Demonstrating TACT: Tunable Accuracy-Cost Trade-offs for Semantic Image Filtering

Tharushi Jayasekara
Cornell University
Ithaca, NY, USA
tkj27@cornell.edu

Immanuel Trummer
Cornell University
Ithaca, NY, USA
itrummer@cornell.edu

ABSTRACT

We present an interactive demonstration system for semantic filtering of large-scale image datasets using natural language predicates, with a focus on flexible control over accuracy–cost trade-offs. Recent semantic query processing systems rely on large language models (LLMs) to evaluate such predicates, but uniformly applying these models across all data is prohibitively expensive and limits adaptability. Our system introduces a flexible multi-stage filtering framework built from a diverse set of operators, including embedding-based filters and LLM-based operators each offering different accuracy and cost characteristics. By composing these operators into cascades, the system can realize a wide spectrum of execution strategies, ranging from low-cost to high-accuracy semantic evaluation. Given user-defined preferences over precision, recall, and monetary cost, the system profiles candidate operators on a sample of data, estimates their performance, and automatically selects operator cascades that best satisfy these preferences. This enables users to explicitly navigate trade-offs and tailor execution behavior to their needs. Through an interactive interface, users can explore how different queries, datasets, and preference settings influence operator selection, Pareto-efficient trade-offs, and execution outcomes, demonstrating the system's ability to adaptively balance precision, recall, and cost in semantic filtering workflows.

PVLDB Reference Format:
Tharushi Jayasekara and Immanuel Trummer. Demonstrating TACT: Tunable Accuracy-Cost Trade-offs for Semantic Image Filtering. PVLDB, 19(12): 4722 - 4725, 2026.
doi:10.14778/3827998.3828106

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/TharushiJay/demo-efficient-semantic-filtering.git.

1 INTRODUCTION

The recent emergence of semantic query processing engines in both industry [1, 2] and academia [3, 5, 6] has enabled the querying of unstructured data using natural language predicates. These systems extend traditional database engines with semantic operators that allow users to express queries over images, text, and video. A key feature in these systems is *semantic filtering*, where a natural language predicate (e.g., "contains a red sports car" or "shows workers without safety helmets") is evaluated over large collections of unstructured data. Modern systems increasingly rely on large language models (LLMs) as predicate evaluators due to their strong semantic understanding. While these systems provide powerful abstractions for semantic querying, they typically treat operator selection and execution strategies as fixed, and do not expose mechanisms for users to explicitly control trade-offs between accuracy and cost. In particular, user preferences over metrics such as precision, recall, and monetary cost are not directly incorporated into query planning.

To address this challenge, we present an operator-driven multi-stage semantic filtering framework that explicitly incorporates user preferences into query execution. Given a dataset, a natural language predicate, and user-specified preferences over accuracy and cost, our system profiles candidate operators on a small sample, estimates their error and cost characteristics, and automatically selects an operator cascade that balances accuracy and cost according to user-defined preferences. Using statistically robust estimates of operator performance, it evaluates alternative single and multi-stage schedules and selects the one that maximizes a weighted utility function. While the proposed approach is general and applicable to a wide range of unstructured data modalities, our current implementation focuses on semantic filtering over image datasets. The selected schedule is then executed on the remaining data, enabling scalable semantic filtering with explicit control over accuracy-cost trade-offs.

Example 1.1. Consider a user searching for "toys for children under 5" on an online marketplace. In this setting, high precision is critical to ensure that returned items are appropriate and safe, as false positives such as toys with small detachable parts or unsuitable materials may pose safety risks. However, the desired trade-off between accuracy and cost can vary across users. For instance, a budget-conscious user searching on platforms such as Craigslist may prioritize low monetary cost, whereas a large retailer with greater resources may be willing to incur higher costs to achieve high precision and ensure product safety. In contrast, some applications may also need to prioritize for recall. Consider a factory management system filtering images for safety violations such as "workers without safety helmets." Here, high recall is paramount, as missed violations may lead to accidents, regulatory penalties, or legal liability. At this scale, naively invoking a large language model such as GPT-4o priced at $2.50 per million input tokens, can cost over $5,000 in inference costs for a single query, without guaranteeing that the desired precision–recall tradeoff is achieved. Alternatively, a cheaper model like GPT-4.1-nano reduces monetary

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828106

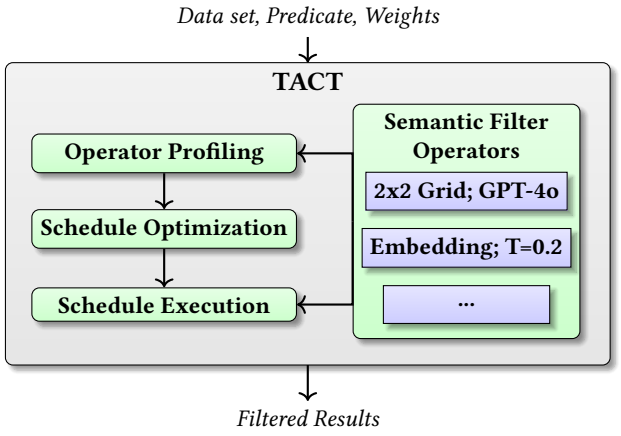


Figure 1: Architecture of the system

cost but may degrade accuracy. These examples illustrate the fundamental challenge: semantic filtering over large image collections requires carefully balancing cost and quality, rather than relying on a single model applied uniformly across all images.

Our system employs multiple semantic filter operators based on large language models (LLMs) and embedding similarity. Rather than invoking an LLM independently for each image, we improve efficiency by batching multiple images into a fixed-size grid and submitting the grid as a single model input. The LLM is prompted to identify the coordinates of sub-images that satisfy the predicate, thereby amortizing inference cost across multiple images. An LLM-based operator is therefore characterized by the underlying model and the grid size. In addition to LLM operators, we support embedding-based filters that compute semantic similarity between image embeddings and the query embedding. Images are ranked according to their similarity scores, and a threshold is applied to determine candidate positives. Each embedding-based operator is parameterized by its similarity threshold, which controls the set of output images.

Each LLM-based operator can be instantiated with different underlying models and grid sizes, yielding a spectrum of operators that trade off inference accuracy against monetary cost. Our system automatically selects cost-efficient multistage filtering pipelines composed of these operators. To enable principled optimization, we profile each operator on a small sample of images and estimate its false positive rate (FPR) and false negative rate (FNR). Using Hoeffding confidence bounds, we obtain conservative quality estimates that support statistically rigorous decision making. The optimizer evaluates operator combinations to identify Pareto-optimal cascades that satisfy user-specified precision and recall preferences while minimizing expected cost.

2 SYSTEM OVERVIEW

As depicted in Figure 1, our system takes as input an image collection, a natural-language predicate, and a set of user-defined weights that capture preferences over accuracy and cost. The system consists of three main components: operator profiling, schedule optimization, and schedule execution, which together produce the final filtered results. We describe each component in detail below.

Operator Profiling. The profiling engine estimates the accuracy of candidate semantic operators on a representative sample of the dataset (typically 10% of the data or 100 images selected via uniform random sampling). It takes as input the search query, a list of operators, and a subset of images. The cost of each operator is largely inherent and can be precomputed directly from model API pricing and grid configuration. However, operator accuracy (FPR/FNR) depends on both the predicate and the image distribution, motivating per-query profiling. Since true labels are unavailable, we use a reference model with sufficient accuracy (e.g., GPT-5) to generate pseudo ground truth values on the profiling sample.

We partition the profiling image set into independent batches of a fixed size. Then, each individual operator is profiled on each batch of the image data set. For each operator configuration, we measure the false positive rate (fraction of negative images incorrectly classified as positive), false negative rate (fraction of positive images incorrectly classified as negative), monetary cost (cost per image based on API pricing and token usage) and latency (processing time per image).

The operators used in our system are of two types: embedding-based and LLM-based. For embedding-based operators, we first generate a similarity ranking based on the search query and the profiling image set. Then, we compute thresholds from the actual score distribution of the profiling sample using percentiles. Specifically, a percentile-p operator passes the top p% of images by similarity score, with the cutoff set to the (100-p)-th percentile of observed scores. This makes each operator dataset-agnostic: a top-10% operator consistently passes roughly 10% of images regardless of how scores are distributed for a given predicate. For LLM-based operators, we utilize a grid-based operator that arranges multiple images (e.g., 3×3 = 9 images) into a single composite grid image. The LLM then identifies which grid cells satisfy the predicate, amortizing the per-call overhead across multiple images. For example, the prompt asks “Identify coordinates of images containing [predicate]" and returns the coordinates as results. E.g., “(0,1), (2,2), (1,0)." For each operator configuration, we record per-batch FPR and FNR against pseudo ground truth labels.

Schedule Optimization. The optimizer uses the operator statistics that were generated in the profiling stage, along with the user weights that specify constraints on the required performance and identifies the optimal combination of operators, known as a schedule. We consider three classes of schedules. A *single-operator schedule* applies one operator to all images and returns the set of predicted positives. A *union schedule* combines two operators using a logical OR: both operators are applied to all images, and an image is accepted if *either* operator predicts it positive. This lowers the false negative rate at the cost of more false positives, making union schedules well suited for high recall requirements. A *intersect schedule* combines two operators using a logical AND: an image is accepted only if *both* operators predict it positive, suppressing false positives at the cost of a higher false negative rate, making it well suited for high precision requirements. To obtain statistically robust estimates, we compute Hoeffding bounds for each operator. Given batch-level measurements $X_1, \ldots, X_n \in [0, 1]$ for FPR and

FNR, the $(1-\delta)$ confidence interval is:

$$\bar{X} \pm \sqrt{\frac{\ln(2/\delta)}{2n}}$$

We use Hoeffding bounds to compute upper confidence bounds for FPR and FNR from profiling samples, ensuring conservative estimates during optimization. Building on these bounds, we devise a custom analytical model to estimate the precision, recall, and cost of candidate operator schedules. We normalize the estimated metrics to be between 0 and 1 using minmax normalization. The cost is inverted when normalizing, as a lower cost is more desirable. The user-defined weights are also normalized such that they are proportional to their sum. Finally, we compute the weighted score and rank by descending order of weighted score to identify the schedule with the highest score.

Schedule Execution. Finally, the execution engine applies the selected schedule to the remaining samples in the dataset. Each operator is executed on the set of candidate images and forwards only those predicted as positive to the next operator in the schedule. This process continues until all operators have been applied. The final set of positives is combined with the positives from the profiling stage. To improve throughput, the execution engine supports parallel execution using configurable thread pools.

3 EXTRACT OF EXPERIMENTAL EVALUATION

Due to space limitations, we present a small extract of our experimental evaluation. All experiments are executed on a server with Intel Core i7-1355U CPU with 16 GB of main memory. Our system is implemented in Python and uses OpenAI's GPT model series.

We compare our system to multiple prior systems that support semantic filtering. ThalamusDB [3] is a semantic query processing engine that handles multimodal data. It employs approximate query processing (AQP) to enable users to balance query quality and execution costs. LOTUS [6] is a query engine that uses large language models (LLMs) to process both structured and unstructured data through a declarative programming interface. It optimizes execution by approximating a predefined "gold algorithm" that implements semantic operators with LLMs, reducing cost and latency while maintaining accuracy guarantees relative to that gold standard. We also compare against an embedding-based approach using CLIP embeddings [7], which are vector representations that map images and text into a shared semantic embedding space. Because this approach produces a similarity ranking between images and text queries, we evaluate performance across different similarity thresholds and report results for the threshold that yields the best score.

We evaluate all baselines on the STL-10 dataset, an image classification benchmark comprising 10 object categories, and present the results in Figure 2. We consider nine queries of varying complexity, including object presence, negation, and logical compositions. For each query, we evaluate all systems under three optimization objectives: prioritizing precision, recall, and cost. Based on their performance under each objective, systems are ranked from 1 (best) to 6 (worst). This ranking provides a comparative view of how each method performs across different cost–accuracy trade-offs. Our system consistently performs well across all queries, achieving the top rank for most complex predicates (Q6–Q9) while remaining competitive on simpler ones (Q1–Q3). Notably, it is never more than one rank behind the optimal worst-case rank, highlighting its robustness and consistent performance across diverse queries.

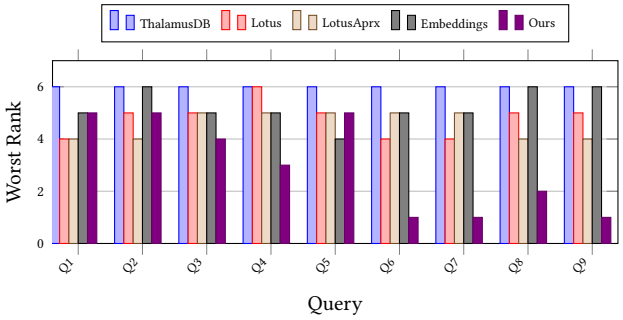


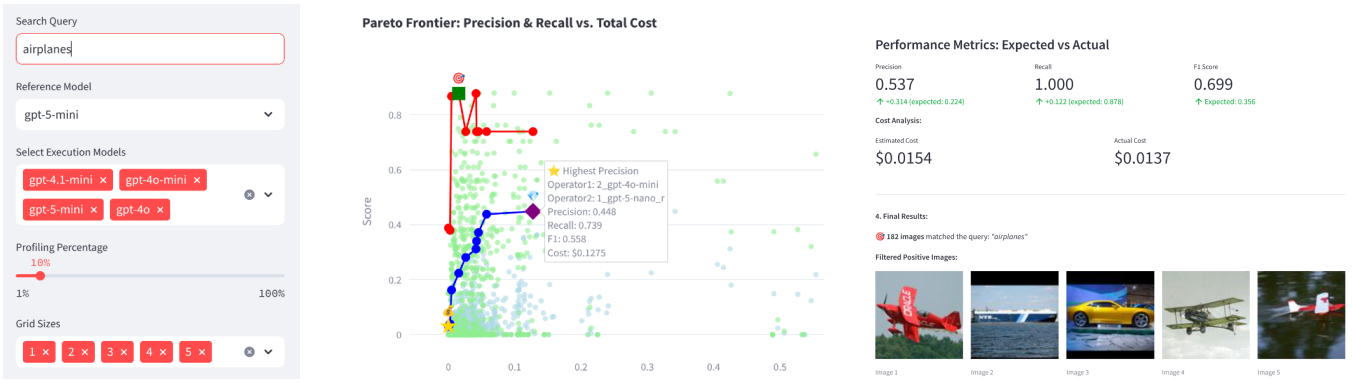
Figure 2: Worst rank achieved by each system across the benchmark queries. Lower ranks indicate better performance.

4 DEMONSTRATION SETUP

A live demonstration will be presented through a web-based interface. The system takes as input an image collection and a natural language predicate specified by the user through the configuration interface shown in Figure 3a. Users select an image dataset (STL-10, Fashion, COCO, FSC147), which cover a diverse range of visual scenarios. They may specify predicates ranging from simple object presence to more complex conditions involving negation or logical operators (e.g., 'images of animals excluding cats'). To configure the semantic filter operator, users can select a reference model (used to generate pseudo ground truth, e.g., GPT-5), execution models, grid sizes for the scan operator (ranging from 1×1 to 10×10) and adjust a profiling percentage that determines the fraction of the dataset used for sampling.

Once configured, users can initiate the profiling process by clicking *Start Profiling*. During profiling, the system samples images, invokes the reference model to generate pseudo ground truth labels, and profiles the selected execution models across different grid configurations. The interface displays model invocations, image grids, and aggregate statistics, including runtime, cost, and profiling progress

After profiling, users can initiate the optimization phase by clicking *Optimization*. The system evaluates candidate operator schedules and visualizes the resulting cost-quality trade-offs using an interactive Pareto frontier as shown in Figure 3b. The x-axis represents cost, while the y-axis represents quality metrics (e.g., precision or recall). Each point corresponds to an evaluated operator configuration. The plot also highlights key configurations, including those achieving the highest precision, highest recall, and lowest cost. To support user-driven optimization, the interface allows users to specify weights for precision, recall, and cost via a sidebar. These weights reflect the relative importance of each metric for the target application. After setting preferences, users can click *Find Operators* to obtain the top recommended operator configuration. The system also displays the top-ranked configurations, along with their expected precision, recall, and cost.



(a) Part of the configuration interface used by users.

(b) Pareto frontier illustrating precision–recall–cost trade-offs across operator configurations.

(c) Execution results showing expected vs. actual metrics and filtered images.

Figure 3: Interactive demonstration of the semantic filtering system. Users configure queries, explore trade-offs between accuracy and cost, and inspect execution results.

Finally, users can initiate the execution phase by clicking *Start Execution*, which evaluates the selected operator schedule over the remaining portion of the dataset. The interface displays intermediate filtering results in real time and summarizes execution statistics, including runtime and cost as shown in Figure 3c. In addition, users can compare the expected performance and cost (as estimated during profiling) with the actual execution results. Users can further explore the results by scrolling through the interface to view all images that satisfy the specified natural language predicate.

5 RELATED WORK

With the growing adoption of large language models (LLMs) across a wide range of applications, semantic query processing engines (SQPEs) have emerged in both academia [3–6, 8] and industry [1, 2]. These systems extend traditional query processing by introducing semantic operators that evaluate predicates expressed in natural language over diverse data types, including both structured data and unstructured modalities such as images, audio, and text. In contrast to prior work that focuses on end-to-end system design, we introduce a semantic filter operator that can be embedded into existing query processing engines. Our operator exposes tunable configurations and enables cost-aware execution. More recent work uses embedding-based methods such as CLIP [7] to map images and textual queries into a shared embedding space, enabling similarity-based retrieval. While these embeddings capture coarse semantic relationships, they often struggle with compositional queries, counting, and nuanced semantic conditions, motivating the use of more expressive but costly LLM-based operators for semantic query processing.

6 CONCLUSION

The proposed demonstration focuses on a system that proposes tunable accuracy-cost trade-offs for large-scale semantic filtering. Visitors will be able to interact with the system using custom search queries on the available data sets and observe the influence of the tuning parameters on the performance metrics.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Award No. 2239326.

REFERENCES

[1] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, and Philipp Unterbrunner. 2016. The Snowflake Elastic Data Warehouse. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 215–226. https://doi.org/10.1145/2882903.2903741

[2] Sérgio Fernandes and Jorge Bernardino. 2015. What is BigQuery?. In *Proceedings of the 19th International Database Engineering & Applications Symposium* (Yokohama, Japan) *(IDEAS '15)*. Association for Computing Machinery, New York, NY, USA, 202–203. https://doi.org/10.1145/2790755.2790797

[3] Saehan Jo and Immanuel Trummer. 2024. Thalamusdb: Approximate query processing on multi-modal data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–26.

[4] Gaurav Tarlok Kakkar, Jiashen Cao, Pramod Chunduri, Zhuangdi Xu, Suryatej Reddy Vyalla, Prashanth Dintyala, Anirudh Prabakaran, Jaeho Bang, Aubhro Sengupta, Kaushik Ravichandran, Ishwarya Sivakumar, Aryan Rajoria, Ashmita Raju, Tushar Aggarwal, Abdullah Shah, Sanjana Garg, Shashank Suman, Myna Prasanna Kalluraya, Subrata Mitra, Ali Payani, Yao Lu, Umakishore Ramachandran, and Joy Arulraj. 2023. EVA: An End-to-End Exploratory Video Analytics System. In *Proceedings of the Seventh Workshop on Data Management for End-to-End Machine Learning* (Seattle, WA, USA) *(DEEM '23)*. Association for Computing Machinery, New York, NY, USA, Article 8, 5 pages. https://doi.org/10.1145/3595360.3595858

[5] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, and Gerardo Vitagliano. 2024. A Declarative System for Optimizing AI Workloads. arXiv:2405.14696 [cs.CL] https://arxiv.org/abs/2405.14696

[6] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. arXiv:2407.11418 [cs.DB] https://arxiv.org/abs/2407.11418

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV] https://arxiv.org/abs/2103.00020

[8] Matthias Urban and Carsten Binnig. 2023. CAESURA: Language Models as Multi-Modal Query Planners. arXiv:2308.03424 [cs.DB] https://arxiv.org/abs/2308.03424