FilterPilot: An Interactive Assistant for Adapting Filtering Predicate to Table Content

Shiwen Wu
The Hong Kong University of Science and Technology
Hong Kong SAR, China
swubs@connect.ust.hk

Yuqi Wang
The Hong Kong University of Science and Technology
Hong Kong SAR, China
ywangyr@connect.ust.hk

Yue Pang
The Hong Kong University of Science and Technology
Hong Kong SAR, China
michellepy@ust.hk

Xiaofang Zhou
The Hong Kong University of Science and Technology
Hong Kong SAR, China
zxf@ust.hk

ABSTRACT

Formulating effective filtering conditions for database queries often requires a deep understanding of the specific values stored within table columns, which can impose a significant cognitive burden on users, especially when the textual columns contain hundreds or even thousands of unique values. To address this challenge, we introduce FilterPilot, an LLM-powered interactive assistant designed to adapt filtering predicates to table content. Given user-provided filtering terms and optional time constraints, FilterPilot assists by suggesting appropriate filtering conditions. At its core, FilterPilot employs a novel iterative recall-then-verify paradigm, combining LLM-based query reformulation with table-value feedback to dynamically expand search terms. This process is further optimized through an Upper Confidence Bound (UCB) selection mechanism and an early stopping strategy. We release the source code, datasets, and a live demo system on https://github.com/wusw14/FilterPilot to encourage community adoption.

PVLDB Reference Format:
Shiwen Wu, Yuqi Wang, Yue Pang, and Xiaofang Zhou. FilterPilot: An Interactive Assistant for Adapting Filtering Predicate to Table Content. PVLDB, 19(12): 4626 - 4629, 2026.
doi:10.14778/3827998.3828082

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/wusw14/FilterPilot.

1 INTRODUCTION

Filtering textual columns is a fundamental task in various data analysis scenarios. However, crafting effective filtering predicates for table columns often demands a deep understanding of the values contained within the column. This challenge is especially significant when dealing with textual columns containing hundreds or thousands of unique values, making it difficult for users to become sufficiently familiar with the data to define accurate filtering predicates. As illustrated in Figure 1, a user searching for "Chinese food" might miss relevant records labeled as "Sichuan food" or "Cantonese food" if relying on exact match or simple keyword search. Therefore, an automated approach is essential to bridge the gap between users' filtering intentions and the actual column values.

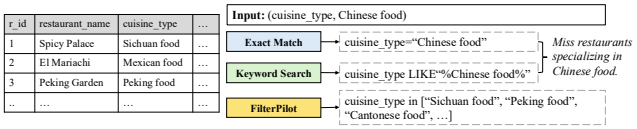

Figure 1: Examples showing the importance of aligning filtering intentions with table content for accurate query results.

Recent advances in Large Language Models (LLMs) offer potential to align filtering intentions with database content through semantic understanding. However, directly scanning all table values with LLMs is computationally infeasible for large-scale databases. Current methods adopt a recall-then-verify (RTV) paradigm [1–3], where a retriever selects candidate values for LLM verification. While efficient, these approaches suffer from low recall when substantial semantic gaps exist between query terms and table content. For instance, as illustrated in Figure 1, entries such as "Sichuan food" and "Peking food" are semantically subsumed under "Chinese food". However, these hierarchical relationships often receive low similarity scores in retrieval models, which might lead to the exclusion of semantically relevant values.

Our Approach. We introduce FilterPilot, an interactive system that leverages an iterative query expansion framework to improve the retrieval of aligned table values. The key insight is that alternative expressions—such as synonyms, narrower concepts, or format variants—might be better matches to database values than the original query term, enabling more effective retrieval. Specifically, FilterPilot enriches query terms by incorporating: (1) the reformulated terms generated by the LLM from multiple perspectives, and (2) table values identified by the LLM as satisfying the specified filtering predicate. To optimize the query terms for retrieval, FilterPilot uses an Upper Confidence Bound(UCB)-based mechanism to balance exploration—by leveraging underexplored terms—and exploitation—by focusing on terms that have demonstrated effectiveness. Additionally, FilterPilot implements an early

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828082
* Xiaofang Zhou is the corresponding author.

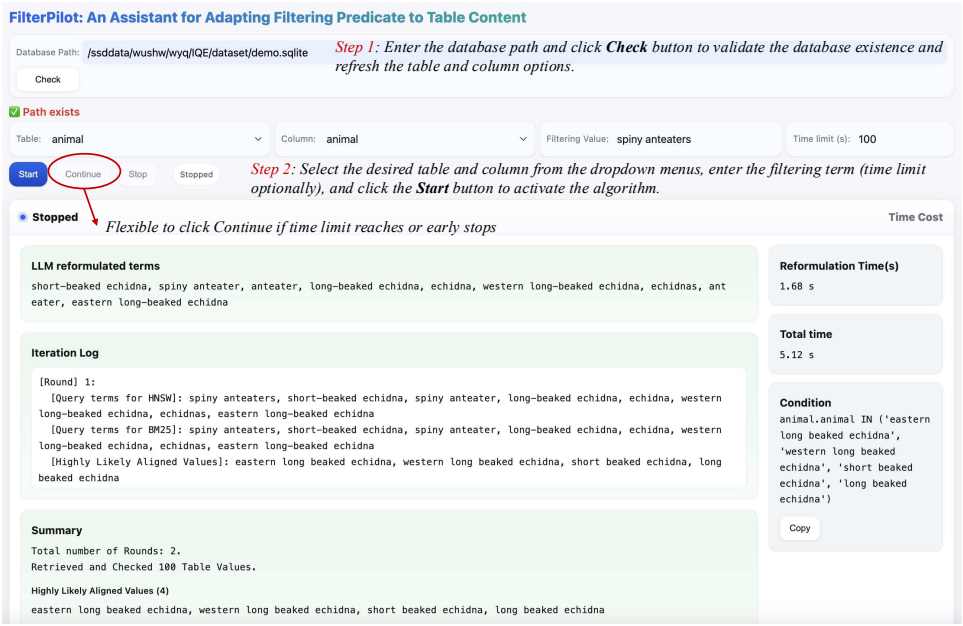

Figure 2: User Interface of FilterPilot.

stopping mechanism to minimize unnecessary LLM evaluations. Importantly, users retain the ability to continue the process even after early stopping or after the exhaustion of the predefined time limit, ensuring flexibility and control in interactive settings.

Demonstration Highlights. FilterPilot is an interactive system that assists the user in writing filtering conditions on textual columns, as shown in Figure 2. The system offers the following key features:

- **Convenient and Easy to Use:** Users can quickly get started by selecting a table and its columns, specifying a filtering value, and optionally setting a time limit. The system then automatically generates suggested filtering predicates.
- **Real-Time Inspection of Intermediate Results:** Throughout the process, users can monitor the reformulations produced by the LLM as well as the verification results in real-time.
- **Efficient and Adaptive Workflow:** The system ensures timely responses by adhering to the specified time limit and employs early stopping mechanisms to skip unnecessary verifications.
- **Flexible Control:** Users retain full control over the process, with the flexibility to assess intermediate results and decide whether to proceed accordingly.

2 KEY TECHNIQUES OF FILTERPILOT

FilterPilot is an interactive system designed to assist users in constructing filtering predicates for textual columns within a database.

2.1 User Interaction Workflow

The user interaction workflow is designed to be intuitive and user-friendly, as illustrated in Figure 2.

(1) **Database Selection:** Users input the database's file path and click *Check* to validate it. If valid, the system dynamically displays the database tables and their respective columns.

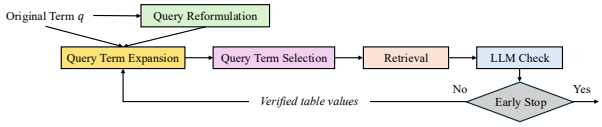

Figure 3: Overall Architecture of FilterPilot.

(2) **Table and Column Selection:** Users choose a table and column from dropdowns, then enter an initial filtering term. A time budget can be optionally set, defaulting to 100 seconds.
(3) **Algorithm Initiation:** Clicking *Start* launches the system's algorithm.
(4) **User Control:** Users maintain control throughout. If the algorithm stops early or the time budget is met, intermediate results (in the Iteration Log) are presented for review, allowing users to decide whether to continue. If aligned table values persist in the final iteration, extending the time is recommended for potentially discovering more.
(5) **Process Summary:** FilterPilot provides a comprehensive summary in the Summary panel. It also generates a recommended filtering predicate, as shown in the lower right of Figure 2.

2.2 System Architecture

To facilitate these capabilities, the backend architecture comprises five core components, as shown in Figure 3. We will briefly introduce each part in the following.

2.2.1 Query Term Expansion. When the original filtering term does not lexically or semantically match the stored data (e.g., querying "Chinese food" when the database contains "Sichuan food"), direct use of the original term may fail to retrieve relevant entries. To

address this, our framework enriches the original term through two complementary approaches: LLM-based Multi-aspect Query Reformulation and Iterative Incorporation of Verified Table Values.

LLM-Based Multi-Aspect Query Reformulation. Leveraging the extensive internal knowledge of LLMs, we generate diverse query expansions to cover potential variations in the database content. The process employs a Multi-Aspect Query Reformulation strategy, guiding the LLM to produce terms from three distinct perspectives:

- **Concept Hierarchy:** The LLM generates specific subclasses or instances of the original term. For example, if the input is "Chinese food," the LLM is expected to expand this to include "Sichuan food," "Cantonese food," or "Peking food."
- **Synonyms and Aliases:** The LLM identifies alternative expressions with equivalent meanings. For instance, "carbon dioxide" might be expanded to include its chemical formula "CO_2."
- **Format-specific Variants:** To address non-semantic discrepancies, the LLM generates format-specific variations, such as transforming "fns 37402 i" into "fns37402i" to align with database conventions.

By combining outputs from these three perspectives, the system generates a comprehensive set of seed terms Q_l that significantly broadens the initial retrieval scope beyond the user's original input.

Iterative Incorporation of Verified Table Values. In addition to reformulation, our framework utilizes a dynamic feedback loop to further expand query terms during the retrieval process. In each iteration, retrieved candidate values are evaluated by the LLM, and those confirmed to match the filtering intent are added to a set of verified values (R^+). These verified values are then used as supplementary query terms in subsequent iterations.

This iterative approach exploits the semantic proximity of verified terms to uncover additional relevant records. For example, once "gaseous propoxur" is verified, it can serve as a query term in the next iteration to retrieve related values, such as "liquid propoxur," which might have been omitted when using the initial query term or reformulated terms for retrieval.

2.2.2 Query Term Selection. In our framework, retrieval query terms are drawn from three complementary sources: (1) the original user query q; (2) LLM-reformulated variants Q_l that capture alternative phrasings or semantic expansions; and (3) table entries R^+ that the LLM has previously verified as satisfying q. This multi-source expansion broadens the semantic coverage of retrieval, increasing the likelihood of identifying relevant table values.

However, not all the terms are equally effective for retrieval. Therefore, our system dynamically selects a subset of query terms using an *Upper Confidence Bound* (UCB) strategy. UCB is a principled approach for balancing *exploitation* (favoring terms that have historically yielded many relevant results) and *exploration* (allocating budget to less-explored terms that may uncover new relevant values).

For each candidate term q_i, the system maintains a UCB score:

$$UCB(q_i) = w_{q_i} + c\sqrt{\frac{\ln N_{all}}{N_{q_i}}}, \tag{1}$$

where w_{q_i} estimates the term's contribution to retrieving satisfying values, N_{q_i} counts how many similar table values have been checked for q_i, N_{all} aggregates evaluations across all terms, and c controls the exploration weight (set to 0.01 by default).

The contribution estimate w_{q_i} is updated iteratively using LLM feedback. After each round of verification, we compute the proportion of checked values that satisfy q and blend this observation with the prior estimate via a moving average:

$$w_{q_i} \leftarrow (w_{q_i} + w'_{q_i})/2, \quad w'_{q_i} = \frac{\#\text{ satisfying values for } q_i}{N'_{q_i}}. \tag{2}$$

This adaptive updating allows the system to progressively refine its understanding of which query terms are most fruitful, leading to more efficient budget allocation over time.

2.2.3 Retrieval and Reranking. This part outlines our hybrid retrieval module, which combines lexical and semantic retrievers for recall relevant values, followed by a reranking mechanism.

Hybrid Retrieval and Aggregation. We employ both lexical (BM25) and semantic retrievers to capture exact matches and contextual relevance, respectively. For each query term, the system retrieves the top-B (defaulting to 500 in our implementation) candidate entries from the database. To handle multiple query terms, we aggregate the results by selecting the maximum relevance score for each entry across all terms.

Adaptive Fusion Reranking. Retrieved entries from both retrievers are merged and reranked using a hybrid fusion score. This score dynamically balances lexical and semantic signals through weighted parameters. Unlike static weighting schemes, our system optimizes these weights based on feedback from previous iterations, employing a grid search strategy to maximize ranking metrics (*e.g.*, NDCG). The hybrid fusion score $s(d)$ is computed as:

$$s(d) = w_1 s'_{lex}(d) + w_2 s'_{sem}(d) + w_3 \max(s'_{lex}(d), s'_{sem}(d)), \tag{3}$$

where $w_1, w_2, w_3 \in [0, 1]$ are weights satisfying $w_1 + w_2 + w_3 = 1$, and $s'_{\text{lex}}(d)$, $s'_{\text{sem}}(d)$ are the normalized scores from the lexical and semantic retrievers, respectively. This adaptive approach prioritizes entries that demonstrate strong relevance in either or both dimensions.

2.2.4 LLM Check. The retrieved values are subsequently sent to the LLM to verify their alignment with the query q. To mitigate the risk of hallucination by the LLM, we implement a two-stage verification process. This process employs prompts with varied phrasings and restricts the LLM's output to one of three deterministic responses: `Yes`, `Unsure`, or `No`. The inclusion of the `Unsure` option enables the system to handle ambiguous cases without forcing the LLM to make potentially inaccurate confident predictions. The prompts employed in the two verification stages are as follows:

- **First Check Prompt:** Is 'd' the same as or a type of 'q'? Respond only with 'Yes', 'Unsure', or 'No'.
- **Recheck Prompt:** Please verify if the value 'd' satisfies the condition: The value is the same as or a type of 'q'. Respond only with 'Yes', 'No', or 'Unsure' without adding any additional text.

For scoring purposes, the responses `Yes`, `Unsure`, and `No` are assigned numerical values of 2, 1, and 0, respectively. Only values that achieve a combined score of 3 or higher across both verification stages are considered aligned with q.

Table 1: Comparison of overall results across methods, where B represents the maximum number of retrieved table values.

Dataset	Animal		Chemical Compound		Product	
Metric	Time(s)	F1(%)	Time(s)	F1(%)	Time(s)	F1(%)
non-LLM-based methods						
Keyword Search	0.00	62.76	0.00	50.02	0.01	11.94
Lexical Sim	0.00	64.30	0.00	52.11	0.02	35.57
Semantic Sim	0.01	57.59	0.01	53.01	0.02	10.40
LLM-based Methods—LLaMA-3.3-70B						
LOTUS	105.23	41.43	55.62	66.01	577.03	46.27
RTV(B=100)	2.16	77.81	2.14	77.15	2.49	53.99
FilterPilot (B=100)	3.59	86.93	3.93	82.86	7.51	76.83
RTV(B=500)	9.00	79.66	8.69	76.73	10.43	76.67
FilterPilot (B=500)	7.18	86.06	9.68	80.85	18.06	78.27

2.2.5 Early Stopping Mechanism. To improve system efficiency, FilterPilot incorporates an adaptive early stopping mechanism, which combines two widely used strategies—score thresholding and consecutive failure counting. The verification loop halts once at least one value aligned with the query has been identified and either of the following conditions is satisfied: (1) the similarity scores of candidate entries fall below a dynamic threshold τ; or (2) no values meet the filtering condition for t consecutive iterations.

In our implementation, τ is dynamically determined based on the similarity scores of the currently satisfying values. Specifically, τ is set slightly below the minimum score of the identified matches. According to our experimental results, the early stopping mechanism generally proves effective in balancing computational efficiency and effectiveness. However, premature termination may occur in certain scenarios, such as when the accuracy of the LLM Check is relatively low, or when users, based on their domain knowledge or intuition, believe that there are still likely candidates meeting the desired conditions. In such cases, users are advised to consider continuing the process.

2.3 Effectiveness Evaluation

We evaluate FilterPilot against several representative baselines: Keyword Search, Lexical Similarity (using BM25), semantic similarity search (using HNSW), LOTUS [4], and RTV. RTV represents a group of recall-then-verify methods, including TableRAG [2], SUQL [3], and Pneuma [1]. For RTV and our method, we compare performances at $B = 100$ and $B = 500$, where B represents the maximum number of retrieved table entries to be checked by the LLM.

Experiments on three datasets (Animal, Chemical Compound, Product) demonstrate that FilterPilot consistently outperforms baseline methods. Compared to LOTUS, which relies on exhaustive verification, FilterPilot achieves higher accuracy while significantly reducing latency by prioritizing high-potential candidates. When compared to RTV, FilterPilot improves F1 scores by 3.60-22.84% at $B = 100$, and 1.60-6.40% at $B = 500$ in F1 scores. Notably, increasing B (and consequently time) does not always yield higher F1 scores, as gains in retrieval recall may be offset by reduced precision. Users can apply filters to refine the results as needed.

3 DEMONSTRATION

Case Study 1: Effectiveness of the Early Stopping Mechanism with Large Time Limits. Figure 2 illustrates the scenario where the user sets a large time limit of 100 seconds. After two iterations, the early stopping mechanism is activated, with the total elapsed time around 5 seconds. In this instance, the system successfully retrieves all aligned table values during the first iteration, rendering additional iterations unnecessary. This demonstrates how the early stopping mechanism significantly reduces computational time without compromising the results.

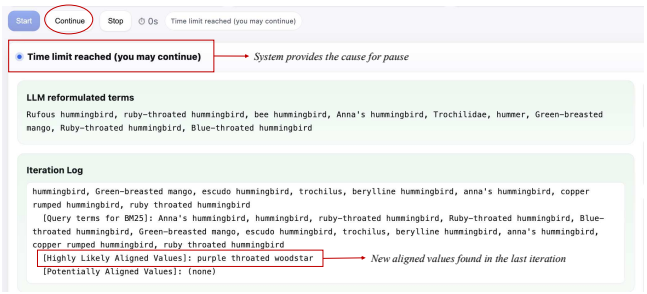

Figure 4: Case 2: Flexible Control with Short Time Limits. The user is provided with the option to extend the time limit by clicking "Continue".

Case Study 2: Flexible Control with Short Time Limits. Figure 4 depicts a scenario where the user specifies the filtering values as "hummingbird" with a 6-second time limit. The process stops after three iterations upon reaching the time limit, retrieving 9 aligned values from a total of 34 values. Notably, the last iteration produces new aligned values, indicating potential for further retrieval given more time. The system clearly shows the termination reason and the suggestion for the action, allowing users to decide whether to continue. Upon clicking "Continue", the system executes an additional iteration, retrieving 7 more aligned values. Each subsequent click on "Continue" triggers one more iteration. This case demonstrates how the flexible control mechanism enables users to balance computational efficiency and result completeness through real-time feedback.

ACKNOWLEDGMENTS

The research work was supported by INSPUR GROUP CO.,LTD (grant#INSPUR24EG01-C). It was conducted in JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

REFERENCES

[1] Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. 2025. Pneuma: Leveraging llms for tabular data representation and retrieval in an end-to-end system. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–28.

[2] Si-An Chen, Lesly Miculicich, Julian Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024. Tablerag: Million-token table understanding with language models. *Advances in Neural Information Processing Systems* 37 (2024), 74899–74921.

[3] Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina Semnani, Chen Yu, and Monica Lam. 2024. SUQL: Conversational Search over Structured and Unstructured Data with Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 4535–4555.

[4] Liana Patel, Siddharth Jha, Parth Asawa, Melissa Pan, Carlos Guestrin, and Matei Zaharia. 2024. Semantic Operators: A Declarative Model for Rich, AI-based Analytics Over Text Data. arXiv:2407.11418 [cs.DB] https://arxiv.org/abs/2407.11418