rubric generation has also emerged as an important research direction in the field of LLM evaluation. Compared with assigning a single holistic score, rubric-based evaluation can provide more interpretable, transparent, and actionable assessments by identifying where a response succeeds and where it fails. In LLM-augmented search and recommender systems, rubrics can further support explainable quality diagnosis, evaluation refinement, and fine-grained feedback for reward modeling or preference optimization.

In online-to-offline (O2O) local-life AI search, user queries often convey complex and concrete consumption needs. For example, users may seek a restaurant for a date, a nearby family-friendly activity, or a hotel that satisfies specific budget and location constraints. Unlike general question-answering tasks, these queries involve not only relevance but also consumption scenarios, geographic constraints, budget requirements, user-group suitability, and merchant reliability. As a result, an effective rubric must capture both user intent and practical decision-making requirements.

Meanwhile, large-scale local-life platforms naturally accumulate massive historical queries with business-category labels. Although individual queries vary in their specific needs, queries within the same category often share common evaluation concerns. For example, food-related queries may emphasize cuisine, price, dining scenarios, and merchant reputation, whereas hotel-related queries may focus on location, room type, facilities, and transportation convenience. This observation suggests that historical queries contain reusable evaluation knowledge that can benefit query-level rubric generation.

However, existing methods do not fully exploit historical queries. While prior approaches can generate rubrics from the current query or leverage

additional supervision signals such as candidate responses, preference data, and external evidence, they often fail to effectively utilize category-level evaluation knowledge embedded in large-scale historical queries. How to extract reusable evaluation dimensions from historical queries and apply them to rubric generation for new queries remains an open challenge.

To address this challenge, we propose Category-Aware Rubric Generation (CaRG). CaRG first constructs reusable general and category-specific candidate rubric pools from historical queries and their category labels in an offline stage. Given a target query, it then incorporates these candidate pools as category-level knowledge to generate query-level rubrics that combine general evaluation requirements, category-specific knowledge, and query-specific intent. It should be clarified, however, that CaRG is not intended to replace expert human judgment: human experts are better suited to defining stable platform-level base rubrics, while CaRG investigates how reusable category-level knowledge from historical queries can help LLMs generate query-specific rubrics for diverse local-life search needs.

Our contributions are as follows:

- We formalize query-level rubric generation for O2O local-life AI search and highlight the challenges posed by complex user intent and decision-oriented evaluation requirements.
- We propose Category-Aware Rubric Generation (CaRG). Without relying on human-written query-specific rubrics, candidate responses, or preference data, CaRG automatically constructs reusable general and category-specific evaluation dimension pools from historical queries.
- We conduct extensive experiments on real-world local-life search queries, demonstrating that CaRG generates higher-quality rubrics and achieves stronger alignment with practical business evaluation of model-response rankings, providing a scalable approach for deploying rubrics in industrial O2O local-life AI search scenarios.

2 Related Work

Rubric-based evaluation has become an important paradigm for assessing LLM outputs. Unlike holistic scoring, rubrics decompose evaluation into multiple fine-grained dimensions, making the evaluation process more interpretable and actionable. Early LLM-as-a-judge work

shows that large language models can approximate human judgments in open-ended generation tasks (Liu et al., 2023; Zheng et al., 2023). Subsequently, PandaLM (Wang et al., 2024b), Auto-J (Li et al., 2024), and JudgeLM (Zhu et al., 2025) further train specialized judge models on manually constructed or real-world collected training sets to evaluate model responses. In addition, LLMBBar (Zeng et al., 2024) and JudgeBench (Tan et al., 2025) construct more challenging meta-evaluation benchmarks and find that LLM judges still have substantial room for improvement on difficult examples involving deceptive qualities or complex reasoning. On this basis, some work further uses rubrics for fine-grained evaluation: Prometheus (Kim et al., 2024) studies training a fine-grained evaluation model given a scoring rubric and reference answer; LLM-Rubric (Hashemi et al., 2024) uses manually constructed rubrics to conduct multidimensional evaluation of natural language texts; ResearchRubrics (Sharma et al., 2026) constructs a benchmark containing expert-written fine-grained rubrics for evaluating the long-form answer quality of Deep Research systems.

Building on the above rubric-guided evaluation studies, recent work has increasingly focused on automatic rubric or evaluation-criteria generation. Automatic rubric or evaluation-criteria generation methods can be broadly divided into two categories. The first category directly generates evaluation criteria from the current query or instruction. Representative methods include checklist generation and analysis (Cook et al., 2024; Wei et al., 2025; Furuhashi et al., 2025), query-related test question generation (Farzi and Dietz, 2024), task decomposition (Saha et al., 2024), recursive expansion (Gao et al., 2026), and self-adaptive or question-specific rubric generation (Fan et al., 2024; Pathak et al., 2025).

Another category of work introduces extra information during rubric generation, refinement, or filtering. RRD (Shen et al., 2026) starts from existing coarse rubrics and obtains finer-grained and less redundant criteria through recursive decomposition and filtering. OpenRubrics (Liu et al., 2026b) and CDRRM (Liu et al., 2026a) use preferred and rejected responses to generate or refine rubrics. EvalAgents (Wadhwa et al., 2025) discovers implicit evaluation criteria from expert-authored web guidance. RubricRAG (Dhole and Agichtein, 2026) retrieves rubric examples from

semantically similar queries as few-shot context for current rubric generation, and another line of work uses human preferences to train a query-specific rubric generator (Lv et al., 2026).

Existing methods do not fully exploit reusable business knowledge accumulated in historical queries, while additional information such as candidate responses and preference pairs is difficult to construct stably in large-scale O2O AI search. To address these limitations, this paper proposes CaRG. CaRG only uses category labels and historical queries, which are usually naturally available in large-scale local-life platforms, and therefore provides a more scalable and reusable source of category-level evaluation knowledge.

3 Method

CaRG is a category-aware rubric generation framework for O2O local search: we first induce reusable evaluation dimension candidate pools from historical platform queries, and when generating the rubric for a target query, we inject both the general candidate pool and the corresponding category-specific candidate pool into the LLM as structured business knowledge.

3.1 Task Formulation

The input to the rubric generation task is a user search query and its first-level business category, denoted as (q, c) , where q is the user query and c is the first-level category. The goal is to generate a set of query-level rubrics, formulated as:

$$R(q, c) = \{r_1, r_2, \dots, r_m\}, \quad (1)$$

where m denotes the number of generated rubric dimensions. Each rubric dimension r_j consists of a dimension name, a positive scoring criterion, and a negative scoring criterion:

$$r_j = (n_j, g_j, b_j). \quad (2)$$

Here, n_j denotes the dimension name, g_j denotes the positive criterion describing how a high-quality textual response should satisfy the user’s need, and b_j denotes the negative criterion describing typical errors that a low-quality response may make under this dimension.

3.2 Overview

Figure 1 presents the unified CaRG framework, which consists of two stages: category-aware

candidate pool construction and candidate-pool-augmented query-level rubric generation. It is worth noting that the candidate pools are not fixed templates, they only serve as an “inspiration knowledge base” for the LLM, providing broad and well-structured evaluation directions. The final rubric still needs to be concretized according to the explicit requirements, implicit concerns, and consumption scenario of the current query.

3.3 Stage I: Category-Aware Candidate Pool Construction

To control cost while covering diverse business scenarios, we sample a representative query set $S = \{(q_i, c_i)\}_{i=1}^N$ from the full historical query dataset D . We first bucket queries by the combination of first-level and second-level categories, assign a base sampling quota to each bucket, allocate the remaining quota preferentially to larger buckets, and then perform systematic interval sampling within each bucket.

For each sampled query (q_i, c_i) , an LLM generates initial general and category-specific dimensions:

$$(G_i, V_i) = \text{LLM}_{\text{init}}(q_i, c_i). \quad (3)$$

General dimensions capture evaluation directions shared across O2O AI search, while category-specific dimensions capture business concerns specific to the current first-level category. Each dimension contains both a generalized definition and a query-specific application; during candidate-pool construction, we only retain the generalized definition to preserve transferable evaluation concepts rather than query-specific examples.

We aggregate all general dimensions into an initial general candidate pool $P_{\text{gen}}^0 = \bigcup_i G_i$, and aggregate category-specific dimensions into category pools $P_c^0 = \bigcup_{i:c_i=c} V_i$. For representation learning, each dimension is normalized into the textual form of “dimension name: generalized definition”. We then encode each normalized dimension text with BGE-M3 (Chen et al., 2024), apply L2 normalization, and conduct hierarchical clustering separately within the general pool and each category-specific pool, using cosine distance, average linkage, and a distance threshold of 0.15. For each cluster, we select the dimension closest to the cluster centroid as its representative, compressing semantically similar dimensions while preserving coverage over different evaluation directions.

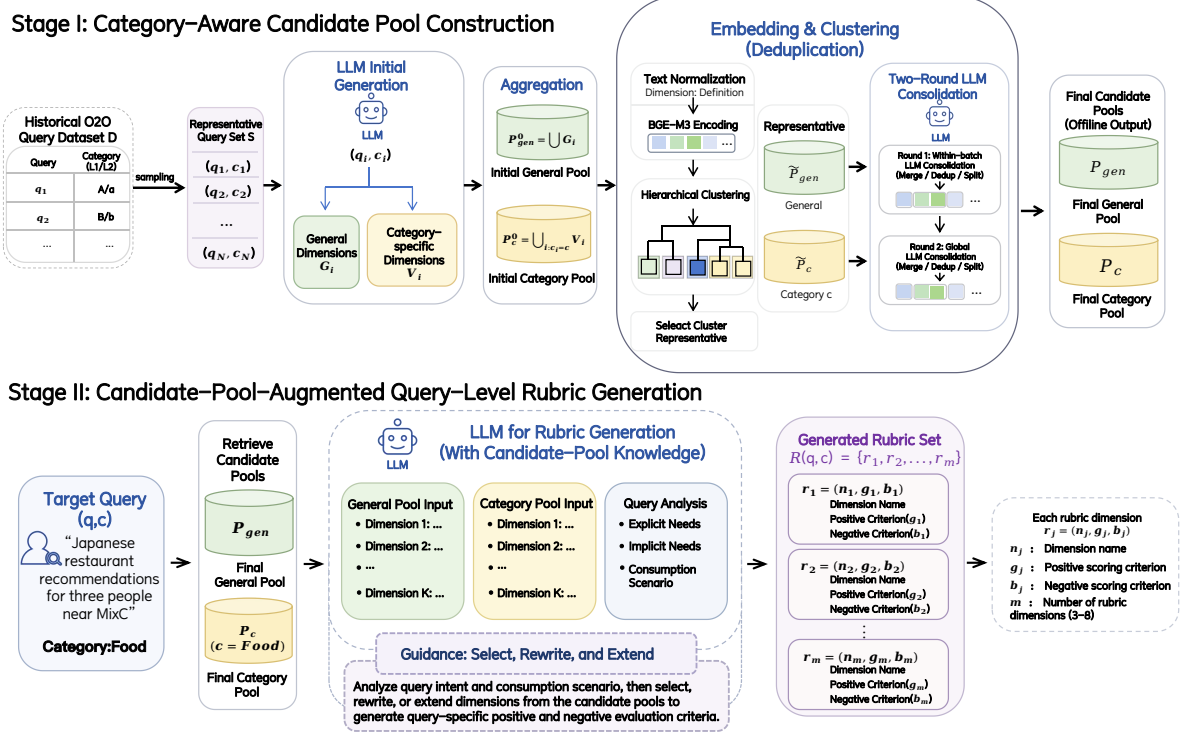


Figure 1: Unified overview of the CaRG framework, including offline construction of general and category-specific evaluation dimension candidate pools from historical query data and candidate-pool-augmented rubric generation for a target query.

Finally, we use two rounds of LLM consolidation to construct the final candidate pools. The first round performs within-batch merging, deduplication, and splitting of entangled dimensions over clustered representatives; the second round performs global consolidation to remove cross-batch redundancy and enforce independence and atomicity. The outputs are the final general pool P_{gen}^* and category-specific pools P_c^* , where each candidate dimension has a standardized name and definition.

3.4 Stage II: Candidate-Pool-Augmented Query-Level Rubric Generation

Given a target query q and its first-level category c , we retrieve the final general candidate pool P_{gen}^* and the corresponding category-specific pool P_c^* from Stage I, and input them together with the target query into the LLM. The final rubric generation process is formulated as:

$$R(q, c) = \text{LLM}_{\text{rubric}}(q, c, P_{gen}^*, P_c^*). \quad (4)$$

Here, P_{gen}^* provides cross-category evaluation directions, while P_c^* provides vertical business knowledge for the current category. The model first analyzes the explicit requirements, implicit intent, and potential consumption scenario behind

the target query. It then selects, rewrites, or extends the most relevant dimensions from the two candidate pools, producing 3 to 8 query-specific rubric dimensions.

Each output dimension follows the task formulation:

$$r_j = (n_j, g_j, b_j), \quad (5)$$

where n_j is the dimension name, and g_j and b_j are the positive and negative scoring criteria under that dimension. When generating dimensions, the model follows the principles that positive and negative criteria should be concrete and actionable, and that different dimensions should avoid overlap as much as possible.

4 Experiments

4.1 Experiment Setup

We evaluate generated rubrics from two complementary perspectives. The Rubric Quality Evaluation focuses more on rubric quality itself, testing whether different methods can better characterize query intent, provide actionable scoring criteria, and maintain a clear dimensional structure. The Online AI Search Evaluation further evaluates whether the scores obtained using automatically

generated rubrics can align with the golden ranking produced by an online production rubric.

4.1.1 Datasets

The candidate pools used by CaRG are constructed from 53,934 historical queries, among which 3,177 representative queries are sampled for candidate-dimension induction. Detailed data construction, candidate-pool implementation, and parameter settings are reported in Appendices A.1, A.2, and A.4. The Rubric Quality Evaluation uses a standard evaluation set of 600 queries sampled proportionally by first-level category; the Online AI Search Evaluation uses responses from two model versions on the same online AI search benchmark, with the version ranking produced by a human-annotated production rubric used as the golden ranking.

4.1.2 Baselines

We compare CaRG with four query-only baselines: Direct-Rubric, TICK (Cook et al., 2024), BSM, and Qworld (Gao et al., 2026). BSM is an adaptation of branch-solve-merge (Saha et al., 2024) to our setting, with implementation details provided in Appendix A.3. To reduce cost, Qworld uses the lightweight configuration. Except that CaRG uses the query’s first-level category to retrieve the category-specific candidate pool, all methods only use the query and do not rely on gold responses, preferred or rejected responses, human annotations, or external retrieval evidence. Detailed baseline and parameter settings are provided in Appendices A.3 and A.4, and prompt templates are provided in Appendix D.

4.1.3 Implementation Details

The Rubric Quality Evaluation adopts blind pairwise LLM meta-evaluation. Specifically, for each query, the judge model receives two anonymized evaluation standards, one from Direct-Rubric and the other from the method under evaluation. The judge uses the following three dimensions to score and compare the two standards, and derives the overall preference based on these three dimensions: 1) #Intent: whether the evaluation standard captures the explicit needs, implicit intent, and local-life business pain points behind the query; 2) #Action: whether the scoring criteria are concrete and executable; and 3) #MECE: whether the dimensions have clear boundaries, sufficient coverage, and avoid repeated penalties. Considering

Generator	Method	Win	#Intent	#Action	#MECE	Total
DeepSeek-v4-flash	Qworld	6.5%	4.75	3.38	5.60	4.38
	BSM	52.5%	6.76	7.07	7.28	7.01
	TICK	1.8%	5.50	4.20	6.27	5.17
	CaRG	58.7%	7.62	7.16	7.03	7.26
GPT-4o-mini	Qworld	3.3%	3.66	2.37	4.54	3.38
	BSM	66.3%	6.42	6.77	6.75	6.60
	TICK	8.8%	4.91	4.22	5.60	4.79
	CaRG	64.3%	7.57	6.55	6.82	6.88
GPT-4.1	Qworld	7.8%	5.11	3.43	5.75	4.55
	BSM	32.0%	5.59	6.27	7.08	6.24
	TICK	0.5%	5.04	3.82	6.04	4.81
	CaRG	81.7%	8.17	7.53	7.07	7.58
Gemini-3.1-pro-preview	Qworld	14.2%	5.94	4.80	6.22	5.54
	BSM	37.0%	7.03	7.05	7.21	7.07
	TICK	0.8%	5.28	4.05	6.45	5.10
	CaRG	80.7%	8.39	7.94	7.56	7.98

Table 1: Results of Rubric Quality Evaluation. Win denotes the pairwise win rate against Direct-Rubric, and each score is the mean score of the evaluated method. Bold marks the highest value in each column under the same generator.

possible position bias, we randomly swap the positions of the two outputs and map the final statistics back to the original sources.

The Online AI Search Evaluation uses rubrics or checklists generated by different methods to score the responses of two model versions on the online AI search benchmark, and compares whether the resulting version-level win/loss decisions agree with the golden ranking produced by the human-annotated production rubric. We use four metrics: Accuracy, Cohen’s Kappa, Pearson r , and Spearman ρ . Accuracy measures agreement on the version-level win/loss label, Kappa measures classification agreement after accounting for chance agreement, Pearson r measures the linear correlation between the automatic score difference and the golden score difference, and Spearman ρ measures rank correlation. Detailed metric computation is provided in Appendix A.5.

4.2 Rubric Quality Evaluation

Table 1 reports results of Rubric Quality Evaluation under different generation models, with gpt-4.1 fixed as the judge model.

From Table 1, we observe two main results: 1) CaRG consistently outperforms Direct-Rubric across all generation models and achieves the best or near-best win rate among query-only baselines. Specifically, CaRG obtains the highest win rate in 3 out of 4 generator settings and remains comparable to BSM under gpt-4o-mini. The gains are especially strong under stronger generators, suggesting that the candidate-pool mechanism helps the model better use evaluation knowledge distilled from historical queries and category structure; 2) Beyond the overall win rate, CaRG is also more balanced across #Intent, #Action, and

#MECE under stronger generators, with a particularly clear advantage in #Intent. This indicates that the candidate pools help the model capture explicit needs, implicit intent, and local-life business scenarios behind the query. Cross-judge robustness results are reported in Appendix B.3.

A qualitative example further illustrates this difference. For the query “nearby self-service hotpot buffet”, Direct-Rubric captures the basic intent of nearby, self-service, and hotpot, but its dimensions remain relatively general, covering category-location matching, decision-information completeness, entity authenticity, and text readability. In contrast, CaRG generates more O2O-specific dimensions, such as buffet-rule and price transparency, dish supply and feature presentation, LBS-aware location description, and reputation reference with pitfall avoidance. These dimensions explicitly cover hidden charges, refill speed, ingredient freshness, and real consumer reputation, making the generated rubric more intent-aware and actionable.

In addition, the component ablation results and the full rubrics generated by CaRG and Direct-Rubric for the same query are provided in Appendices B.2 and C.

4.3 Online AI Search Evaluation

Because the evaluation set is smaller, we additionally report Qworld-full, the full configuration of Qworld, in the Online AI Search Evaluation. TICK (binary) denotes the original scoring scheme in which each checklist item is scored as 0 or 1, while TICK (0–5) denotes a fine-grained scheme in which each checklist item is scored from 0 to 5. The Online AI Search Evaluation further applies different rubric generation methods to an online AI search capability benchmark, with the goal of comparing the response quality of two online model versions. Specifically, for the same set of online benchmark queries, we collect responses generated by the two model versions and use a human-annotated production rubric to evaluate the two sets of responses, obtaining the golden ranking between the model versions. We then use rubrics or checklists generated by CaRG and other baselines to rescore the same model responses, and compare whether the resulting version-level win/loss decisions agree with the golden ranking.

As shown in Table 2, CaRG achieves the highest agreement among all methods. It ranks first in Accuracy, Kappa, and Spearman ρ , indicat-

Method	Accuracy	Kappa	Pearson r	Spearman ρ
CaRG	68.91%	0.3530	0.4126	0.4445
TICK (binary)	67.83%	0.3330	0.2859	0.4001
TICK (0–5)	66.43%	0.3075	0.4333	0.4286
BSM	66.43%	0.3065	0.2917	0.3461
Qworld-full	65.73%	0.2804	0.3442	0.3537
Direct-Rubric	63.16%	0.2419	0.3892	0.4251
Qworld	58.04%	0.1488	0.2145	0.2481

Table 2: Results of Online AI Search Evaluation. Bold indicates the best value in each metric.

ing that in the real online AI search benchmark for model-version evaluation, CaRG best reproduces the evaluation conclusions produced by the human-annotated production rubric. TICK (0–5) obtains the highest Pearson r , suggesting stronger score-margin correlation, but it remains lower than CaRG in version-level win/loss agreement and ranking consistency. Therefore, from the practical needs of online model-version evaluation, CaRG performs best on the more critical agreement metrics.

Compared with Direct-Rubric, the improvement of CaRG shows that candidate pools not only improve surface-level rubric quality, but also improve alignment with real model-response evaluation. Qworld-full improves over the lightweight Qworld configuration, but the full configuration requires more LLM calls and is costly for large-scale online use. TICK is competitive when used directly as a scoring tool, but together with the Rubric Quality Evaluation, its results suggest that checklist-only outputs are still not suitable as final structured rubrics. BSM is relatively stable, but remains weaker than CaRG in alignment with the human-annotated production-rubric golden ranking.

Overall, the two evaluations provide complementary evidence for CaRG. The experimental results suggest that reusable candidate pools constructed from historical queries and category structure are useful not only for generating better query-level rubrics, but also for supporting practical O2O AI search evaluation.

5 Conclusion

In O2O local-life AI search, user queries often contain concrete consumption needs, category-related constraints, and decision-oriented evaluation requirements. Based on this, we propose CaRG, a category-aware rubric generation framework. CaRG constructs reusable general and category-specific evaluation dimension candidate

pools from historical queries, and when generating rubrics for a target query, injects these candidate pools into the LLM as structured business knowledge, thereby generating query-level rubrics with concrete positive and negative scoring criteria. Blind pairwise LLM meta-evaluation on 600 stratified sampled queries shows that CaRG consistently improves over Direct-Rubric and exhibits more stable overall quality than other query-only baselines. Further evaluation on an online AI search benchmark shows that CaRG is most consistent with the ranking decisions produced by a human-annotated production rubric. These results indicate that historical queries and category structure can provide an effective and scalable source of reusable evaluation knowledge for O2O rubric generation.

Limitations

The current rubric schema is designed for text-based evaluation and cannot directly evaluate multimodal or interactive search results, such as UI layout, map cards, merchant images, navigation buttons, filtering widgets, or visual presentation quality. CaRG uses reliable category labels that have been stably applied in the production system. Therefore, when transferring the framework to other platforms or business domains, the reliability of the category labels must also be ensured to prevent the candidate pools from being contaminated by noise. Meanwhile, if other platforms have a similar structure, they may also leverage their own historical queries and labels to explore CaRG-style candidate-pool construction methods. CaRG focuses on query-specific rubric generation and does not define platform-level evaluation principles by itself; expert-written base rubrics and periodic human audits are still needed for safety, fairness, compliance, and business-policy requirements.

Ethical Considerations

All data used in this paper were accessed and processed by the two student authors during their internships at Meituan. Before use, the data were sufficiently de-identified, used solely for offline research, and could not be used to identify individual users. To comply with the company’s data governance and privacy protection requirements, we do not release raw queries, model responses, or complete production rubrics. For transparency, we

report the method pipeline, prompts, evaluation protocol, and metric computation details in the main text and appendix. Although CaRG does not use user profiles and generates rubrics solely from the input query and category information, LLM-generated rubrics may still exhibit stereotypes or biases and therefore require targeted mitigation.

Writing Assistance

We used GPT-5.5 for English translation and grammar checking during paper writing. All research questions, method design, experimental setup, result analysis, and final manuscript decisions were reviewed and confirmed by the authors.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No. 2024YFF0729003, the Beijing Nova Program under Grant No. 202604841283, and the Fundamental Research Funds for the Central Universities.

References

- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.
- Jonathan Cook, Tim Rocktäschel, Jakob Foerster, Dennis Aumiller, and Alex Wang. 2024. [Ticking all the boxes: Generated checklists improve llm evaluation and generation](#). *arXiv preprint arXiv:2410.03608*.
- Kaustubh Dhole and Eugene Agichtein. 2026. [Rubricrag: Towards interpretable and reliable llm evaluation via domain knowledge retrieval for rubric generation](#). In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '26*, page 36813687, New York, NY, USA. Association for Computing Machinery.
- Zhiyuan Fan, Weinong Wang, Xing Wu, and Debing Zhang. 2024. [Sedareval: Automated evaluation using self-adaptive rubrics](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16916–16930.
- Naghme Farzi and Laura Dietz. 2024. [Pencils down! automatic rubric-based evaluation of retrieve/generate systems](#). In *Proceedings of the 2024 acm sigir international conference on theory of information retrieval*, pages 175–184.

- Jiazhan Feng, Chongyang Tao, Xiubo Geng, Tao Shen, Can Xu, Guodong Long, Dongyan Zhao, and Daxin Jiang. 2024. [Synergistic interplay between search and large language models for information retrieval](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9571–9583.
- Momoka Furuhashi, Kouta Nakayama, Takashi Kodama, and Saku Sugawara. 2025. [Are checklists really useful for automatic evaluation of generative tasks?](#) In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10641–10664.
- Shanghai Gao, Yuchang Su, Pengwei Sui, Curtis Ginder, and Marinka Zitnik. 2026. [Qworld: Question-specific evaluation criteria for llms](#).
- Wei Guo, Jiale Mao, Yiqi Tong, Chuyu Fang, Xiao Zhang, Yikun Ban, Zhaojun Hu, Yiyang Duan, and Fuzhen Zhuang. 2026. [Alignment-free multi-modality large-small model bidirectional collaboration with missing modality](#). In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 1451–1461.
- Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. 2024. [Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts](#). In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 13806–13834.
- Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. [Prometheus: Inducing fine-grained evaluation capability in language models](#). In *International Conference on Learning Representations*, volume 2024, pages 29927–29962.
- Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2024. [Generative judge for evaluating alignment](#). In *International Conference on Learning Representations*, volume 2024, pages 27547–27574.
- Dengcan Liu, Fengkai Yang, Xiaohan Wang, Shurui Yan, Jiajun Chai, Jiahao Li, Yikun Ban, Zhendong Mao, Wei Lin, and Guojun Yin. 2026a. [Cdrmm: Contrast-driven rubric generation for reliable and interpretable reward modeling](#). In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 3152–3163.
- Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. 2026b. [Openrubrics: Towards scalable synthetic rubric generation for reward modeling and llm alignment](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17417–17437.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. [G-eval: Nlg evaluation using gpt-4 with better human alignment](#). In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522.
- Changze Lv, Jie Zhou, Wentao Zhao, Jingwen Xu, Shihan Dou, Zisu Huang, Muzhao Tian, Xiaohua Wang, Yang Liu, Pluto Zhou, Tao Gui, Le Tian, Xiao Zhou, Xiaoqing Zheng, Xuanjing Huang, and Jie Zhou. 2026. [Learning query-specific rubrics from human preferences for deepresearch report generation](#). *arXiv preprint arXiv:2602.03619*.
- Aditya Pathak, Rachit Gandhi, Vaibhav Uttam, Arnav Ramamoorthy, Pratyush Ghosh, Aaryan Raj Jindal, Shreyash Verma, Aditya Mittal, Aashna Ased, Chirag Khatri, Yashwanth Nakka, Devansh, Jagat Sesh Challa, and Dhruv Kumar. 2025. [Rubric is all you need: Improving llm-based code evaluation with question-specific rubrics](#). In *Proceedings of the 2025 ACM Conference on International Computing Education Research V. 1*, pages 181–195.
- Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. 2024. [Branch-solve-merge improves large language model evaluation and generation](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8352–8370.
- Manasi Sharma, Chen Bo Calvin Zhang, Chaithanya Bandi, Clinton Wang, Ankit Aich, Huy Nghiem, Tahseen Rabbani, Ye Htet, Brian Jang, Sumana Basu, Aishwarya Balwani, Denis Peskoff, Marcos Ayestaran, Sean M. Hendryx, Brad Kenstler, and Bing Liu. 2026. [Researchrubrics: A benchmark of prompts and rubrics for evaluating deep research agents](#). In *International Conference on Learning Representations*, volume 2026, pages 90447–90472.
- William F Shen, Xinchu Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. 2026. [Rethinking rubric generation for improving llm judge and reward modeling for open-ended tasks](#).
- Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. 2025. [Judgebench: A benchmark for evaluating llm-based judges](#). In *International Conference on Learning Representations*, volume 2025, pages 63277–63303.
- Manya Wadhwa, Zayne Sprague, Chaitanya Malaviya, Philippe Laban, Junyi Jessy Li, and Greg Durrett. 2025. [Evalagent: Discovering implicit evaluation criteria from the web](#). *arXiv preprint arXiv:2504.15219*.
- Yancheng Wang, Ziyang Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu,

- Xiaojiang Huang, and Yingzhen Yang. 2024a. [Recmind: Large language model powered agent for recommendation](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364.
- Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024b. [Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization](#). In *International Conference on Learning Representations*, volume 2024, pages 43573–43593.
- Tianjun Wei, Wei Wen, Ruizhi Qiao, Xing Sun, and Jianghong Ma. 2025. [Rocketeval: Efficient automated llm evaluation via grading checklist](#). In *International Conference on Learning Representations*, volume 2025, pages 58593–58619.
- Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. [Evaluating large language models at evaluating instruction following](#). In *International Conference on Learning Representations*, volume 2024, pages 40193–40219.
- Fuwei Zhang, Xiaoyu Liu, Xinyu Jia, Yingfei Zhang, Zenghua Xia, Fei Jiang, Fuzhen Zhuang, Wei Lin, and Zhao Zhang. 2025. [Hiergr: Hierarchical semantic representation enhancement for generative retrieval in food delivery search](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 444–455.
- Fuwei Zhang, Xiaoyu Liu, Dongbo Xi, Jishen Yin, Huan Chen, Peng Yan, Fuzhen Zhuang, and Zhao Zhang. 2026a. [Multi-aspect cross-modal quantization for generative recommendation](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 16271–16279.
- Fuwei Zhang, Yanzhao Zhang, Mingxin Li, Dingkun Long, Lexiang Hu, Pengjun Xie, Zhao Zhang, and Fuzhen Zhuang. 2026b. [Core-bench: A comprehensive benchmark for code retrieval in the era of agentic coding](#).
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). *Advances in neural information processing systems*, 36:46595–46623.
- Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2025. [Judgelm: Fine-tuned large language models are scalable judges](#). In *International Conference on Learning Representations*, volume 2025, pages 51257–51296.

A Implementation Details

A.1 Data Construction

This subsection describes how the data used in candidate-pool construction and the two experiments is constructed. For candidate-pool construction, we start from 53,934 processed historical queries. We sample 3,177 representative queries for candidate-dimension induction. During sampling, queries are first bucketed by the combination of first-level and second-level categories, and systematic interval sampling is then performed within each bucket to cover different business categories and long-tail scenarios.

For the Rubric Quality Evaluation, we construct a 600-query standard evaluation set from the 3,177 representative queries. The 600 queries are sampled proportionally by first-level category with random seed 42. This evaluation set is used for blind pairwise LLM meta-evaluation of rubric quality. For each query, each method generates a rubric or checklist, and the generated evaluation standard is compared against Direct-Rubric under the same query.

For the Online AI Search Evaluation, the data is constructed from responses produced by two online model versions on the same online AI search benchmark. The two online models directly answer the same set of benchmark queries. We then use a human-annotated production rubric to score the two model responses for each query and obtain the golden ranking between the two online model versions. Each automatic method is then used to generate a rubric or checklist for the same query, score the two online model responses, and compare its win/loss decision with the human-annotated production-rubric golden ranking.

The parameter settings, metric computation details, and prompt templates used in the experiments are summarized in Appendices A.4, A.5, and D.

A.2 Candidate-Pool Implementation

This subsection summarizes the implementation details of candidate-pool construction used by CaRG. The overall procedure is described in Section 3; here we focus on the concrete settings used in the experiments.

In the initial dimension generation stage, we use Gemini-3.1-pro-preview. For each sampled query, the model generates both general dimensions and category-specific dimensions. We then use a lo-

cally deployed bge-m3 model to extract embeddings for the corresponding dimension definitions. Specifically, for each dimension, we take the CLS token representation from the last hidden layer of bge-m3 as the dimension embedding and apply L2 normalization.

After obtaining the embeddings of all dimensions, we perform hierarchical clustering. The distance metric is cosine distance, the linkage criterion is average linkage, and the distance threshold is set to 0.15. After obtaining the clusters, we select the dimension closest to the cluster centroid by cosine similarity as the representative of that cluster.

After clustering, we use Gemini-3.1-pro-preview for two rounds of LLM consolidation to deduplicate and merge dimensions. In the first round, every 100 representative dimensions are processed as one batch for within-batch consolidation. In the second round, the first-round outputs are consolidated globally, producing the final general candidate pool and category-specific candidate pools. During final query-level rubric generation, the model outputs 3 to 8 dimensions for each query, and each dimension consists of a dimension name, a positive criterion, and a negative criterion.

A.3 Baseline Details

- **Direct-Rubric.** Direct-Rubric is the most direct baseline. It only takes the query as input and directly generates a rubric through a single LLM call. Its output format is consistent with CaRG: it generates 3 to 8 rubric dimensions, and each dimension contains a dimension name, positive scoring criterion, and negative scoring criterion. Direct-Rubric can be viewed as an ablation-style comparison for CaRG, since the only difference is whether candidate pools are introduced.
- **TICK.** TICK generates checklist questions for each query as the rubric result. In the Online AI Search Evaluation, we use two scoring schemes for TICK. The first is binary checklist scoring, which follows the original paper: each checklist item receives one point if the response satisfies it, and zero otherwise. The second is 0–5 fine-grained checklist scoring: each checklist item is scored from 0 to 5 according to its degree of satisfaction, then all checklist-item scores are

summed and divided by the maximum possible checklist score under the current query to obtain the normalized final score.

- **BSM.** BSM adapts the branch-solve-merge framework to O2O rubric generation. It contains three stages: the Branch stage decomposes evaluation directions, the Solve stage generates a rubric item for each branch, and the Merge stage merges and deduplicates candidate dimensions while preserving the MECE structure as much as possible. In the Rubric Quality Evaluation, BSM is run under the four selected generation models; in the Online AI Search Evaluation, Gemini-3.1-pro-preview is used for all three stages.
- **Qworld.** Qworld is adapted from the open-source Qworld implementation. Starting from a query, it sequentially generates scenarios, perspectives, and criteria, and then performs semantic deduplication. In the Rubric Quality Evaluation, to control cost, we use a lightweight Qworld configuration: it generates only one initial scenario and one initial perspective, performs no additional scenario or perspective expansion, performs only one round of criteria expansion, and sets the semantic deduplication threshold to 0.8. In the Online AI Search Evaluation, because the number of queries is smaller, we additionally introduce Qworld-full, which uses the full tree-expansion configuration. Based on the generated results, Qworld-full requires roughly 10 to 20 LLM calls per query on average, and is therefore unsuitable for large-scale online use.

A.4 Parameter Settings

For candidate-dimension induction, CaRG uses Gemini-3.1-pro-preview with temperature 0.6 and maximum output length 4096. For embedding extraction, bge-m3 uses maximum input length 512 and batch size 64. For LLM consolidation after clustering, we use Gemini-3.1-pro-preview with temperature 0.3 and maximum output length 16,384. For final rubric generation, CaRG and Direct-Rubric use temperature 0.6 and maximum output length 4096. The other rubric-generation baselines use the same generation setting as Direct-Rubric unless otherwise stated. In the Rubric Quality Evaluation, the rubric-pair judge uses temperature 0.0 and maximum output length 8192. In

the Online AI Search Evaluation, the response-scoring judge uses temperature 0.0 and maximum output length 4096.

A.5 Metric Computation

In the Rubric Quality Evaluation, the judge scores each rubric on #Intent, #Action, and #MECE using a 1–9 scale. The total score is computed as

$$\begin{aligned} \text{Total} = & 0.3 \cdot \# \text{Intent} + 0.4 \cdot \# \text{Action} \\ & + 0.3 \cdot \# \text{MECE}. \end{aligned} \quad (6)$$

The pairwise decision is determined by the total scores after the anonymized A/B positions are mapped back to the true method identities.

In the Online AI Search Evaluation, we compare each automatic method with the human-annotated production-rubric golden ranking. Accuracy is the fraction of queries whose win/loss/tie label exactly matches the golden label. Cohen’s Kappa is computed over the three labels {A, B, Tie} to measure agreement after removing chance agreement. For score-margin correlation, the judge-side score is first normalized by the maximum possible score under the current rubric or checklist, and the predicted margin is computed as the normalized score of Model A minus that of Model B. Pearson’s r measures linear correlation between predicted and golden score margins, while Spearman’s ρ measures rank correlation after converting margins into ranks.

B Additional Experiment Results

B.1 Split Robustness and Data Leakage Discussion

The original 600-query Rubric Quality Evaluation set is sampled from the 3,177 representative queries used for candidate-dimension induction, which may raise concerns about data leakage. We first clarify that Stage I of CaRG does not retain directly reusable query-specific rubrics. Specifically, in the initial dimension generation stage, the LLM generates both general dimensions and category-specific dimensions for each sampled query, and each dimension contains both a generalized definition and a query-specific application. However, during candidate-pool construction, we only retain the generalized definition, with the goal of preserving transferable evaluation concepts rather than query-specific examples. These generalized definitions are then processed through embedding-based clustering, repre-

sentative selection, and two rounds of LLM-based deduplication and consolidation. The final candidate pools therefore contain more abstract general and category-level evaluation dimensions, which serve as inspiration for query-level rubric generation rather than final query-specific rubrics.

In earlier trials, we also tried directly selecting dimensions from the candidate pools during Stage II to generate rubrics, but this brought limited gains. Through case analysis, we found that dimensions in both the general pool and the category-specific pool are still relatively generic and cannot directly satisfy fine-grained query-level evaluation needs. Therefore, the gains of CaRG do not come from retaining or reusing query-specific rubrics or query-specific examples.

Nevertheless, we agree that the original data split may still raise concerns. To further address this concern, we additionally sampled 600 queries from the original 53,934 historical queries. Only 41 of them overlap with the candidate-pool construction set, accounting for 6.8%, while the remaining 559 queries are not used in candidate-pool construction. With gpt-4.1 fixed as the pairwise judge, CaRG still clearly outperforms Direct-Rubric under gpt-4o-mini, gpt-4.1, and Gemini-3.1-pro-preview generators, with win rates of 60.5%, 79.7%, and 84.0%, respectively. These results remain consistent with the original trend, indicating that CaRG’s gains are not driven by overlap between evaluation queries and candidate-pool construction data.

B.2 Component Ablation

To analyze the respective contributions of the general candidate pool and the category-specific candidate pool, we further conduct a component-level ablation on the newly sampled 600-query set. Specifically, we compare CaRG without the common/general pool, CaRG without the category-specific pool, and full CaRG. In this experiment, both the rubric generation model and the pairwise judge model are fixed as gpt-4.1.

Setting	Direct Win	Method Win	Tie
w/o Common Pool	24.5%	72.8%	2.7%
w/o Category-Specific Pool	21.7%	75.7%	2.7%
Full CaRG	18.8%	79.7%	1.5%

Table 3: Component ablation on the newly sampled 600-query set. Both the rubric generation model and pairwise judge model are fixed as gpt-4.1.

Table 3 shows that removing either the com-

mon/general pool or the category-specific pool still outperforms Direct-Rubric, while full CaRG achieves the highest win rate. Full CaRG improves over w/o Common Pool by 6.9 percentage points and over w/o Category-Specific Pool by 4.0 percentage points. These results indicate that both pools contribute useful reusable knowledge, and that combining general and category-specific candidate dimensions brings additional gains.

B.3 Cross-Judge Robustness

Table 4 reports the cross-judge robustness results of CaRG in the Rubric Quality Evaluation, with Gemini-3.1-pro-preview fixed as the generation model. CaRG remains preferred over Direct-Rubric across all judge models.

Judge	Direct Win	CaRG Win	Tie	Direct Total	CaRG Total	Δ Total
GPT-4.1	17.0%	80.7%	2.3%	6.891	7.978	+1.087
DeepSeek-v4-flash	21.3%	77.8%	0.8%	6.871	7.763	+0.892
GLM-4.6	24.7%	75.0%	0.3%	7.377	8.190	+0.813
GPT-4o-mini	18.0%	49.3%	32.7%	6.245	7.092	+0.848

Table 4: Cross-judge robustness in the Rubric Quality Evaluation. Δ Total denotes CaRG minus Direct-Rubric.

C Rubric Examples

This section provides an example of rubrics generated by CaRG and Direct-Rubric for the same query.

```
Query: nearby self-service hotpot buffet
First-level category: Food

CaRG output:
{
  "intent_analysis": "The user wants to find nearby buffet
    ↳ restaurants centered on hotpot. The core requirements
    ↳ are proximity and a self-service dining mode. Potential
    ↳ pain points include hidden charges common in buffets,
    ↳ such as separately charged soup bases, stale
    ↳ ingredients, or slow dish refills. The user is highly
    ↳ sensitive to price, dish variety, and authentic
    ↳ reputation.",
  "rubric_dimensions": [
    {
      "dimension_name": "Intent Precision: Category and
        ↳ Service-Mode Matching",
      "good_response_criteria": "The recommended merchants
        ↳ must strictly satisfy both the hotpot and
        ↳ self-service conditions, such as self-service
        ↳ rotating hotpot, hotpot-and-barbecue buffet, or
        ↳ seafood hotpot buffet, and the text should
        ↳ explicitly state the self-service attribute.",
      "bad_response_criteria": "The response deviates from
        ↳ the core intent, for example recommending ordinary
        ↳ a la carte hotpot restaurants such as Haidilao or
        ↳ Banu, or recommending pure barbecue buffets, pure
        ↳ Japanese buffets, or other restaurants that do not
        ↳ include hotpot."
    },
    {
      "dimension_name": "LBS Awareness and Location
        ↳ Description",
      "good_response_criteria": "The textual answer should
        ↳ respond to the nearby constraint and provide
        ↳ concrete location references, such as business
        ↳ district, street, landmark, or mall floor, so that
        ↳ the user can roughly perceive the distance and
        ↳ direction of the store from the text alone.",
```

```
      "bad_response_criteria": "The response completely lacks
        ↳ location descriptions, only lists national chain
        ↳ brands without pointing to specific stores, or
        ↳ contains obvious address hallucinations, such as
        ↳ describing a store in City A as being in City B."
    },
  ],
  "dimension_name": "Price Transparency and Buffet Rules",
  "good_response_criteria": "The response should clearly
    ↳ provide an estimated per-person price or group-buy
    ↳ package price, and proactively explain key buffet
    ↳ rules, such as dining time limits, whether a
    ↳ clean-plate deposit is charged, and whether soup
    ↳ bases, condiments, or drinks require extra fees.",
  "bad_response_criteria": "The response lacks key price
    ↳ information, or says nothing about common hidden
    ↳ costs in buffets, such as separately charged soup
    ↳ bases or limited portions of premium seafood, which
    ↳ may mislead users into incorrect price
    ↳ expectations."
},
{
  "dimension_name": "Dish Supply and Feature
    ↳ Presentation",
  "good_response_criteria": "The response should
    ↳ specifically list core high-value ingredients,
    ↳ such as freshly sliced beef, live shrimp, or beef
    ↳ tripe, or distinctive soup bases, and state whether
    ↳ additional categories such as barbecue, desserts,
    ↳ or drinks are included, helping users concretely
    ↳ judge whether the buffet is worth the price.",
  "bad_response_criteria": "The response only stacks
    ↳ generic phrases such as rich dishes, good taste, or
    ↳ everything is available, without any concrete
    ↳ ingredient or flavor details."
},
{
  "dimension_name": "Reputation Reference and Pitfall
    ↳ Avoidance",
  "good_response_criteria": "The response should
    ↳ objectively summarize real consumer feedback, not
    ↳ only extracting key selling points but also
    ↳ providing practical pitfall-avoidance tips or
    ↳ expectation management, such as long queues at meal
    ↳ times, slow refills for a popular meat item, or a
    ↳ somewhat crowded environment.",
  "bad_response_criteria": "The response reads like
    ↳ merchant marketing copy, exaggerates excessively
    ↳ without any flaws, or fabricates false reviews that
    ↳ do not match the actual store situation."
},
]
}

Direct-Rubric output:
{
  "intent_analysis": "The user is looking for a self-service
    ↳ hotpot restaurant near the current location. The core
    ↳ needs are immediate dining and high cost performance;
    ↳ the pain points are being recommended a la carte hotpot
    ↳ restaurants, non-hotpot buffets, or stores that are too
    ↳ far away.",
  "rubric_dimensions": [
    {
      "dimension_name": "Category and Location Matching",
      "good_response_criteria": "The response should
        ↳ accurately recommend restaurants that have both
        ↳ self-service and hotpot attributes, such as
        ↳ self-service rotating small hotpot, wagyu sukiyaki
        ↳ buffet, or seafood hotpot buffet, and clearly
        ↳ reflect the distance concept of nearby,
        ↳ surrounding, or same business district in the
        ↳ text.",
      "bad_response_criteria": "The response has category
        ↳ drift, such as recommending a la carte hotpot
        ↳ restaurants like Haidilao, or only recommending
        ↳ barbecue buffets or ordinary hotel buffets without
        ↳ hotpot; alternatively, it ignores the nearby intent
        ↳ and recommends well-known stores across districts
        ↳ or cities."
    },
    {
      "dimension_name": "Completeness of Decision
        ↳ Information",
```

```

    "good_response_criteria": "For the recommended
    ↪ self-service hotpot restaurants, the response
    ↪ should provide key consumption decision
    ↪ information, including but not limited to estimated
    ↪ per-person price, main features such as unlimited
    ↪ wagyu beef, unlimited beef tripe, or
    ↪ budget-friendly student options, and a specific
    ↪ business district or street address.",
    "bad_response_criteria": "The response lacks
    ↪ information or stacks empty phrases, only listing
    ↪ store names without price or feature descriptions,
    ↪ or using generic statements such as this store is
    ↪ delicious or the environment is nice."
  },
  {
    "dimension_name": "Entity Authenticity
    ↪ (Anti-Hallucination)",
    "good_response_criteria": "The recommended store names
    ↪ and brands should conform to real-world restaurant
    ↪ knowledge, such as common chain self-service hotpot
    ↪ brands or reasonable local merchant names.",
    "bad_response_criteria": "The model shows serious
    ↪ hallucination by fabricating nonexistent stores,
    ↪ such as Nearby Self-Service Hotpot No. 1 or
    ↪ Selected Self-Service Hotpot Buffet, or forcibly
    ↪ describing well-known pure a la carte hotpot brands
    ↪ as buffets."
  },
  {
    "dimension_name": "Text Structure and Readability",
    "good_response_criteria": "The response should use a
    ↪ structured list or segmented format, such as store
    ↪ name - price - features, and provide 3 to 5 options
    ↪ with different price ranges or features for quick
    ↪ scanning and horizontal comparison.",
    "bad_response_criteria": "The response mixes
    ↪ information about multiple stores into one long
    ↪ paragraph, lacking clear formatting and
    ↪ punctuation separation, making it difficult for
    ↪ users to quickly extract useful options."
  }
]
}

```

decision. This illustrates how candidate-pool augmentation helps CaRG generate rubrics that better capture implicit local-life concerns and produce more executable scoring criteria.

In this example, both methods identify the core intent of finding nearby self-service hotpot, but CaRG produces a more O2O-specific and actionable rubric. First, CaRG separates category matching from LBS-aware location description, making it possible to evaluate whether a response both matches the hotpot-buffet intent and provides usable local-position information. Second, CaRG introduces price transparency and buffet rules as an independent dimension, explicitly covering common local-life risks such as separately charged soup bases, extra condiment or drink fees, dining time limits, and other hidden costs. Third, CaRG makes the consumption experience more concrete by requiring details about dish supply, high-value ingredients, and refill-related expectations, rather than only asking for generic decision information. Finally, CaRG includes reputation reference and pitfall avoidance, encouraging the evaluator to check whether the response provides realistic consumer feedback and practical warnings. By contrast, Direct-Rubric covers the basic intent and response format, but some of its dimensions, such as entity authenticity and text readability, are more general and less tied to the user's actual dining

D Prompt Templates

This section directly lists the prompts used in the implementation.

D.1 Candidate-Dimension Induction Prompt

The following is the translated English version.

```
build_data stage prompt: generation for the 3,177 sampled
↳ queries
system = ""
# Role
You are a senior local-life search algorithm expert and
↳ consumer behavior analyst.

# Background
We are building a platform-wide "search evaluation dimension
↳ standard dictionary (Schema)" for a local-life search
↳ platform.
To ensure that the dictionary has both "high generalization and
↳ reusability" and "real business-guidance value",
you need to deeply analyze the input Query and strictly follow
↳ the output requirements.

# Task
I will provide a real [User Query] and the [first-level
↳ business category] that the Query belongs to.
Please extract 3-5 core evaluation dimensions and strictly
↳ divide them into "general dimensions" and "specific
↳ dimensions".
For each dimension, you must output both [generalized
↳ definition] and [query-specific scoring criteria for the
↳ current Query].

# Guidelines
1. Strict classification and naming:
  - General dimension: [General] - Dimension name
  - Specific dimension: [[current first-level category]] -
    ↳ Dimension name

2. Strict separation between generalization and concretization
↳ (extremely important!):
  - general_definition: a highly abstract dictionary-style
    ↳ explanation; it must never include concrete words from
    ↳ the current Query
  - query_specific_application: tightly follow the Query
    ↳ context and include concrete dishes/places/scenarios
    ↳ when writing scoring criteria""
user = ""
Current first-level category: {category}
User Query: {query}
""
```

D.2 Candidate-Pool Consolidation Prompt

The following is the translated English version.

```
Pool construction stage after embedding:
SYSTEM_PROMPT = (
  "You are a top-level local-life platform data architect and
  ↳ taxonomy expert."
  "You are good at taking a global perspective to decouple,
  ↳ deduplicate, and reorganize redundant label systems,"
  "and ultimately build a Golden Schema that satisfies the
  ↳ MECE principle (mutually exclusive and collectively
  ↳ exhaustive)."
```

```
)
USER_PROMPT_TEMPLATE = ""
# Background
I provide you with a set of [representative evaluation
↳ dimensions]. These dimensions are "cluster
↳ representatives" extracted from different data clusters in
↳ the previous step.
However, from a global perspective, there are still serious
↳ problems among these representative dimensions, such as
↳ "semantic overlap", "hypernym-hyponym inclusion", and
↳ "concept stitching (one dimension mixes two unrelated physical
↳ concepts)".

# Task
Please perform global deduplication, decoupling, and final
↳ merging on this batch of input dimensions, and output an
↳ absolutely clean, mutually exclusive, and atomic core
↳ dimension dictionary.
```

```
# Guidelines
1. Strong deduplication and merging: dimensions with the same
↳ core meaning must be forcefully merged; for inclusion
↳ relations, keep the higher-level dimension
2. Decouple and split stitched monsters (extremely important!):
↳ dimensions that contain two independent variables at the
↳ same time must be split
3. Strictly prevent over-merging: maintain the "atomicity" of
↳ dimensions and do not squeeze independent important
↳ dimensions together
4. Write final definitions: use the naming format "[{prefix}] -
↳ Dimension name", and use sentence patterns such as
↳ "evaluate whether the merchant..."

# Output Format (strict JSON)
{
  "global_analysis": "About 100 Chinese characters, describing
  ↳ the overlap/stitching problems found",
  "final_dimensions": [
    {
      "standard_name": "[{prefix}] - Final standard dimension
      ↳ name",
      "standard_definition": "Highly inclusive and mutually
      ↳ exclusive objective dictionary-style definition",
      "merged_source_ids": [1, 5, 8]
    }
  ],
  "discarded_ids": []
}
All input IDs must and can only appear in merged_source_ids or
↳ discarded_ids; none may be omitted!

# Input
Current category prefix being processed: [{prefix}]
Representative dimension list to be deduplicated and decoupled:
{candidates_json_with_ids}
""
```

D.3 CaRG Rubric Generation Prompt

The following is the translated English version.

```
Rubric-generation prompt for CaRG
SYSTEM_PROMPT = "You are a senior local-life search quality
↳ evaluation expert. Your core task is to create a
↳ professional scoring rubric for a specific user search term
↳ (Query), used to evaluate the quality of the [pure-text]
↳ final answer/result page given by a large language model or
↳ search system."
USER_TEMPLATE = ""\n
# Background
```

To help you build the most professional scoring rubric that
↳ best fits the business scenario, I provide you with two
↳ sets of [evaluation-dimension candidate pools] as your
↳ "inspiration knowledge base":

1. **[General candidate pool] (foundation of basic
↳ experience)**:
 - **Meaning**:: represents the general basic concerns and
↳ experience bottom lines when users search across the
↳ local-life platform.
 - **Function**:: covers dimensions such as LBS
↳ spatial-distance matching, business-status accuracy,
↳ intent semantic alignment, and review authenticity,
↳ ensuring that the recommendations given by the large
↳ model are "accurate to find, do not lead to wasted
↳ trips, and contain truthful information".
2. **[Category-specific candidate pool] (vertical business
↳ depth)**:
 - **Meaning**:: represents the specific business pain points
↳ and consumer psychology under the particular vertical
↳ category to which the current Query belongs.
 - **Function**:: for example, "taste authenticity/pricing
↳ transparency" in the food category, and "hygiene
↳ conditions/facility completeness" in the hotel category,
↳ ensuring that the recommendations given by the large
↳ model "understand category rules, avoid hidden pitfalls,
↳ and have depth".

These two candidate pools provide you with a panoramic map of
↳ evaluation directions and serve as the inspiration source
↳ and reference baseline for creating the final scoring
↳ criteria.

Task

Please strictly follow the steps below to create the scoring
→ rubric:

****Step 1: Deep intent decomposition and scenario inference****
- Analyze the explicit needs of the [User Query] (what the user
→ literally wants to find) and the implicit needs (what
→ potential concerns or specific preferences lie behind it).
- Infer the user's current consumption scenario (e.g.,
→ immediate craving, weekend gathering, or out-of-town
→ travel check-in), and identify whether this Query has
→ error-prone points (e.g., typos, easily confused imitation
→ brands, etc.).

****Step 2: Extract and reorganize core evaluation dimensions****
- Based on the intent analysis in Step 1, examine the two
→ [candidate pools] I provide.
- Extract ****3-8**** core evaluation dimensions that determine
→ whether the answer to this Query is good or bad.
- ****Operational guidance****: you may directly select dimensions
→ from the candidate pools that are highly suitable; you may
→ also slightly adjust dimensions from the candidate pools to
→ better fit the current Query; if the Query is extremely
→ special, you may even go beyond the candidate pools and
→ summarize new evaluation dimensions yourself.

****Step 3: Create concrete positive and negative scoring
→ criteria****
- For each dimension you extract, write highly targeted
→ positive (high-score) and negative (deduction) evaluation
→ criteria.
- The high-score criteria should describe how the large model
→ should perfectly solve the user's pain points through text;
→ the deduction criteria should precisely anticipate
→ mistakes the large model is likely to make (such as rigid
→ word splitting, common-sense errors, empty verbose content,
→ etc.).

Evaluation Guidelines

1. ****Use the candidate pools as flexible inspiration****: the
→ candidate pools provide inspiration and reference for
→ evaluation directions. You can directly use dimension
→ names from the candidate pools, or you can independently
→ extract, modify, or combine more suitable dimension names
→ according to the special context of the current Query,
→ without being rigidly constrained by the candidate pools.
2. ****The evaluation object is pure text****: the criteria you
→ create must be directly judgeable by evaluators only by
→ reading the text generated by the large model. You cannot
→ require evaluators to verify offline real-time status (such
→ as current queue length or whether a dish is sold out), and
→ you cannot require the large model to have frontend UI
→ capabilities (such as map cards or one-click navigation).
3. ****Dimensions must not overlap****: the selected or extracted
→ dimensions must each have their own responsibility
→ (satisfying the MECE principle); the same error can only be
→ penalized under one dimension, and logical overlap is never
→ allowed.
4. ****Scoring criteria must be concrete****: good criteria should
→ tightly follow the Query context, and bad criteria should
→ point out mistakes the large model is genuinely likely to
→ make (such as answering off-topic, hallucinating addresses,
→ or lacking key decision-making information). Reject
→ generic empty statements that apply everywhere.

Output Format (strictly output JSON; do not output any
→ Markdown markers or extra text)

```
{{
  "intent_analysis": "Briefly describe the user's core need and
  → potential pain points (within 50 Chinese characters).",
  "rubric_dimensions": [
    {{
      "dimension_name": "Selected or self-extracted dimension
      → name",
      "good_response_criteria": "[Tightly follow the Query]
      → Specific high-score pure-text answer performance...",
      "bad_response_criteria": "[Tightly follow the Query]
      → Specific deduction pure-text answer performance..."
    }}
  ]
}}
```

Input

User Query: [{query}]

[General candidate pool] (covers basic evaluation directions,
→ for inspiration and reference):
{common_pool_json}

[Category-specific candidate pool] (covers evaluation
→ directions specific to the [{category}] category, for
→ inspiration and reference):
{category_pool_json}"""

D.4 Direct-Rubric Prompt

The following is the translated English version.

rubric_direct prompt:
SYSTEM_PROMPT = "You are a senior local-life search quality
→ evaluation expert. Your core task is to create a
→ professional scoring rubric for a specific user search term
→ (Query), used to evaluate the quality of the [pure-text]
→ final answer/result page given by a large language model or
→ search system."
USER_TEMPLATE = """\n
Task

Please strictly follow the steps below to create the scoring
→ rubric:

****Step 1: Deep intent decomposition and scenario inference****
- Analyze the explicit needs of the [User Query] (what the user
→ literally wants to find) and the implicit needs (what
→ potential concerns or specific preferences lie behind it).
- Infer the user's current consumption scenario (e.g.,
→ immediate craving, weekend gathering, or out-of-town
→ travel check-in), and identify whether this Query has
→ error-prone points (e.g., typos, easily confused imitation
→ brands, etc.).

****Step 2: Extract core evaluation dimensions****
- Based on the intent analysis in Step 1, use your local-life
→ business knowledge to brainstorm and extract ****3-8**** core
→ evaluation dimensions that determine whether the answer to
→ this Query is good or bad.

****Step 3: Create concrete positive and negative scoring
→ criteria****
- For each dimension you extract, write highly targeted
→ positive (high-score) and negative (deduction) evaluation
→ criteria.
- The high-score criteria should describe how the large model
→ should perfectly solve the user's pain points through text;
→ the deduction criteria should precisely anticipate
→ mistakes the large model is likely to make (such as rigid
→ word splitting, common-sense errors, empty verbose content,
→ etc.).

Evaluation Guidelines

1. ****The evaluation object is pure text****: the criteria you
→ create must be directly judgeable by evaluators only by
→ reading the text generated by the large model. You cannot
→ require evaluators to verify offline real-time status (such
→ as current queue length or whether a dish is sold out), and
→ you cannot require the large model to have frontend UI
→ capabilities (such as map cards or one-click navigation).

2. ****Dimensions must not overlap****: the extracted dimensions
→ must each have their own responsibility (satisfying the
→ MECE principle); the same error can only be penalized under
→ one dimension, and logical overlap is never allowed.

3. ****Scoring criteria must be concrete****: good criteria should
→ tightly follow the Query context, and bad criteria should
→ point out mistakes the large model is genuinely likely to
→ make (such as answering off-topic, hallucinating addresses,
→ or lacking key decision-making information). Reject
→ generic empty statements that apply everywhere.

Output Format (strictly output JSON; do not output any
→ Markdown markers or extra text)

```
{{
  "intent_analysis": "Briefly describe the user's core need and
  → potential pain points (within 50 Chinese characters).",
```

```

"rubric_dimensions": [
  {{
    "dimension_name": "Extracted dimension name",
    "good_response_criteria": "[Tightly follow the Query]
    ↳ Specific high-score pure-text answer performance...",
    "bad_response_criteria": "[Tightly follow the Query]
    ↳ Specific deduction pure-text answer performance..."
  }}
]
}}

# Input

User Query: [{query}]
Category: [{category}]""

```

D.5 BSM Prompt

The following is the translated English version.

Rubric-generation prompt for BSM:
BSM_Q_BRANCH_SYSTEM_PROMPT = ""

You are a senior local-life search quality evaluation expert.
↳ Your task is to decompose the broad goal of "how to
evaluate whether a pure-text answer satisfies this need"
↳ into several independent and concrete evaluation branches
↳ according to the user Query.

""

BSM_Q_BRANCH_USER_TEMPLATE = ""

Task

Please deeply analyze the given user Query and decompose it
↳ into a set of Query-specific evaluation branches.

These branches will serve as the skeleton for subsequently
↳ generating the formal scoring rubric.

Requirements

1. ****Independence and mutual exclusiveness****: each Branch must
↳ correspond to an independent evaluation dimension, and
↳ branches should have as little semantic overlap as
↳ possible.
2. ****Tightly follow the context****: each Branch must tightly
↳ follow the explicit needs and potential pain points of the
↳ current Query. It is strictly forbidden to generate
↳ universally applicable empty branches such as "relevance",
↳ "accuracy", or "whether it is helpful".
3. ****Pure-text constraint****: do not include requirements that
↳ cannot be judged through a "pure-text answer" (such as UI
↳ cards, frontend buttons, images, map interactions, offline
↳ real-time queue status, etc.).
4. ****Quantity control****: decompose into 3-8 core Branches.

Output Format (strictly output JSON; do not output any
↳ Markdown markers)

```

{{
  "branches": [
    {{
      "id": "B1",
      "name": "Short Chinese dimension name",
      "description": "Explain in detail what this branch
      ↳ specifically evaluates, and it must be combined with
      ↳ the concrete context of the current Query."
    }}
  ]
}}

# Input

```

User Query: [{query}]

""

BSM_Q_SOLVE_SYSTEM_PROMPT = ""

You are a search-evaluation Rubric generation expert. Your task
↳ is to generate concrete and implementable scoring criteria
↳ for each branch according to the user Query and the
↳ previously decomposed evaluation branches.

""

BSM_Q_SOLVE_USER_TEMPLATE = ""

Task

I will provide a user Query and a set of evaluation branches.

Please generate one formal Rubric Item for each Branch, used to
↳ evaluate the quality of the [pure-text answer] generated by
↳ a large model for this Query.

Requirements

1. ****Faithfulness to source****: each Rubric Item must originate
↳ from a certain Branch and must not freely diverge away from
↳ the Branch.
2. ****Concrete criteria****: high-score/deduction criteria must
↳ be extremely concrete and combined with the specific
↳ context of the current Query. It is strictly forbidden to
↳ use empty terms such as "relevant" or "accurate" alone.
3. ****Pure-text constraint****: the criteria must be judgeable by
↳ evaluators only by reading pure text, and must never
↳ require verifying offline real-time status or frontend UI.
4. ****Prevent overlap****: try to ensure that Items generated for
↳ different Branches evaluate different types of errors.

Output Format (strictly output JSON; do not output any
↳ Markdown markers)

```

{{
  "candidate_rubrics": [
    {{
      "dimension_name": "Dimension name extracted based on the
      ↳ Branch",
      "good_response_criteria": "[Tightly follow the Query]
      ↳ What kind of pure-text answer performs well on this
      ↳ dimension...",
      "bad_response_criteria": "[Tightly follow the Query] What
      ↳ kind of pure-text answer performs poorly on this
      ↳ dimension (explicit deduction points)...",
      "source_branch_ids": ["B1"]
    }}
  ]
}}

```

Input

User Query: [{query}]

Evaluation Branches:

{branches_json}

""

BSM_Q_MERGE_SYSTEM_PROMPT = ""

You are a top-level search-evaluation Rubric merging and review
↳ expert.

Your core task is to merge, deduplicate, and compress the
↳ initially generated Candidate Rubrics, and output the
↳ final scoring rubric that conforms to the strict format.

```

"""
BSM_Q_MERGE_USER_TEMPLATE = """

# Task

I will provide a user Query and a set of candidate scoring
↪ criteria generated from different branches.

Please globally review, merge, and compress these candidate
↪ criteria to generate the final Rubric.

# Core Constraints

1. **MECE principle (deduplication and merging)**: this is the
↪ most important task! Carefully inspect all candidate
↪ criteria; if semantically duplicated or highly similar
↪ dimensions are found, they must be merged into one
↪ dimension. It is never allowed for the same error to be
↪ penalized repeatedly in the final Rubric.

2. **Remove empty statements**: decisively delete invalid
↪ criteria that are overly generic, not implementable, or
↪ violate the "pure-text evaluation" principle.

3. **Preserve the core**: do not delete very important explicit
↪ constraints and business pain points in the Query.

4. **Quantity control**: after merging and compression, the
↪ number of final Rubric dimensions must be strictly
↪ controlled between **3-8**.

# Output Format (strictly output JSON; must exactly match the
↪ structure below; do not output extra text)

{{
  "intent_analysis": "Based on the candidate criteria,
↪ comprehensively and briefly describe the user's core
↪ need and potential pain points (within 50 Chinese
↪ characters).",
  "rubric_dimensions": [
    {{
      "dimension_name": "Final merged dimension name",
      "good_response_criteria": "[Tightly follow the Query]
↪ Specific high-score pure-text answer performance...",
      "bad_response_criteria": "[Tightly follow the Query]
↪ Specific deduction pure-text answer performance...",
      "source_branch_ids": ["B1", "B2"]
    }}
  ]
}}

# Input

User Query: [{query}]

Candidate Rubrics:

{candidate_rubrics_json}
"""

```

D.6 TICK Prompt

The following is the translated English version.

Rubric-generation prompt for TICK:
TICK_Q_CHECKLIST_SYSTEM_PROMPT = """

You are a rigorous search quality evaluation standard design
↪ expert. Your task is to generate a set of Checklist
↪ Questions based on the user Query, used to evaluate the
↪ quality of [pure-text answers] from large language
↪ models/intelligent search systems.

These Checklist Questions will later be converted into a formal
↪ scoring rubric, so they must be specific, objectively
↪ judgeable, and extremely aligned with the context of the
↪ current Query.

```

"""
TICK_Q_CHECKLIST_USER_TEMPLATE = """

```

Task

Please generate a set of Checklist Questions based on the given
↪ user Query, used to judge whether a [pure-text
↪ recommendation, merchant list, or comprehensive answer]
↪ perfectly satisfies this Query.

Requirements

1. ****Yes/No format****: each question must be an interrogative
↪ sentence that can be clearly answered with "Yes" or "No"
↪ (starting with "Does the answer...").
2. ****Single dimension****: each question should evaluate only one
↪ independent dimension, avoiding stitching multiple
↪ requirements together (satisfying the MECE principle).
3. ****Tightly follow the context****: questions must tightly
↪ follow the explicit needs and potential pain points of the
↪ current Query; do not generate universally applicable empty
↪ questions such as "Is the answer relevant" or "Is the
↪ answer helpful".
4. ****020 scenario coverage****: prioritize key dimensions of
↪ local-life search, such as:
 - Category/service/brand matching
 - Location/distance/business-district/city constraints
 - Scenario and user-group fit
 - Merchant quality and risk avoidance (such as avoiding
↪ closed counters and imitation stores)
 - Offline wayfinding assistance (such as transportation and
↪ specific location)
5. ****Pure-text constraint****: never require evaluators to verify
↪ information outside the pure text (such as offline
↪ real-time inventory or queue status), and never require the
↪ answer to include frontend UI (such as clickable buttons,
↪ map cards, or real photos).
6. ****Quantity limit****: generate 4-8 core questions.

Output Format (strictly output JSON; do not output any
↪ Markdown markers)

```

{{
  "checklist_questions": [
    {{
      "id": "C1",
      "question": "Does the answer clearly point out...?",
      "focus": "The evaluation dimension this question focuses
↪ on (such as brand consistency or address accuracy)"
    }}
  ]
}}

```

Input

User Query: [{query}]

"""

D.7 TICK_Q Prompt

The following is the translated English version.

TICK_Q uses the checklist-generation prompt listed above as
→ Stage 1.

TICK_Q_MATERIALIZE_SYSTEM_PROMPT = """

You are a rigorous rubric materialization expert. Your task is
→ to rewrite a set of Yes/No Checklist Questions into formal
→ rubric dimensions with positive and negative criteria.
The rewritten rubric must be strictly faithful to the original
→ Checklist Questions.
"""

TICK_Q_MATERIALIZE_USER_TEMPLATE = """\

Task

I will provide a user Query and a set of Checklist Questions.
Please rewrite each Checklist Question into a formal Rubric
→ dimension, used to evaluate the quality of a [pure-text
→ answer] generated by a large language model for this Query.

Core Constraints

1. ****Do not add new dimensions****: do not introduce evaluation
→ dimensions that are not grounded in the Checklist
→ Questions.
2. ****Do not delete important constraints****: important
→ constraints contained in the Checklist Questions must be
→ preserved in the rubric.
3. ****Yes-to-good, No-to-bad mapping****: the performance
→ corresponding to answering "Yes" to a Checklist Question
→ should become `good\_response\_criteria`; the performance
→ corresponding to answering "No" should become
→ `bad\_response\_criteria`.
4. ****Faithful concretization is allowed****: you may make the
→ criteria more concrete according to the Query context, but
→ you must not introduce new business requirements that have
→ no basis in the Query or the Checklist.
5. ****Merge only highly redundant checklist items****: if multiple
→ Checklist Questions are semantically highly overlapping,
→ they may be merged into one rubric dimension.

Output Format (strictly output JSON; do not output any
→ Markdown markers or extra text)

```
{{
  "intent_analysis": "Briefly describe the user's core need and
  → potential pain points (within 50 Chinese characters).",
  "rubric_dimensions": [
    {{
      "dimension_name": "Short dimension name",
      "good_response_criteria": "High-score criterion",
      "bad_response_criteria": "Deduction criterion",
      "source_checklist_ids": ["C1", "C2"]
    }}
  ]
}}
```

Input

User Query: [{query}]

Checklist Questions:
{checklist_questions_json}
"""

TICK_Q_FAITHFULNESS_CHECK_SYSTEM_PROMPT = """

You are a rigorous rubric faithfulness auditor. Your task is to
→ check whether the generated rubric dimensions are faithful
→ to the original Checklist Questions.

You must distinguish reasonable concretization from invalid
→ fabrication.
"""

TICK_Q_FAITHFULNESS_CHECK_USER_TEMPLATE = """\

Task

I will provide the original Checklist Questions and the
→ generated rubric dimensions.
Please check whether each rubric dimension is faithful to the
→ original Checklist Questions.

Judgment Rules

Pass:

- The rubric only concretizes abstract concepts already present
→ in the Checklist Questions.
- The rubric preserves the intent and constraints of the
→ corresponding Checklist Questions.

Fail:

- The rubric introduces evaluation points that are not
→ mentioned or implied by any Checklist Question.
- The rubric deletes important constraints from the Checklist
→ Questions.
- The rubric changes the meaning of the original Checklist
→ Questions.

Output Format (strictly output JSON; do not output any
→ Markdown markers or extra text)

```
{{
  "audit_results": [
    {{
      "dimension_name": "Dimension name",
      "is_faithful": true,
      "reasoning": "Brief explanation",
      "source_checklist_ids": ["C1"]
    }}
  ],
  "overall_pass": true
}}
```

Input

Checklist Questions:
{checklist_questions_json}

Generated Rubric Dimensions:
{rubric_dimensions_json}
"""

D.8 Rubric Quality Evaluation Judge Prompt

The following is the translated English version.

Prompt for the judge model in the Rubric Quality Evaluation:

```
SYSTEM_PROMPT = (
  'You are a top-level academic "search evaluation standard
  → review expert (Meta-Evaluator)".'
  'Your core task is to conduct an absolutely objective
  → double-blind comparison and evaluation of two
  → different "scoring rubrics".'
)
```

USER_TEMPLATE = """\

Crucial Context

Please keep in mind: the final use of these two Rubrics is to
→ evaluate the quality of [pure-text recommendations,
→ merchant lists, or comprehensive answers] generated by a
→ large language model/intelligent search system for this
→ Query.

An excellent Rubric must clearly guide evaluators to examine:
→ ****whether the merchants recommended by the system are**
→ **accurate/high-quality/compliant**, whether the information
→ in the recommendation list is rich, whether it can
→ effectively assist user decision-making, etc.**

Task

I will provide you with an original [User Query], as well as
→ [Rubric A] and [Rubric B] generated for this Query.

Please carefully read the two Rubrics, strictly follow the
→ evaluation metrics and scoring anchors below to conduct a
→ deep comparison (1-9 scale), and select the final winner.

Evaluation Metrics & Scoring Anchors

1. Intent Depth & Alignment - weight 30%

- Evaluation focus: Does the Rubric accurately capture the
→ explicit and implicit needs of the Query? Does it perceive
→ the specific hidden pain points under this Query (such as
→ thresholds for specific groups, avoiding imitation stores,
→ demand for time-honored brands, etc.)?

- **[7-9 points] Deep insight**:
→ fully sees through the user's
→ real underlying concerns, accurately extracts the business
→ pain points specific to this Query, and even proactively
→ considers long-tail or abnormal cases (such as typo
→ correction and fallback for closed stores).

- **[4-6 points] Conventional understanding**:
→ accurately understands the user's literal intent and basic needs, but
→ lacks deep mining of deeper hidden pain points or specific
→ business scenarios.

- **[1-3 points] Superficial/deviated**:
→ only performs mechanical literal translation, has shallow intent
→ understanding, or even has serious semantic narrowing bias
→ (such as locking broad intent into a single scenario).

2. Actionability for Text Evaluation - weight 40%

- Evaluation focus: Are the Good/Bad Cases extremely concrete?
→ Can they clearly guide downstream evaluators to judge
→ quality only by reading "pure text"?

- **[7-9 points] Extremely implementable**:
→ the criteria are extremely clear; Good/Bad Cases tightly follow the quality
→ of pure-text recommendation lists (for example, clearly
→ pointing out what kind of low-quality merchants must be
→ penalized), and evaluators can directly follow them.

- **[4-6 points] Somewhat guiding**:
→ has some guidance value, but some descriptions are subjective (such as "the answer
→ is bad"), or contain a small amount of "correct but empty
→ statements".

- **[1-3 points] Not actionable**:
→ filled with universally applicable empty statements, or **contains
→ non-implementable criteria (such as requiring evaluators
→ to verify merchants' real-time physical inventory/queue
→ status, or requiring the system to display frontend UI
→ cards/buttons)***, with no practical value.

3. MECE & Structure - weight 30%

- Evaluation focus: Is there serious semantic overlap among the
→ selected dimensions (stitched monsters)? Is the
→ decomposition of dimensions professional and reasonable?

- **[7-9 points] Logically rigorous**:
→ dimensions are mutually independent and collectively exhaustive (satisfying the
→ MECE principle), logical decoupling is extremely clear,
→ and one error corresponds to deduction under only one
→ dimension.

- **[4-6 points] Minor flaws**:
→ the overall structure is reasonable, but there is slight semantic overlap between a
→ few dimensions, or the decomposition is somewhat
→ fragmented/bloated.

- **[1-3 points] Structural collapse**:
→ serious semantic duplication exists (causing the same error to be penalized
→ repeatedly), or too many dimensions cause loss of focus, or
→ too few dimensions miss key information.

Strict Guidelines

1. **Severely penalize "correct but empty statements"**: if the
→ criteria of a Rubric are universally applicable (can also
→ be used under other Queries, without incorporating the
→ specific context of the current Query), severe deductions
→ must be given on "actionability"!

2. **Severely penalize "non-implementable hallucinated
→ criteria"**: if a Rubric proposes requirements outside the
→ scope of pure-text evaluation for large models (for
→ example, requiring one-click direct buttons, requiring
→ high-definition real photos, requiring checking real-time
→ sold-out status, etc.), you must point out that they are
→ unrealistic and penalize them!

3. **Mandatory evidence citation**:
→ when writing comparison reasons, you must cite specific original text fragments
→ from Rubric A or B as evidence.

Output Format (strictly output JSON; do not output any
→ Markdown markers or extra text)

```
{{
  "evaluation_reasoning": {{
    "critical_flaws_analysis": "[Critique] Respectively point
    → out the biggest flaws in Rubric A and Rubric B (such as
    → whether they contain correct but empty statements,
    → non-implementable UI/physical-state requirements, or
    → overlapping dimensions; original text must be cited).",
    "intent_depth_comparison": "[Comparison] Which Rubric is
    → better in intent depth? Why? (original text must be
    → cited as evidence).",
    "actionability_comparison": "[Comparison] Which Rubric
    → better guides the evaluation of pure-text
    → recommendation lists? Whose Good/Bad Cases are more
    → concrete and implementable?"
  }},
  "scores": {{
    "rubric_A": {{
      "intent_depth": 0,
      "actionability": 0,
      "mece_structure": 0,
      "total_score": 0
    }},
    "rubric_B": {{
      "intent_depth": 0,
      "actionability": 0,
      "mece_structure": 0,
      "total_score": 0
    }}
  }},
  "final_decision": "Rubric A | Rubric B | Tie"
}}
```

total_score calculation rule: intent_depth x 0.3 +
→ actionability x 0.4 + mece_structure x 0.3, rounded to one
→ decimal place. final_decision must be strictly based on the
→ total_score comparison and must not be judged subjectively
→ by feeling.

Input

Original user Query: [{query}]

=====

[Rubric A]

{rubric_A_json_str}

=====

[Rubric B]

```
{rubric_B_json_str}"""
```

```
Model Answer from Model B:  
{answer_B}  
"""
```

D.9 Online AI Search Evaluation Score Prompts

The following is the translated English version.

All rubric/checklist generation prompts in the Online AI Search
→ Evaluation are the same as the corresponding prompts listed
→ above. Therefore, we only list the scoring prompt here.

```
score_prompt:  
SYSTEM_PROMPT = """You are a rigorous local-life search quality  
→ evaluator. Your task is to use the given evaluation rubric  
→ to score two model answers to the same user Query.  
[Model answer description: insert the description of the model  
→ answer here.]"""
```

```
USER_TEMPLATE = """\  
# Task
```

```
I will provide a user Query, an evaluation rubric, and two  
→ model answers produced by two online model versions.  
Please score both model answers strictly according to the given  
→ rubric.
```

```
# Evaluation Guidelines
```

1. Score each answer only by reading the provided text. Do not
→ use external knowledge, real-time business status, maps,
→ images, or any information outside the provided answer.
2. For each rubric dimension, assign an integer score from 0 to
→ 5.
 - 5 means the answer fully satisfies the positive criterion.
 - 0 means the answer clearly matches the negative criterion.
 - 1-4 indicate different degrees of partial satisfaction.
3. Penalize hallucinated, off-topic, generic, or non-actionable
→ content according to the corresponding rubric dimension.
4. The final decision must be based on the total score. If Model
→ A has a higher total score, output "A"; if Model B has a
→ higher total score, output "B"; otherwise output "Tie".

```
# Output Format (strictly output JSON; do not output any  
→ Markdown markers or extra text)
```

```
{{  
  "scores": {{  
    "A": {{  
      "dimension_scores": [  
        {{  
          "dimension_name": "Dimension name",  
          "score": 0,  
          "reason": "Brief reason for this score"  
        }}  
      ],  
      "total_score": 0,  
      "normalized_score": 0  
    }},  
    "B": {{  
      "dimension_scores": [  
        {{  
          "dimension_name": "Dimension name",  
          "score": 0,  
          "reason": "Brief reason for this score"  
        }}  
      ],  
      "total_score": 0,  
      "normalized_score": 0  
    }}  
  }},  
  "final_decision": "A | B | Tie"  
}}
```

```
# Input
```

```
User Query: [{query}]
```

```
Evaluation Rubric:  
{rubric_json}
```

```
Model Answer from Model A:  
{answer_A}
```