FACET: Decomposing Prompts into Reusable Primitives for Transfer

Jixian Su[1] **Yusen Zhang**[2] **Eugene Wu**[2] **Kexin Rong**[1]
[1]Georgia Institute of Technology, Atlanta, USA
[2]Columbia University, New York City, USA
{jixiansu, krong}@gatech.edu {yz5296, ew2493}@columbia.edu

Keywords: prompt optimization, transferability, nl2sql, table QA, LLM agents

1. INTRODUCTION

As LLM agents grow more complex, prompts remain the primary mechanism for guiding their behavior. Methods such as OPRO [10] and GEPA [1] optimize prompts within a single benchmark, while frameworks such as DSPy [6] and SPEAR [2] encourage modular reuse. However, prompt development is still largely benchmark-specific: methods optimize prompts within a single dataset, leaving it unclear whether they transfer to new settings.

We study this question in structured table tasks such as NL2SQL and table QA, where benchmarks share a common objective but differ in their query distributions and table schemas. They repeatedly invoke similar operations and reasoning patterns over tables, suggesting existing well-performing prompts may remain useful across benchmarks.

Our key insight is that prompts are structured objects composed of reusable primitives rather than flat text, and that their effects are context-dependent. We present FACET, a framework for analyzing and transferring prompts based on this view. FACET models prompts as compositions of *prompt primitives*: typed, reusable prompt operators that act on different parts of a prompt and characterizes each query instance with lightweight *context predicates*. Together, these enable FACET to characterize the context-dependent effects of individual primitives and support selective reuse across benchmarks.

2. METHODOLOGY

FACET takes as input (i) a collection of prompts for a given task and (ii) a set of queries from one or more benchmarks. The prompt primitives are drawn from a source benchmark and evaluated on target benchmarks for the same task.

Rather than treating each prompt as a monolithic string, FACET represents prompts as compositions of reusable *prompt primitives*: composable prompt operators that act on different parts of prompt. These primitives include operators that add or organize semantic content (e.g., rules, examples, sections), control formatting choices, and switch reasoning modes (e.g., enabling chain-of-thought). This representation turns prompts from flat text into structured configurations and makes primitives the basic units for analysis and reuse.

Each query instance from the benchmark is associated with a set of *context predicates*–observable properties of the query, its underlying table, and the execution setting. These predicates capture factors such as the required operation (e.g., aggregation, filtering), table structure, column types, and the model used.

FACET evaluates different prompt primitive compositions across queries and measures the effect of each primitive both globally and within predicate-defined subgroups. This reveals *conditional effects*: a primitive may help one subgroup while hurting another. These effects provide a more stable view of transferability than benchmark-level averages, which can obscure important variation due to differing query distributions.

Finally, FACET exploits these conditional effects through *predicate-based routing*. Instead of applying prompt primitives uniformly, it learns simple predicate-based policies that activate each primitive only when its predicted effect is positive, enabling context-aware reuse.

3. PRELIMINARY RESULTS

In preliminary experiments, we evaluate primitive-level transfer across benchmarks within the same task. We evaluate FACET on two table tasks using Qwen2.5-7B [9]: NL2SQL on BIRD [7] and Spider [11] with 24 shared prompt primitives, and table QA on WTQ [8], SQA [5], HiTab [4], and TabFact [3] with 15. Not every primitive in the shared pool is applicable to every benchmark, so the number of primitives compared varies by benchmark pair.

Conditional effects recur across benchmarks when global effects disagree. WTQ and SQA agree on 10 of 12 primitives at the benchmark level, while other pairs agree on as few as 4 of 11. Yet conditioning on predicates reveals effects that remain consistent even when global effects disagree. In NL2SQL, a primitive that encourages query decomposition with common table expressions has near-zero effect on BIRD and lowers Spider by 1.9% overall, but improves queries over deeply nested schemas by 1.6% on BIRD and 2.3% on Spider. In table QA, "read all column headers" yields a 0.9% gain on SQA and negligible change on TabFact overall, but improves filtering queries by 3.0% on SQA and 4.6% on TabFact.

Routing at the predicate level outperforms static selection. Predicate-based routing exploits conditional structure learned within the same task. In NL2SQL, we train on BIRD a predicate-aware policy that decides when each primitive should be activated under different predicate configurations. Transferred to Spider, this policy improves performance by 1.7%, whereas statically applying the best primitive from BIRD reduces performance by 0.7%.

References

[1] P. Agrawal, O. Khattab, et al. Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

[2] U. Cetintemel, S. Chen, A. W. Lee, and D. Raghavan. Making prompts first-class citizens for adaptive llm pipelines. *arXiv preprint arXiv:2508.05012*, 2025.

[3] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. Tabfact: A large-scale dataset for table-based fact verification. In *ICLR*, 2020.

[4] Z. Cheng, H. Dong, Z. Wang, R. Jia, J. Guo, Y. Gao, S. Han, J.-G. Lou, and D. Zhang. Hitab: A hierarchical table dataset for question answering and natural language generation. In *ACL*, 2022.

[5] M. Iyyer, W.-t. Yih, and M.-W. Chang. Search-based neural structured learning for sequential question answering. In *ACL*, 2017.

[6] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Mober, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

[7] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. C. Chang, F. Huang, R. Cheng, and Y. Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111*, 2023.

[8] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In *ACL*, 2015.

[9] Q. Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[10] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[11] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, 2018.