Semantic Caching for Low-Cost LLM Serving: From Offline Learning to Online Adaptation

Xutong Liu*[1], Baran Atalar*[2], Xiangxiang Dai†[3], Jinhang Zuo[4], Siwei Wang[5],
John C.S. Lui[3], Wei Chen[5], Carlee Joe-Wong[2]
[1]University of Washington, [2]Carnegie Mellon University, [3]The Chinese University of Hong Kong,
[4]City University of Hong Kong, [5]Microsoft Research

***Abstract*—Large Language Models (LLMs) are revolutionizing how users interact with information systems, yet their high inference cost poses serious scalability and sustainability challenges. Caching inference responses, allowing them to be retrieved without another forward pass through the LLM, has emerged as one possible solution. Traditional exact-match caching, however, overlooks the semantic similarity between queries, leading to unnecessary recomputation. Semantic caching addresses this by retrieving responses based on semantic similarity, but introduces a fundamentally different cache eviction problem: one must account for mismatch costs between incoming queries and cached responses. Moreover, key system parameters, such as query arrival probabilities and serving costs, are often unknown and must be learned over time. Existing semantic caching methods are largely ad-hoc, lacking theoretical foundations and unable to adapt to real-world uncertainty. In this paper, we present a principled, learning-based framework for semantic cache eviction under unknown query and cost distributions. We formulate both offline optimization and online learning variants of the problem based on the combinatorial multi-armed bandit framework, and develop provably efficient algorithms with state-of-the-art guarantees. We also evaluate our framework on a synthetic dataset, showing that our proposed algorithms perform matching or superior performance compared with baselines.**

I. Introduction

The emergence of large language models (LLMs) such as GPT-4 and Gemini has enabled remarkable advances in natural language understanding, reasoning, and multimodal capabilities, powering applications ranging from coding assistants to conversational search engines. Yet this power comes at a steep cost: each query can require billions of floating point operations, making LLM inference orders of magnitude more expensive than conventional web queries [1]. As usage continues to rise, service providers face growing concerns over latency, compute efficiency, and energy consumption.

A natural way to mitigate these costs is through caching—reusing previously computed responses when similar queries reappear. However, most existing LLM caching frameworks are built on exact string or token matching [2], [3], which are ill-suited to LLM workloads. Semantically equivalent queries such as "What is LLM caching?" and "How does caching work in large language models?" are treated as completely different entries, resulting in frequent cache misses even when a suitable response exists. Recent research shows that over 30% of LLM queries are semantically similar [4]. This exposes a major inefficiency: today's cache systems are blind to semantic similarity and conversational context.

To address this gap, recent work has explored the problem of semantic caching, i.e., designing caching systems that reason about the meaning and context of queries, rather than their exact text [4]–[7]. For example, GPTCache implements an industry-scale semantic caching framework by embedding prompts into vector spaces and storing responses in a vector database for efficient similarity-based retrieval. Building on this foundation, follow-up work has incorporated student–teacher models [6], hierarchical clustering [7], and federated learning [4] to further improve hit rates and cost savings.

Despite these advances, there still exist limitations that can be significantly improved. First, most existing systems assume that query arrival distributions and response costs are known a priori—an assumption that rarely holds in real and uncertain deployment environments. Second, many current approaches are ad hoc and data-driven, lacking theoretical foundations and rigorous performance guarantees that are essential for understanding and optimizing semantic caching at scale. These gaps lead to a fundamental open question: *Can we build a principled semantic caching framework that optimizes the total cost under uncertainty, with provable performance guarantees?*

A. Our Contributions

To answer this question, our contributions are as follows:
(1) Semantic Caching Model: We introduce a novel semantic caching framework that generalizes traditional exact-match caching by incorporating a *mismatch cost*, which models potential utility loss when serving semantically similar, but not identical, responses. This introduces a new algorithmic challenge: balancing the mismatch cost against the *serving cost* incurred by querying the LLM for a fresh response. We formalize this trade-off via a unified loss function that supports *arbitrary distance metrics* over queries. Based on this model, we systematically study three settings of increasing uncertainty: (i) the *oracle setting*, where both query arrival probabilities and serving costs are known; (ii) the *offline learning setting*, where these parameters are unknown but learned from a static dataset; and (iii) the *online adaptive setting*, where the agent learns and adapts in real time from partial feedback.

*Xutong Liu and Baran Atalar are co-first authors. †Xiangxiang Dai is the corresponding author. The work of Carlee Joe-Wong is supported by NSF grant 2312761 and Department of Energy grant DESC0025652. The work of John C.S. Lui is supported in part by RGC GRF-14202923. Xiangxiang Dai is supported by the National Natural Science Foundation of China (625B2163).

(2) Algorithm Design: For the *oracle setting*, we show that computing the optimal cache is NP-hard even with full knowledge of parameters. We then prove that the loss function is non-increasing and supermodular, and design the `Reverse Greedy` algorithm with provable approximation guarantees. For the *offline learning setting*, where serving costs and query arrival probabilities are unknown but historical data is available (e.g., logs or pre-deployment feedback [2], [3]), we develop a pessimistic learning algorithm, `CUCB-SC`. It estimates the parameters while penalizing uncertainty due to limited historical data, and integrates with `Reverse Greedy` to yield robust cache decisions with finite-sample guarantees. For the *online adaptive setting*, we face a unique challenge: balancing exploration and exploitation under a *low-switching constraint*, since each cache update incurs additional serving cost. We propose `CLCB-SC-LS`, which combines stage-based cache switching with the principle of optimism in the face of uncertainty, and is provably efficient with minimal switching.
(3) Theoretical Guarantees: We provide rigorous theoretical results across all three settings grounding to combinatorial multi-armed bandits frameworks. In the oracle setting, we show our algorithm achieves near-optimal approximation with low time complexity, and can be improved to exact optimality with increased computation. In the offline setting, we prove the suboptimality gap of our algorithm scales as $\tilde{O}(\sqrt{m/n})$, where m is the number of queries, n is the number of samples, and $\tilde{O}$ ignores the log factors. In the online setting, we prove a regret bound of $\tilde{O}(\sqrt{mT})$ over T rounds, with only $O(m \log \log T)$ cache switches. Notably, our offline and online algorithms improve upon prior methods [2] and match state-of-the-art performance in the special case of exact-match caching [3].
(4) Performance Evaluation: We conduct extensive experiments on synthetic datasets for offline and online settings. We show that the reverse greedy algorithm closely approximates the optimal cache with minimal loss. In the online setting, our algorithms consistently outperform baselines regarding average regret, with at least 11.75% improvements observed across varying cache sizes, query distributions, and time horizons. Moreover, `CLCB-SC-LS` achieves the lowest number of cache switches and running time (reducing up to 85%) while maintaining strong regret results.

B. Related Work

Low-Cost LLM Serving. There have been tremendous efforts recently to reduce the inference cost and latency of large language model (LLM) serving, through approaches such as model quantization [8], [9], speculative decoding [10], [11], and cloud-edge offloading [12], [13]. A particularly promising line of work leverages caching to trade memory for cost or latency reduction. Existing studies explore caching at various levels: attention-level (KV-cache) [5], [14]–[16], query-level [2]–[4], [7], [17], and model/API-level [18]–[20]. Our work falls under the query-level LLM caching category. Unlike prior query-level semantic caching studies [4], [7], [17] that focus primarily on data-driven heuristics or assume known system parameters, our work provides the first unified treatment of semantic caching under uncertainty based on the combinatorial multi-armed bandits framework [3], [21]–[23], with provable theoretical guarantees in both offline and online learning settings.

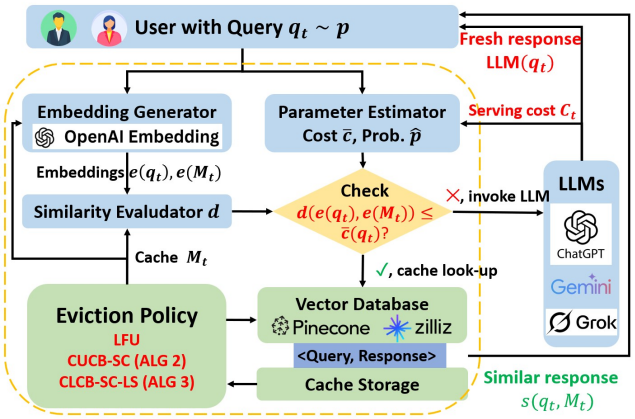


Fig. 1: Semantic Caching System for Low-Cost LLM Serving

Offline Optimization and Online Learning for Caching. Classical cache eviction algorithms assume known query arrival probabilities and costs. For non-uniform arrivals, policies such as Least Frequently Used (LFU) and Least Recently Used (LRU) are commonly employed [24]. When both arrival rates and costs vary, the Least Expected Cost (LEC) policy was proposed [25]. More recent work has examined statistical learning for caching in both offline and online contexts [2], [3], [26], but these efforts are largely limited to exact-match caching. We extend this line of work to semantic caching under unknown parameters, which enables us to use responses from semantically similar queries in the cache. Our algorithm design and theoretical results recover prior results [2] in the degenerate (exact match) case. Finally, while there is a line of work that studies similarity caching [26]–[29], their formulation differs in both the loss function and algorithmic design, and does not explicitly control cache switching costs as we do.

II. System Model

In this section, we present the system model of the semantic caching problem for LLM serving, as shown in Fig. 1.

A. LLM Serving System

We consider an LLM serving system with the following core components.
Queries and Embeddings. Let $\mathcal{Q} = \{q_1, \ldots, q_m\}$ denote a finite set of $m \in \mathbb{Z}_+$ distinct queries, where each query $q_i \in \mathcal{X}$ is a natural language prompt (e.g., "What is LLM caching?") drawn from a language space $\mathcal{X}$. The system operator (i.e., learning agent) has access to a large language model (`LLM`) (e.g., ChatGPT) and an associated embedding generator $e : \mathcal{X} \to \mathbb{R}^{d_e}$ (e.g., OpenAI Embedding API), which maps each query into a d_e-dimensional vector representation.
Arrival Probabilities and Serving Costs. Each query $q \in \mathcal{Q}$ is associated with an unknown expected serving cost $c(q) \in (0, 1]$ and an unknown query arrival probability $p(q) \in (0, 1]$, such that $\sum_{q \in \mathcal{Q}} p(q) = 1$. For example, this cost could represent

the latency or the computational resources expended from a forward pass through the LLM. At each round t, whenever the agent invokes the LLM to serve a query q, it incurs a random cost $C_t(q) = c(q) + \epsilon_t(q)$, where $\epsilon_t(q)$ is zero-mean sub-Gaussian noise capturing cost variability (which is due to fluctuating latency or random output token length) [2].

Similarity Distance and Mismatch Costs. We define a similarity distance function $d : \mathcal{Q} \times \mathcal{Q} \to \mathbb{R}_+$, where a smaller value indicates stronger semantic similarity (e.g., the distance between "What is LLM caching?" and "How does caching work in large language models?" is small). We adopt the Euclidean distance as the default similarity metric: $d(q_1, q_2) = \|e(q_1) - e(q_2)\|_2$, yet we can generalize to *any metric*, such as cosine distance $d(q_1, q_2) = e(q_1)^\top e(q_2)/(\|e(q_1)\|_2 \|e(q_2)\|_2)$, without changing the results of the current work.

We further define the distance from a query q to a set $\mathcal{M} \subseteq \mathcal{Q}$ as $d(q, \mathcal{M}) \coloneqq \min_{u\in\mathcal{M}} d(q, u)$, and let $s(q, \mathcal{M}) \coloneqq \operatorname{argmin}_{u\in\mathcal{M}} d(q, u)$ denote the closest cached query to q.

When a query q is served, the agent may either invoke the LLM to generate a fresh response $a(q)$, or reuse a cached response $a(u)$ from a similar query $u \in \mathcal{M}$, where $\mathcal{M}$ represents the set of cached queries and their responses. Reusing a cached response incurs an expected *mismatch cost* $\gamma \cdot d(q, u)$, where $\gamma > 0$ is a scaling parameter. Since our results hold for any distance function d, we normalize this cost such that (i) the fresh response of q incurs zero expected mismatch cost, and (ii) we set $\gamma = 1$ without loss of generality.

B. Semantic Caching for Low-cost LLM Serving

To reduce inference cost, the system maintains a semantic cache $\mathcal{M}$ containing at most k query-response pairs:

$$\mathcal{M} = \{(q_1, a(q_1)), \ldots, (q_k, a(q_k))\},$$

stored in a vector database [30]. Let $\boldsymbol{p} = (p(q_1), \ldots, p(q_m))$ and $\boldsymbol{c} = (c(q_1), \ldots, c(q_m))$ denote the arrival probability and serving cost vectors, respectively.

In each round t, a query $q_t \sim \boldsymbol{p}$ arrives. The agent can decide between two actions:

- **Cache Lookup:** Return a cached response $a(q)$ for some $q \in \mathcal{M}$, incurring a mismatch cost $d(q_t, q)$;
- **LLM Query:** Invoke the LLM to generate a fresh response $a(q_t)$, incurring the expected cost $c(q_t)$.

Obviously, the optimal decision rule is to compare the cost of querying the LLM $c(q)$ with the mismatch cost $d(q_t, \mathcal{M})$ incurred by using the most similar cached query $s(q_t, \mathcal{M}) = \operatorname{argmin}_{u\in\mathcal{M}} d(q_t, u)$:

$$a_t = \begin{cases} \texttt{LLM}(q_t) & \text{if } c(q_t) \le d(q_t, \mathcal{M}), \\ a(s(q_t, \mathcal{M})) & \text{otherwise.} \end{cases} \tag{1}$$

This rule then raises the question: *how should we determine the query-response pairs stored in the cache?* In the remainder of the paper, we seek to answer this question.

Given the random query arrivals, the expected loss incurred by a cache $\mathcal{M}$ is defined as:

$$\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d) = \textstyle\sum_{q\in\mathcal{Q}} p(q) \cdot \min\{c(q), d(q, \mathcal{M})\} \tag{2}$$

when using the optimal decision rule in (1). Our goal is to find optimal $\mathcal{M}^*$ of size at most k that minimizes this loss:

$$\mathcal{M}^* = \operatorname{argmin}_{\mathcal{M}\subseteq\mathcal{Q}, |\mathcal{M}|\le k} \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d). \tag{3}$$

Remark 1 (Special Cases of the distance function d)**.** Our framework captures a wide range of semantic and structural caching models by varying the distance function d:

(i) Bipartite Graph Coverage. Consider a threshold-based distance function for some $\epsilon > 0$:

$$d(q_1, q_2) = \begin{cases} 0 & \text{if } \|e(q_1) - e(q_2)\|_2 \le \epsilon, \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

This induces a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ where $\mathcal{U} = \mathcal{V} = \mathcal{Q}$ and $(u, v) \in \mathcal{E}$ if and only if $d(u, v) = 0$. Define $N(\mathcal{M}) = \{v \in \mathcal{V} : \exists\, u \in \mathcal{M} \text{ s.t. } (u, v) \in \mathcal{E}\}$ as the set of covered nodes. Then the general loss as in Eq. (2) becomes:

$$\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d) = \textstyle\sum_{v\notin N(\mathcal{M})} p(v)c(v). \tag{5}$$

This special case is used in proving NP-hardness (Lemma 1).

(ii) Exact-Match Cache. Setting $\epsilon = 0$ in the threshold distance of Eq. (4) reduces the model to exact-match caching as in prior work [2], [3], where only identical queries match. The loss simplifies to $\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d) = \sum_{q\notin\mathcal{M}} p(q)c(q)$.

In later sections, we conduct a systematic study on how to find $\mathcal{M}^*$ under different settings of increasing uncertainty:

1) **Oracle setting (Section III):** $\boldsymbol{p}$ and $\boldsymbol{c}$ are known a priori;
2) **Offline learning setting (Section IV):** an offline dataset is used to estimate the unknown $\boldsymbol{p}$ and $\boldsymbol{c}$;
3) **Online learning setting (Section V):** the agent can adaptively change the cache in each round, and the unknown $\boldsymbol{p}, \boldsymbol{c}$ are learned through sequential user interactions.

III. Approximate Solution for Semantic Caching with Known Parameters

In this section, we study the semantic caching problem when the system parameters are known a priori. We first establish the computational hardness of the problem Eq. (3), even for a special case of the general loss:

Lemma 1 (NP-hardness)**.** *It is NP-hard to compute the optimal cache $\mathcal{M}^*$ defined in* (5)*.*

Given this computational hardness, a natural question is whether a good approximate solution can be obtained efficiently. We answer this in the affirmative by leveraging the non-increasing and supermodular property of the loss function, which enables the use of greedy algorithms with provable guarantees. Intuitively, non-increasing means that as we cache more query-answer pairs, the expected loss decreases; supermodularity further implies that the marginal benefit of caching an additional item diminishes as the cache grows.

Lemma 2 (Loss Function Properties)**.** *The loss function $\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d)$ is non-increasing and supermodular regarding the set $\mathcal{M}$. That is, for any $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{Q}$ and $q \in \mathcal{Q}\backslash\mathcal{B}$, we have: (i) $\ell(\mathcal{B}; \boldsymbol{p}, \boldsymbol{c}, d) \le \ell(\mathcal{A}; \boldsymbol{p}, \boldsymbol{c}, d)$, and (ii) $\ell(\mathcal{A} \cup \{q\}; \boldsymbol{p}, \boldsymbol{c}, d) - \ell(\mathcal{A}; \boldsymbol{p}, \boldsymbol{c}, d) \le \ell(\mathcal{B} \cup \{q\}; \boldsymbol{p}, \boldsymbol{c}, d) - \ell(\mathcal{B}; \boldsymbol{p}, \boldsymbol{c}, d)$.*

Algorithm 1 `Reverse_Greedy` [31]: Removing Least Beneficial Cache Items

Input: Queries $\mathcal{Q}$, probabilities $\boldsymbol{p}$, serving costs $\boldsymbol{c}$, distance function d, cache size k
Initialize: $\mathcal{M}_0 \leftarrow \mathcal{Q}$
for $i = 1$ to $m - k$ **do**
  Compute $\ell(\mathcal{M}_{i-1} \setminus \{q\}; \boldsymbol{p}, \boldsymbol{c}, d)$ for all $q \in \mathcal{M}_{i-1}$
  Select $q_i = \operatorname{argmin}_{q \in \mathcal{M}_{i-1}} \ell(\mathcal{M}_{i-1} \setminus \{q\}; \boldsymbol{p}, \boldsymbol{c}, d)$
  Update $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \setminus \{q_i\}$
end for
Return: $\mathcal{M} = \mathcal{M}_{m-k}$

Algorithm 2 `CUCB-SC`: Combinatorial Upper Confidence Bound Algorithm for the Semantic Caching Problem

Input: Dataset $\mathcal{D} = \{(\mathcal{M}_t, q_t, C_t)\}_{t=1}^n$, queries $\mathcal{Q}$, solver `Reverse_Greedy`, embedding generator e, distance function d, probability δ.
for query $q \in \mathcal{Q}$ **do**
  Calculate counters $N(q) = \sum_{t=1}^n \mathbb{I}\{q = q_t\}$ and $N_c(q) = \sum_{t=1}^n \mathbb{I}\{q = q_t \text{ and } C_t \neq \emptyset\}$.
  Calculate empirical means $\hat{p}(q) = N(q)/n$ and $\hat{c}(q) = \sum_{t \in [n]} \mathbb{I}\{q = q_t \text{ and } C_t \neq \emptyset\} C_t / N_c(q)$.
  Calculate UCB of the cost $\bar{c}(q) = \hat{c}(q) + \sqrt{\frac{\log(\frac{6mn}{\delta})}{2N_c(q)}}$.
end for
Compute $\hat{\mathcal{A}} = \texttt{Reverse\_Greedy}\,(\mathcal{Q}, \hat{\boldsymbol{p}}, \bar{\boldsymbol{c}}, d, k)$.
Call `LLM` to obtain responses $a(q)$ for $q \in \hat{\mathcal{A}}$.
Return: $\hat{\mathcal{M}} = \{q, a(q)\}_{q \in \hat{\mathcal{A}}}$.

Based on this supermodularity property, we design a *reverse greedy* algorithm in Algorithm 1. Unlike standard greedy algorithms that add items from scratch, Algorithm 1 starts by assuming all query-answer pairs are cached and then iteratively removes the least useful cached item until only k remain.

We now provide a theoretical guarantee on the performance of Alg. 1 following [31]. Let us define the curvature $c = 1 - \min_{q \in \mathcal{Q}} \frac{\ell(\mathcal{Q} \setminus \{q\}; \boldsymbol{p}, \boldsymbol{c}, d) - \ell(\mathcal{Q}; \boldsymbol{p}, \boldsymbol{c}, d)}{\ell(\emptyset; \boldsymbol{p}, \boldsymbol{c}, d) - \ell(\{q\}; \boldsymbol{p}, \boldsymbol{c}, d)}$, measuring how much the function deviates from the additive function (whose c is 0).

Theorem 1 (Approximation Guarantee [31])**.** *Let $\mathcal{M}$ be the solution returned by Algorithm 1. Then, $\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d) \leq \frac{e^\beta - 1}{\beta} \cdot \ell(\mathcal{M}^*; \boldsymbol{p}, \boldsymbol{c}, d)$, where $\beta = \frac{c}{1-c}$ and $c \in [0, 1]$ is the curvature.*

IV. Offline Learning for the Semantic Caching Problem with Unknown Parameters

In this section, we consider the setting where the cost and arrival probabilities are unknown and must be learned from offline data. We first introduce the formal offline learning setup, then present the `CUCB-SC` algorithm in Algorithm 2 for the semantic caching problem, and finally provide its theoretical analysis and implications.

A. Offline Learning Setup

We assume access to a historical dataset $\mathcal{D} = (\mathcal{M}_t, q_t, C_t)_{t=1}^n$ collected by an experimenter, such as logs of user interactions with an LLM service. where $\mathcal{M}_t$ is the selected cache at round t, q_t is the user query, and $C_t = C_t(q_t)$ is the observed cost. If $C_t = \emptyset$, it indicates that the cached response from the nearest neighbor in $\mathcal{M}_t$ was used and no cost was observed. If $C_t > 0$, the LLM was invoked and the cost feedback was recorded. Define $\nu(q) = \Pr[C_t(q_t) \neq \emptyset \mid q_t = q] > 0$ as the probability of observing the cost feedback for query q.

Since computing the optimal cache is NP-hard even with known parameters (Lemma 1), our goal is to learn an approximate solution that competes with the α-approximate oracle defined given by Algorithm 1. Specifically, we aim to minimize the approximate suboptimality:

$$\text{SubOpt}(\mathcal{M}) = \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d) - \alpha \cdot \ell(\mathcal{M}^*; \boldsymbol{p}, \boldsymbol{c}, d), \quad (6)$$

where α is the approximation ratio from Theorem 1.

B. Algorithm Design of `CUCB-SC`

The `CUCB-SC` algorithm (shown in Algorithm 2) proceeds in two key steps. First, it computes high probability estimates of the arrival probabilities $\boldsymbol{p}$ and the serving costs $\boldsymbol{c}$. For the serving costs, we use the upper confidence bounds (UCB) as pessimistic estimations of the true query costs. Specifically, we penalize each cost estimation $\hat{c}(q)$ by its confidence interval, $\sqrt{\log(\frac{6mn}{\delta})/2N_c(q)}$ in Line 5. This approach, rooted in the pessimism principle [32], mitigates the impact of high fluctuations in empirical estimates caused by limited observations, effectively addressing the uncertainty inherent in passively collected data. For the arrival probabilities, we leverage the full-feedback nature of the dataset and use empirical means $\hat{p}(q)$. Secondly, we invoke the reverse greedy algorithm, which iteratively removes the least beneficial query-response pair based on estimated $\hat{\boldsymbol{p}}$ and $\bar{\boldsymbol{c}}$, to select the cache $\hat{\mathcal{M}}$ in Line 9.

C. Theoretical Result and Its Discussion

We now present the performance guarantee for `CUCB-SC`.

Theorem 2 (Suboptimality Gap Bound for Algorithm 2)**.** *Let $\mathcal{D}$ be an offline dataset of n samples. Suppose $n \geq \frac{8 \log(1/\delta)}{\min_{q \in \mathcal{Q}} p(q)\nu(q)}$. Then with probability of at least $1-\delta$, the cache $\hat{\mathcal{M}}$ returned by `CUCB-SC` satisfies: SubOpt$(\hat{\mathcal{M}}) \leq 4\sqrt{\frac{2\sum_{q \in \mathcal{Q}} \frac{1}{\nu(q)} \log(\frac{6mn}{\delta})}{n}}$.*

Remark 2 (Discussion of Theorem 2)**.** According to Theorem 2, the suboptimality gap decreases at a rate of $\sqrt{1/n}$ with respect to the number of offline data samples n. In addition, the gap scales proportionally with $\sqrt{\sum_{q \in \mathcal{Q}} 1/\nu(q)}$, where $\nu(q)$ can be interpreted as the data generation rate for query q. In the special case where $\nu(q) = 1$ for all q meanwhile the cache operates in an exact-match setting (see Remark 1.(ii)), our formulation reduces to the setting in Zhu et al. [2]. Under this regime, our result improves upon theirs by a factor of $O(k/\sqrt{C_1})$, where $C_1 = \min_{q \in \mathcal{Q}} c(q)$ is the minimal query cost. This improvement arises from our separate treatment of arrival and cost distributions, leveraging their distinct feedback structures, in contrast to the coupled treatment in Zhu et al. [2]. More recently, Liu et al. [3] achieve a similar improvement

while also operating in the degenerate exact-match cache setting. Our work not only matches their theoretical guarantees but also extends them to the more general semantic caching setting.

D. Suboptimal Gap Analysis

To establish the result in Theorem 2, we begin with the following concentration bounds.

Lemma 3 (Concentration of Estimates [3])**.** *For any $\delta > 0$, define the following events: $\mathcal{E}_{arvl} := \left\{\sum_{q\in\mathcal{Q}} |\hat{p}(q) - p(q)| \le \sqrt{\frac{2m\log(2/\delta)}{n}}\right\}$, $\mathcal{E}_{cost} := \left\{|\hat{c}(q) - c(q)| \le \sqrt{\frac{\log(2mn/\delta)}{2N_c(q)}}, \ \forall q \in \mathcal{Q}\right\}$, $\mathcal{E}_{counter} := \left\{N_c(q) \ge \frac{n\cdot p(q)\nu(q)}{2}, \ \forall q \in \mathcal{Q}\right\}$. Then, assuming $n \ge \frac{8\log(m/\delta)}{\min_q p(q)\nu(q)}$, all three events occur with probability at least $1-\delta$.*

Given arm-level estimation errors in Lemma 3, we next proceed to relate the arm-level error to the cache-level suboptimal gap. Specifically, we bound the sensitivity of the loss function to perturbations in its parameters.

Lemma 4 (Loss Function Sensitivity)**.** *For any cost parameters $\boldsymbol{c}, \boldsymbol{c}' \in [0,1]^m$, arrival probabilities $\boldsymbol{p}, \boldsymbol{p}' \in [0,1]^m$, and any distance function $d : \mathcal{Q}\times\mathcal{Q} \to [0,1]$, the loss function satisfies that for any cache $\mathcal{M} \subseteq \mathcal{Q}$, $|\ell(\mathcal{M};\boldsymbol{c},\boldsymbol{p},d) - \ell(\mathcal{M};\boldsymbol{c}',\boldsymbol{p}',d)| \le \sum_{q\in Q}|p(q) - p'(q)| + \sum_{q\in Q} p(q)|c(q) - c(q')|$.*

Proof of Theorem 2. We ignore d in the loss function $\ell(\mathcal{M};\boldsymbol{p},\boldsymbol{c},d) = \ell(\mathcal{M};\boldsymbol{p},\boldsymbol{c})$ when contexts are clear. Under the events of $\mathcal{E}_{\text{arvl}}, \mathcal{E}_{\text{cost}}, \mathcal{E}_{\text{counter}}$, $\ell(\hat{\mathcal{M}};\boldsymbol{c},\boldsymbol{p}) - \alpha\ell(\mathcal{M}^*;\boldsymbol{c},\boldsymbol{p}) = \underbrace{\alpha\ell(\mathcal{M}^*;\bar{\boldsymbol{c}},\hat{\boldsymbol{p}}) - \alpha\ell(\mathcal{M}^*;\boldsymbol{c},\boldsymbol{p})}_{\text{estimation gap}} + \underbrace{\ell(\hat{\mathcal{M}};\bar{\boldsymbol{c}},\hat{\boldsymbol{p}}) - \alpha\ell(\mathcal{M}^*;\bar{\boldsymbol{c}},\hat{\boldsymbol{p}})}_{\text{greedy gap}} + \underbrace{\ell(\hat{\mathcal{M}};\boldsymbol{c},\boldsymbol{p}) - \ell(\hat{\mathcal{M}};\bar{\boldsymbol{c}},\hat{\boldsymbol{p}})}_{\text{pessimism gap}} \le \alpha\ell(\mathcal{M}^*;\bar{\boldsymbol{c}},\hat{\boldsymbol{p}}) - \alpha\ell(\mathcal{M}^*;\boldsymbol{c},\boldsymbol{p}) + \ell(\hat{\mathcal{M}};\bar{\boldsymbol{c}},\boldsymbol{p}) - \ell(\hat{\mathcal{M}};\bar{\boldsymbol{c}},\hat{\boldsymbol{p}}) \le \sum_{q\in\mathcal{Q}} p(q)|\bar{c}(q) - c(q)| + 2\|\hat{\boldsymbol{p}} - \boldsymbol{p}\|_1 \le \sum_{q\in\mathcal{Q}} p(q)|\bar{c}(q) - c(q)| + 2\sqrt{\frac{2m\log(\frac{2}{\delta})}{n}}$, where the equality is due to adding and subtracting $\alpha\ell(\hat{\mathcal{M}}^*;\bar{\boldsymbol{c}},\hat{\boldsymbol{p}})$ and $\ell(\hat{\mathcal{M}};\bar{\boldsymbol{c}},\hat{\boldsymbol{p}})$, the first inequality is due to the greedy gap ≤ 0 by the approximation ratio guarantee of the `Reverse Greedy`, $\bar{\boldsymbol{c}} \ge \boldsymbol{c}$ by Lemma 3, and $\ell(\mathcal{M};\boldsymbol{p},\boldsymbol{c})$ is monotone regarding $\boldsymbol{c}$, the second inequality is due to Lemma 4, and the last inequality is due to Lemma 3.

Then we have for the first term: $\sum_{q\in\mathcal{Q}} p(q)\,|\bar{c}(q) - c(q)| \le 2\sum_{q\in\mathcal{Q}} p(q)\sqrt{\frac{\log(2mn/\delta)}{2N_c(q)}} \le 2\sum_{q\in\mathcal{Q}} p(q)\sqrt{\frac{\log(2mn/\delta)}{n\cdot p(q)\nu(q)}} \le 2\sqrt{\sum_{q\in\mathcal{Q}} p(q)}\sqrt{\sum_{q\in\mathcal{Q}}\frac{p(q)\log(2mn/\delta)}{n\cdot p(q)\nu(q)}} = 2\sqrt{\frac{\sum_{q\in\mathcal{Q}}\frac{1}{\nu(q)}\log(2mn/\delta)}{n}}$, where the first two inequalities are due to the definition of high-probability events in Lemma 3, the third inequality is due to the Cauchy-Schwarz inequality. Finally, we can set $\delta' = 1/(3\delta)$ to conclude the theorem. □

V. Online Adaptive Learning for the Semantic Caching Problem with Unknown Parameters

Now that we have developed a caching policy for the offline setting, we move to the online setting, in which we collect user data while learning the optimal cache eviction policy, and which does not require an offline dataset. For example, a deployed chatbot must decide whether to serve each incoming query from the cache or invoke the LLM at a cost for a stream of users, while continuously estimating query distributions and serving costs. We first introduce the online learning setup, then present the proposed algorithm `CLCB-SC-LS` in Algorithm 3, and finally provide theoretical guarantees and analysis.

A. Online Learning Setup

We consider a T-round sequential decision-making problem. At the beginning of each round $t \in [T]$, the system maintains a semantic cache $\mathcal{M}_t$ of size k.

A user with query $q_t \sim \boldsymbol{p}$ arrives. As described in Section II-B, the agent chooses between two actions:

- Serve q_t using the cached response of its nearest neighbor $s(q_t,\mathcal{M}_t)$, incurring a mismatch cost $d(q_t,\mathcal{M}_t)$ but receiving no feedback on the true serving cost $c(q_t)$;
- Query the LLM to obtain $a(q_t)$, incurring a realized cost $C_t(q_t)$ and observing this cost.

At the end of round t, the agent decides whether to update the cache. If $\mathcal{M}_t \neq \mathcal{M}_{t-1}$, the agent must query the LLM to fill the responses for the updated queries in $\mathcal{M}_t \setminus \mathcal{M}_{t-1}$. This models a *switching cost* incurred for updating the cache.

Learning Objective. The objective is to minimize the cumulative regret relative to the best fixed α-approximate cache in hindsight, while accounting for both serving and switching costs. Formally, letting $\mathcal{M}_0 = \emptyset$, the regret with switching cost is defined as:

$$\mathrm{Reg}(T) = \mathbb{E}\left[\sum_{t\in[T]}\ell(\mathcal{M}_t;\boldsymbol{p},\boldsymbol{c},d) + \sum_{q\in\mathcal{M}_t\setminus\mathcal{M}_{t-1}} c(q)\right] - \sum_{t\in[T]}\alpha\ell(\mathcal{M}^*;\boldsymbol{p},\boldsymbol{c},d) \tag{7}$$

where the expectation is taken on the randomness of the query arrival, the serving cost, and the cache selection.

As in standard online learning, minimizing regret requires balancing *exploration*—to acquire feedback and refine estimates of the unknown serving costs—and *exploitation*—to utilize the best-known cache to minimize loss [33]. This exploration-exploitation tradeoff is especially challenging in our setting because the agent only observes the serving cost when querying the LLM for a fresh response. Such queries can be costly, making effective exploration both essential and expensive.

B. Algorithm Design of `CLCB-SC-LS`

In this section, we present `CLCB-SC-LS`, a low-switching online learning algorithm for semantic caching The pseudocode is provided in Algorithm 3, with subroutines `Switch Cache` in Algorithm 4 and `Serve and Update` in Algorithm 5.

The algorithm proceeds in three key phases: (i) determining when to switch the cache based on the number of observations, (ii) actively interacting with the environment to explore the unknown serving costs and arrival probabilities, and (iii) serving

Algorithm 3 `CLCB-SC-LS`: Combinatorial Lower Confidence Bound Algorithm for Semantic Caching with Low Switching

1: **Input:** Queries $\mathcal{Q}$, solver `Reverse_Greedy`, embedding generator e, distance function d, rounds T, probability δ.
2: **Initialize:** For each query $q \in \mathcal{Q}$, set counter $N_{p,0}(q) = 0, N_{c,0}(q) = 0$, cumulative serving cost $L_{c,0}(q) = 0$, stage index $\tau_q = 1$, and round index sets $\mathcal{T}(q, \tau_q) = \emptyset$. And initialize $\tau_p = 1$, $\mathcal{T}(p, \tau_p) = \emptyset$.
3: **for** round $t = 1, ..., T$ **do**
4: User t arrives with query $q_t \sim \boldsymbol{p}$.
5: **if** $\exists q \in [m]$ s.t. $|\mathcal{T}(q, \tau_q)| \geq 1 + \sqrt{\frac{T \cdot \sum_{\tau=1}^{\tau_q - 1} |\mathcal{T}(q,\tau)|}{m}}$ **then**
6: Set $\tau_q = \tau_q$ + 1, $\mathcal{T}(q, \tau_q) = \emptyset$.
7: Set `Switch` = True.
8: **else if** $|\mathcal{T}(p, \tau_p)| \geq 1 + \sqrt{T \cdot \sum_{\tau=1}^{\tau_p - 1} |\mathcal{T}(p, \tau_p)|}$ **then**
9: Set $\tau_p = \tau_p + 1$, $\mathcal{T}(p, \tau_p) = 0$.
10: Set `Switch` = True.
11: **end if**
12: **if** `Switch` == True **then**
13: Call `Switch Cache` and get $\mathcal{M}_t$.
14: Set `Switch` = False
15: **else**
16: Remain $\mathcal{M}_t = \mathcal{M}_{t-1}$.
17: **end if**
18: Call `Serve and Update`
19: **end for**

user queries while collecting feedback to refine parameter estimates over time.

Key Differences from Offline Learning. Compared to the offline algorithm (`CUCB-SC`), which optimizes a single cache using a static dataset, the online setting requires the agent to explore different caches over time to learn the unknown serving costs. However, changing the cache too frequently incurs switching overhead. Thus, `CLCB-SC-LS` must both: (i) ensure sufficient exploration to estimate serving costs accurately, and (ii) limit the number of switches to control switching costs.

Stage-Based Cache Switching (Lines 5-10 of Algorithm 3). To control switching frequency while ensuring sufficient exploration, we introduce a stage-based switch mechanism for both the serving cost estimates and the arrival probabilities.

For each query $q \in \mathcal{Q}$, we partition the time slots into stages, where a new stage begins once sufficient serving cost feedback has been collected. Let $\mathcal{T}(q, \tau)$ denote the set of rounds in which query q was submitted to the LLM to get a fresh response during stage τ. Let $\tau_q(t)$ denote the current stage index of q at round t. As shown in Line 5, stage τ_q is incremented if: $|\mathcal{T}(q, \tau_q)| \geq 1 + \sqrt{\frac{T \cdot \sum_{\tau=1}^{\tau_q - 1} |\mathcal{T}(q,\tau)|}{m}}$, i.e., when the number of observations in the current stage exceeds a threshold (whose value is tuned to bound the number of switches as in Lemma 7) based on the cumulative prior observations.

Similarly, we maintain a global stage index τ_p for arrival probability estimation. This stage is incremented once the number of user arrivals in the current stage exceeds a corresponding threshold. As shown in Line 8, stage τ_p is updated when: $|\mathcal{T}(p, \tau_p)| \geq 1 + \sqrt{T \cdot \sum_{\tau=1}^{\tau_p - 1} |\mathcal{T}(p, \tau_p)|}$.

Whenever a new stage is triggered for query q or the arrival probability, the flag `Switch` is changed to True and signals a new cache update in Algorithm 4.

Algorithm 4 `Switch Cache`

1: **for** query $q \in \mathcal{Q}$ **do**
2: Calculate empirical means $\hat{p}_t(q) = N_{p,t}(q)/t$ and $\hat{c}_t(q) = L_{c,t}(q)/N_{c,t}(q)$.
3: Calculate LCB of the cost $\underline{c}_t(q) = \hat{c}_t(q) - \sqrt{\frac{\log(4mT^3)}{2N_{c,t}(q)}}$.
4: **end for**
5: Compute $\mathcal{M}_t = \texttt{Reverse\_Greedy}\left(\mathcal{Q}, \hat{\boldsymbol{p}}_t, \boldsymbol{c}_t, d, k\right)$.
6: Call `LLM` to obtain responses $a(q)$ for all $q \in \mathcal{M}_t \setminus \mathcal{M}_{t-1}$.
7: **Return:** $\mathcal{M}_t$.

Active Exploration via Optimism Principle (Algorithm 4). To efficiently explore unknown serving costs, we follow the principle of optimism in the face of uncertainty [34], [35]. Specifically, we compute *lower confidence bounds* (LCBs) as optimistic estimates in Line 3 of Algorithm 4: $\underline{c}_t(q) = \hat{c}_t(q) - \sqrt{\frac{2\log(\frac{4mn}{\delta})}{N_{c,t}(q)}}$. The intuition is that for queries with limited feedback, subtracting the confidence radius from the empirical mean creates an optimistic estimate that encourages the learner to favor these uncertain queries, thus collecting more feedback on their unknown costs.

Once the `Switch` flag is true, we invoke `Switch Cache`, which uses the `Reverse_Greedy` oracle with the estimated probabilities $\hat{\boldsymbol{p}}_t$ and optimistic costs $\boldsymbol{c}_t$ to construct a new cache $\mathcal{M}_t$. The agent then calls the LLM to populate all new queries in $\mathcal{M}_t \setminus \mathcal{M}_{t-1}$. After updating the cache, the agent proceeds to serve queries and refine estimates using `Serve and Update` in the following.

Serving and Updating (Algorithm 5). After determining the cache $\mathcal{M}_t$ for the current round, the agent compares the lower confidence bound (LCB) estimate $\underline{c}_t(q_t)$ with the mismatch cost incurred by using the nearest cached query $s(q_t, \mathcal{M}_t)$ (see Line 2). The `LLM` is queried only when $\underline{c}_t(q_t) < d(q_t, \mathcal{M}_t)$, as shown in Line 3 of Algorithm 5. The use of optimistic LCBs—where $\underline{c}_t(q_t) \leq c(q_t)$ with high probability—promotes exploration by increasing the likelihood of collecting real cost feedback. This mechanism is crucial for enabling the agent to gather informative data in the early rounds and for accelerating the learning of the unknown parameters.

If $\underline{c}_t(q_t)$ is smaller than $d(q_t, \mathcal{M}_t)$, the agent queries the `LLM`, incurs the realized serving cost $C_t(q_t)$, and uses the feedback to update its estimate of the expected cost $c(q_t)$. Otherwise, it returns the cached response $a(s(q_t, \mathcal{M}_t))$ without receiving any feedback. Finally, the agent updates the arrival probability statistics based on the observed query q_t (see Line 9).

C. Theoretical Result and Its Discussion

In this section, we present the main theoretical result characterizing the performance of `CLCB-SC-LS`.

Algorithm 5 `Serve and Update`

$N_{p,t+1}(q) = N_{p,t}(q), N_{c,t+1}(q) = N_{c,t}(q), L_{c,t+1}(q) = L_{c,t}(q)$ for all $q \in \mathcal{Q}$.
if $\underline{c}_t(q_t) < d(q_t, \mathcal{M}_t)$ **then**
    Call $a_t = \texttt{LLM}(q_t)$ with cost $C_t(q_t) \sim c(q_t)$, and serve the user with response a_t.
    Update $N_{c,t+1}(q_t) = N_{c,t}(q_t) + 1$ and $L_{t+1}(q_t) = L_t(q_t) + C_t(q_t)$.
    Update $\mathcal{T}(i, \tau_q) = \mathcal{T}(i, \tau_q) \cup \{t\}$.
else
    Serve the user with $a_t = s(q_t, \mathcal{M}_t)$.
end if
Set $N_{p,t+1}(q_t) = N_{p,t}(q_t) + 1$, $\mathcal{T}(p, \tau_p) = \mathcal{T}(p, \tau_p) \cup \{t\}$.

Theorem 3 (Regret of `CLCB-SC-LS`)**.** *For the online semantic caching problem, the regret of* `CLCB-SC-LS` *is upper bounded as:* $\mathrm{Reg}(T) \le O\left(\sqrt{mT\log(mT)}\log\log T\right)$.

Remark 3 (Discussion of Theorem 3)**.** Theorem 3 shows that the regret of `CLCB-SC-LS` grows sublinearly with the number of rounds T. Notably, the number of cache switches is bounded by $O(\log\log T)$, as established in Lemma 7. In the special case of exact matching (i.e., where semantic distance reduces to exact query identity), our result improves upon the regret bound in Zhu et al. [2] by a factor of $O(k\sqrt{m}/C_1)$, where $C_1 = \min_{q\in\mathcal{Q}} c(q)$ denotes the minimum query cost. Furthermore, our bound matches the state-of-the-art result in Liu et al. [3] up to an $O(\log\log T)$ multiplicative factor, thanks to our stage-based cache update design. In contrast, both of these prior algorithms incur linear $O(T)$ switching costs.

D. Regret Analysis

To prove Theorem 3, we establish a sequence of lemmas. We first show that the empirical estimates of the arrival probabilities and serving costs concentrate around their true values uniformly over time. Compared to the offline setting, this requires an *any-time* guarantee, introducing an additional T term within the logarithmic terms due to union bounding over $t \in [T]$.

Lemma 5 (Any-time concentration of cost and probability estimates)**.** *For any $\delta > 0$, we define the concentration events as follows:* $\mathcal{E}_{arvl} := \left\{\sum_{q\in\mathcal{Q}} |\hat{p}_t(q) - p(q)| \le \sqrt{\frac{2m\log(2T/\delta)}{t}} \text{ for } t \in [T]\right\}$, $\mathcal{E}_{cost} := \left\{|\hat{c}_t(q) - c_t(q)| \le \sqrt{\frac{\log(2mT^2/\delta)}{2N_{c,t}(q)}} \text{ for } q \in \mathcal{Q}, t \in [T]\right\}$. *Then both events hold with high probability:* $\Pr[\mathcal{E}_{arvl}] \ge 1-\delta$ *and* $\Pr[\mathcal{E}_{cost}] \ge 1-\delta$.

To handle the partial feedback inherent in the caching system—where no serving cost is observed when the cache is used—we require a stronger loss function sensitivity result than Lemma 4, specifically tailored for optimistic cost estimates based on lower confidence bounds.

Lemma 6 (Sensitivity of Loss under Optimistic Cost Estimates.)**.** *For any cost parameters $\underline{\boldsymbol{c}}, \boldsymbol{c} \in [0,1]^m$ such that $\underline{c}(q) \le c(q)$ for all $q \in \mathcal{Q}$, the loss function satisfies that for any cache $\mathcal{M} \subseteq \mathcal{Q}$, $\ell(\mathcal{M}; \boldsymbol{c}, \boldsymbol{p}, d) - \ell(\mathcal{M}; \underline{\boldsymbol{c}}, \boldsymbol{p}, d) \le \sum_{q\in\mathcal{Q}:\, \underline{c}(q)\le d(q,\mathcal{M})} p(q)(c(q) - \underline{c}(q))$.*

We also upper bound the number of cache switches for each query $q \in \mathcal{Q}$ and the arrival probability.

Lemma 7 (Number of cache switches)**.** *We can upper bound the number of stages $\tau_q(T)$ and $\tau_p(T)$ by:* $\mathbb{E}\left[\sum_{q\in\mathcal{Q}} \tau_q(T)\right] \le O(m\log\log T), \mathbb{E}[\tau_p(T)] \le O(\log\log T)$.

Proof of Theorem 3. For the ease of exposition, we define auxiliary variables $N'_{c,t}(q), \hat{\boldsymbol{c}}'_t, \underline{\boldsymbol{c}}'_t$ and $N'_{t,p}, \hat{\boldsymbol{p}}'_t$ as follows: If the selected cache at time t is just switched in Algorithm 3 (i.e., `Switch` == True), then $N'_{c,t}(q) = N_{c,t}(q), \hat{\boldsymbol{c}}'_t = \hat{\boldsymbol{c}}_t, \underline{\boldsymbol{c}}'_t = \underline{\boldsymbol{c}}_t$ and $N'_{t,p} = t, \hat{\boldsymbol{p}}'_t = \hat{\boldsymbol{p}}_t$; Otherwise, if the previous cache is maintained (i.e., `Switch` == False), then $N'_{c,t}(q) = N'_{c,t-1}(q), \hat{\boldsymbol{c}}'_t = \hat{\boldsymbol{c}}'_{t-1}$ and $N'_{t,p} = N'_{t-1,p}, \hat{\boldsymbol{p}}'_t = \hat{\boldsymbol{p}}'_{t-1}$.

Under the high-probability events $\mathcal{E}_{\text{arvl}}$ and $\mathcal{E}_{\text{cost}}$, we decompose the regret (without switching cost) into three terms: $\ell(\mathcal{M}_t; \boldsymbol{c}, \boldsymbol{p}) - \alpha\ell(\mathcal{M}^*; \boldsymbol{c}, \boldsymbol{p}) = \underbrace{\ell(\mathcal{M}_t; \boldsymbol{c}, \boldsymbol{p}) - \ell(\mathcal{M}_t; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t)}_{\text{estimation gap}} + \underbrace{\ell(\mathcal{M}_t; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t) - \alpha\ell(\mathcal{M}^*; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t)}_{\text{greedy gap}} + \underbrace{\alpha\ell(\mathcal{M}^*; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t) - \alpha\ell(\mathcal{M}^*; \boldsymbol{c}, \boldsymbol{p})}_{\text{optimistic gap}} \le \ell(\mathcal{M}_t; \boldsymbol{c}, \boldsymbol{p}) - \ell(\mathcal{M}_t; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t) + \alpha\ell(\mathcal{M}^*; \boldsymbol{c}, \hat{\boldsymbol{p}}'_t) - \alpha\ell(\mathcal{M}^*; \boldsymbol{c}, \boldsymbol{p}) \le \sum_{q\in\mathcal{Q}:\, \underline{c}'_t(q)\le d(q,\mathcal{M})} p(q)(c(q) - \underline{c}'_t(q)) + 2\|\hat{\boldsymbol{p}}'_t - \boldsymbol{p}\|_1 \le \sum_{q\in\mathcal{Q}:\, \underline{c}'_t(q)\le d(q,\mathcal{M})} 2p(q)\sqrt{\frac{\log(\frac{2mT^2}{\delta})}{2N'_{c,t}(q)}} + 2\sqrt{\frac{2m\log(\frac{2T}{\delta})}{N'_{t,p}}}$.

where the first equality is due to adding and subtracting $\ell(\mathcal{M}_t; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t, d)$ and $\alpha\ell(\mathcal{M}^*; \underline{\boldsymbol{c}}'_t, \hat{\boldsymbol{p}}'_t, d)$, the first inequality is due to the greedy gap ≤ 0 by the approximation ratio guarantee of the `Reverse Greedy`, $\underline{\boldsymbol{c}}'_t \le \boldsymbol{c}$ by Lemma 3, and $\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c})$ is monotone regarding $\boldsymbol{c}$, the second inequality is due to applying Lemma 6 and Lemma 4, the last inequality is due to Lemma 5.

Now let us denote $\gamma := \log(2mT^2/\delta)$ for the first term, we have: $\mathbb{E}[\sum_{t=1}^T \sum_{q\in\mathcal{Q}:\, \underline{c}'_t(q)\le d(q,\mathcal{M}_t)} p(q)\sqrt{2\gamma/N'_{c,t}(q)}]$
$= \mathbb{E}[\sum_{t=1}^T \sum_{q=\tilde{q}_t} \sqrt{2\gamma/N'_{c,t}(q)}]$
$= \mathbb{E}[\sum_{i\in\mathcal{Q}} \sum_{\tau\in\tau_q(T)} \sum_{t\in\mathcal{T}(q,\tau)} \sqrt{2\gamma/N'_{c,t}(q)}]$
$\le \mathbb{E}[\sum_{i\in\mathcal{Q}} \sum_{\tau\in\tau_q(T)} \sum_{t\in\mathcal{T}(q,\tau)} \sqrt{2\gamma/\sum_{\tau'=1}^{\tau-1} |\mathcal{T}(q,\tau')|}]$
$\le \mathbb{E}[\sum_{i\in\mathcal{Q}} \sum_{\tau\in\tau_q(T)} |\mathcal{T}(q,\tau)|\sqrt{2\gamma/\sum_{\tau'=1}^{\tau-1} |\mathcal{T}(q,\tau')|}]$
$\le \mathbb{E}[\sum_{i\in\mathcal{Q}} \sum_{\tau\in\tau_q(T)} 2\sqrt{T\cdot\sum_{\tau'=1}^{\tau_q-1} |\mathcal{T}(q,\tau')|/m}$
$\cdot\sqrt{2\gamma/\sum_{\tau'=1}^{\tau-1} |\mathcal{T}(q,\tau')|}] = \mathbb{E}[\sum_{q\in\mathcal{Q}} \tau_q(T)]\cdot 2\sqrt{2T\gamma/m}$
$\le O(\sqrt{mT\gamma}\cdot\log\log T)$, where the first equality is due to the fact that any query q is observed if and only if $\underline{c}'_t(q) \le d(q, \mathcal{M}_t)$ and by the triggering probability equivalence trick of [35] in expectation $\sum_{\underline{c}'_t(q)\le d(q,\mathcal{M}_t)} p(q)\cdot x_q$ equals to $\mathbb{E}[\sum_{q=\tilde{q}_t} x_q]$ where $\tilde{q}_t = q_t$ if q_t is the observed or $\tilde{q}_t = \emptyset$ otherwise, the second equality is due by exchanging the order of summation, the first inequlaity is due to $N'_{c,t}(q) \ge \sum_{\tau'=1}^{\tau-1} |\mathcal{T}(q,\tau')|$, the last inequality is due to Line 5 of Algorithm 3.

For the second term, we can similarly derive that $\mathbb{E}\left[\sum_{t=1}^T 2\sqrt{2m\log(2T/\delta)/N'_{t,p}}\right] \le$

$O\left(\sqrt{mT\log(2T/\delta)}\cdot\log\log T\right)$. Finally, we set $\delta = 1/T$ and note that the number of cache switches is bounded by $O((m+1)\log\log T)$ by Lemma 7, yielding a lower-order regret term $\mathbb{E}\left[\sum_{q\in\mathcal{M}_t\setminus\mathcal{M}_{t-1}} c(q)\right] \leq O(m^2\log\log T)$ in Eq. (7), completing the proof. □

VI. Experiments

In this section, we present the experimental results for all three of our settings, i.e., Algorithms 1, 2, and 3.

Experimental Setup. We test these algorithms' performance in a synthetic caching setting where we generate the **queries** ($m = 20$ is used for all experiments except for the ablation study on m, and ChatGPT was used to generate the queries), and there is a differing variety of similarity between queries in terms of their content. Some queries are quite similar to each other, while others are more distantly related. For example "Rome attractions", "top 10 tourist attractions and hidden gems in Rome", "AI papers", "recent breakthroughs in AI and machine learning research" are some of the queries in our dataset. Queries are sampled using the uniform distribution. To construct the **expected serving cost** $c(q)$ associated with each query q, we tokenize these queries and define the cost as the number of tokens associated with each query that is min-max normalized after adding $\epsilon_t(q)$ zero-mean Gaussian noise with a standard deviation of 0.05. To form the **embeddings** $e(q)$ associated with each query, we use a sentence transformer model [36] that maps queries to a 384-dimensional vector representation. The similarity metric d is the Euclidean distance between query embeddings. Similarly, the similarity metric is min-max normalized to ensure that it is in the range of 0-1. The shaded regions in Figure 2 indicate the standard deviations as 10 runs were made with each run having a different random seed, which affects the query sampling for the incoming query stream and the noise associated with the cost of the queries.

Optimality of Reverse-Greedy with Known Parameters. First we explore how Algorithm 1 (Reverse Greedy) approximates the optimal loss $\ell(\mathcal{M}^*; \boldsymbol{p}, \boldsymbol{c}, d)$ which we refer to here as the Brute Force optimal loss. We also plot the loss for the Least Frequently Used (LFU) caching strategy, which prioritizes caching the most frequent queries [2]. We show the variation of the optimal loss with the cache size k in Figure 2(a). It can be seen that Reverse Greedy perfectly approximates the Brute Force optimal loss in this setting.

Offline Setting. Now we test how Algorithm 2 performs in the offline setting as seen in Figure 2(b). For these experiments we keep $m = 20$ and $k = 5$ and vary the length of the incoming query stream. We also consider the LCB variant of Algorithm 2 and Epsilon Greedy and LFU baselines. Epsilon Greedy, a standard benchmark for bandit algorithms, uses Reverse Greedy to construct the cache but uses the empirical cost $\hat{c}$ and randomly removes a query with probability ϵ_g (which we set to be 0.2 for all experiments). CUCB-SC and CLCB-SC perform comparably in terms of the final suboptimality gap while Epsilon Greedy does slightly worse. This aligns with expectations: CUCB is theoretically optimal, while CLCB, though it may optimistically estimate costs, performs well in typical (non-worst-case) scenarios.

Online Setting. Next, we conduct an experiment for the online caching setting where we test the performance of CLCB-SC-LS (Algorithm 3) and the online variant of CUCB-SC (Algorithm 2) and its LCB variant (CLCB-SC, without adding the switching cost in regret) by recording the variation of the average regret with the number of rounds t as shown in Figure 2(c). Finally, we consider LFU as a static baseline in this setting as we let it consider the full query stream before deciding on the cache. It can be seen that CLCB-SC-LS and CLCB-SC have sublinear regret with the average regret approaching zero as round t increases and achieves at least 11.75% improvement compared with the most competitive Epsilon-Greedy baseline.

We now present ablation studies regarding the variation of the final average regret with k and m. Figure 2(d) shows that CLCB-SC and CLCB-SC-LS get lower final regret as the cache size increases while CUCB-SC and LFU get higher final regret, showing our enlarged improvement compared with the strongest baseline (from 11.75% to 54.04%). Figure 2(e) shows the variation of the final average regret with m, notably CLCB-SC-LS and CLCB-SC achieve lower final average regret than the other baselines for as m values, validating our tight regret bound in Theorem 3.

Finally, Figure 2(f) presents the total number of cache switches and running time in the online setting. As expected, the low-switching algorithm CLCB-SC-LS achieves the fewest switches and fastest runtime, reducing switches by up to 90.91% and running time by up to 85.40% compared to the baselines. Despite this constraint, its performance remains strong—Figure 2(c) shows that its regret closely tracks that of CLCB-SC, which incurs nearly twice as many switches.

VII. Conclusion and Future Directions

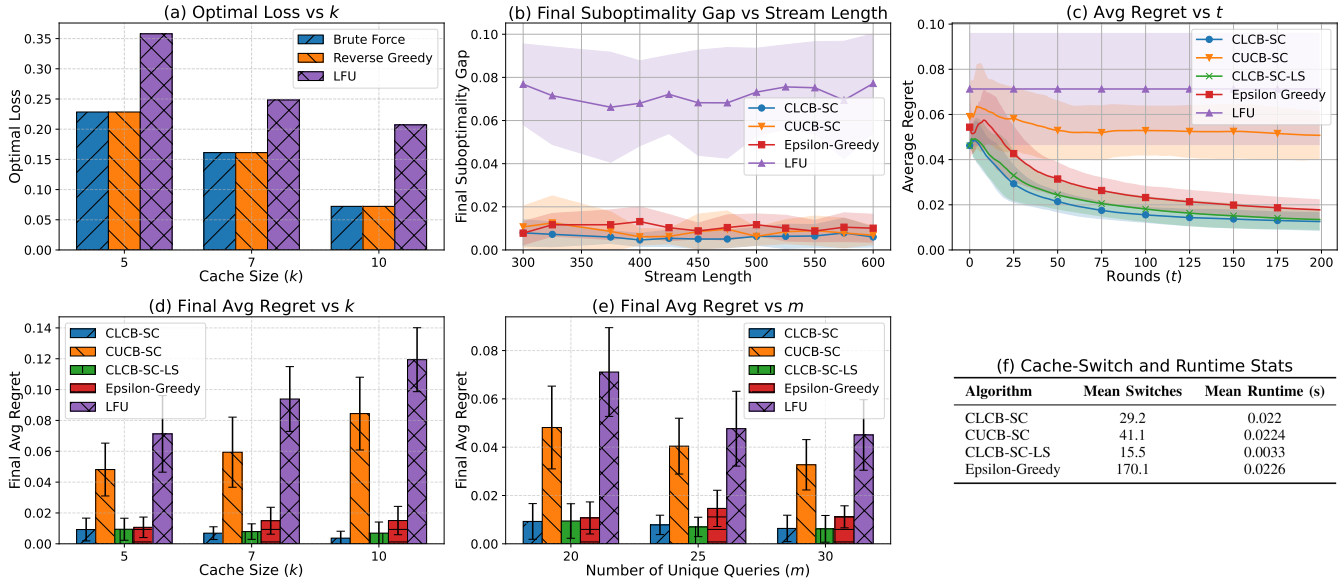
We present a principled, learning-based semantic caching framework for low-cost LLM serving. Our study spans three settings—oracle, offline, and online—offering a systematic study under different information regimes. The proposed algorithms achieve state-of-the-art performance guarantees and show superior empirical results. A promising future direction is to explore hybrid approaches that integrate offline training with online adaptation for more robust and efficient caching.

VIII. Appendix

Here we provide the missing proofs in the main body.

Proof of Lemma 1. For Eq. (5), the optimization problem is equivalent to $\mathcal{M}^* = \operatorname{argmax}_{\mathcal{M}\subseteq\mathcal{U},|\mathcal{M}|\leq k} \sum_{v\in N(\mathcal{M})} p(v)c(v)$, which is essentially the maximum vertex cover problem on bipartite graphs and is NP-hard as proved in [37]. □

Proof of Lemma 2. For each query $q \in \mathcal{Q}$, define the per-query loss as $\ell_q(\mathcal{M}) = \min\{d(q,\mathcal{M}), c(q)\} = \min_{u\in\mathcal{M}\cup\{q_0\}} d(q,u)$, where q_0 is a virtual element with $d(q,q_0) = c(q)$. As submodularity and monotonicity are preserved under non-negative linear combinations. It suffices to show that $\ell_q(\mathcal{M})$ is a *monotone decreasing submodular*



(f) Cache-Switch and Runtime Stats

Algorithm	Mean Switches	Mean Runtime (s)
CLCB-SC	29.2	0.022
CUCB-SC	41.1	0.0224
CLCB-SC-LS	15.5	0.0033
Epsilon-Greedy	170.1	0.0226

Fig. 2: (a) Variation of the loss with the cache size k for a fixed set of queries with $m = 20$ (b) Variation of the final suboptimality gap with the stream length in the offline setting, we test the performance of Algorithm 2 and its offline variant CLCB-SC (c) Variation of the average regret with the number of rounds t for the online setting. CUCB-SC and CLCB-SC are the online variants of Algorithm 2 and its LCB variant (d) Ablation study on the final average regret's change with the cache size k in the online setting (e) Ablation study on the final average regret's change with the number of distinct queries m in the online setting (f) Table showing the mean number of switches and runtime of the algorithms in the online setting

function. Let $A \subseteq B \subseteq \mathcal{Q}$ and $u \notin B$. For **Monotonicity**, since $\ell_q(\mathcal{M})$ takes the minimum over a larger set when moving from A to B, it holds that $\ell_q(B) \leq \ell_q(A)$. For **Submodularity**, we need to show the diminishing returns property: $\ell_q(A \cup \{u\}) - \ell_q(A) \leq \ell_q(B \cup \{u\}) - \ell_q(B)$. We exhaustively consider three cases based on the relative values of $\ell_q(A)$, $\ell_q(B)$, and $d(q, u)$; in all, the inequality holds. □

Proof of Theorem 1. The approximation guarantee follows from invoking Corollary 4 of [31], which provides bounds for minimizing a non-increasing supermodular function under a cardinality constraint using the reverse greedy algorithm. □

Proof of Lemma 3. For $\mathcal{E}_{arvl}$, $\Pr[\mathcal{E}_{arvl}] \geq 1-\delta$ holds because of Weissman et al. [38]. For $\mathcal{E}_{cost}$ and $\mathcal{E}_{arvl}$, we follow Lemma 5 and Lemma 6 of Liu et al. [3], respectively. □

Proof of Lemma 4. For any cache $\mathcal{M}$, any $\boldsymbol{p}, \boldsymbol{p}', \boldsymbol{c}, \boldsymbol{c}' \in [0,1]^m$, we can rearrange $\bar{p} = (\max\{p(q), p'(q)\})_{q\in\mathcal{Q}}$, $\underline{p} = (\min\{p(q), p'(q)\})_{q\in\mathcal{Q}}$, $\bar{c} = (\max\{c(q), c'(q)\})_{q\in\mathcal{Q}}$, $\underline{c} = (\min\{c(q), c'(q)\})_{q\in\mathcal{Q}}$. Let us fix any d and rewrite $\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}) := \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}, d)$ for simplicity, then we have: $|\ell(\mathcal{M}; \boldsymbol{p}', \boldsymbol{c}') - \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}')| \leq \ell(\mathcal{M}; \bar{\boldsymbol{p}}, \boldsymbol{c}') - \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}') = \sum_{q\in\mathcal{Q}} (\bar{p}(q) - \underline{p}(q)) \min\{c'(q), d(q, \mathcal{M})\} \leq \sum_{q\in\mathcal{Q}} (\bar{p}(q) - \underline{p}(q)) = \sum_{q\in\mathcal{Q}} |p(q) - p'(q)|$. We can also prove: $|\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}') - \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c})| \leq \ell(\mathcal{M}; \boldsymbol{p}, \bar{\boldsymbol{c}}) - \ell(\mathcal{M}; \boldsymbol{p}, \underline{\boldsymbol{c}}) = \sum_{q\in A} p(q)(\bar{c}(q) - \underline{c}(q)) + \sum_{q\in B} p(q)(d(q, \mathcal{M}) - \underline{c}(q)) + \sum_{q\in C} p(q)(d(q, \mathcal{M}) - d(q, \mathcal{M})) \leq \sum_{q\in A\cup B} p(q)(\bar{c}(q) - \underline{c}(q)) = \sum_{\underline{c}(q)\leq d(q,\mathcal{M})} p(q)(\bar{c}(q) - \underline{c}(q)) \leq \sum_{q\in\mathcal{Q}} p(q)(\bar{c}(q) - \underline{c}(q)) = \sum_{q\in\mathcal{Q}} p(q)|c(q) - c'(q)|$, where $A = \{q \in \mathcal{Q} : d(q, \mathcal{M}) \geq \bar{c}(q) > \underline{c}(q)\}$, $B = \{q \in \mathcal{Q} : \bar{c}(q) \geq d(q, \mathcal{M}) > \underline{c}(q)\}$, and $C = \{q \in \mathcal{Q} : \bar{c}(q) \geq \underline{c}(q) > d(q, \mathcal{M})\}$. Finally, using the fact that $|\ell(\mathcal{M}; \boldsymbol{p}', \boldsymbol{c}') - \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c})| \leq |\ell(\mathcal{M}; \boldsymbol{p}', \boldsymbol{c}') - \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}')| + |\ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c}') - \ell(\mathcal{M}; \boldsymbol{p}, \boldsymbol{c})|$ concludes the Lemma 4. □

Proof of Lemma 5. We follow the same proof of Lemma 3, but use the union bound over all $t \in [T]$ and $N_{c,t}(q) \in [T]$. □

Proof of Lemma 6. We follow the proof of Lemma 4 with $\bar{\boldsymbol{c}} = \boldsymbol{c}, \underline{\boldsymbol{c}} = \boldsymbol{c}'$. □

Proof of Lemma 7. Let us denote $s_0 = \log\log(T/m)$. By the switch condition in Line 5 of Algorithm 4 and Lemma 2 in Dong et al. [39], we have: $|\mathcal{T}(q, \tau_q)| \geq (T/m)^{1-2^{-\tau_q+1}} \geq (T/m)^{1-2^{\log\log(T/m)}} = \frac{T}{e\cdot m}$ for any $\tau_q \geq s_0 + 1$ and τ_q is not the last stage. Since the total number of observations is at most T (one query each round), there are at most $m(e+1)$ (including their last stages for $q \in \mathcal{Q}$) query-stage pairs (q, τ) s.t. for $q \in \mathcal{Q}$, $\tau_q \in [\tau_q(T)]$ s.t. $\tau_q \geq s_0 + 1$ (which implies $\tau_q(T) \geq s_0 + 1$). Thus, we have $\sum_{q\in\mathcal{Q}} \mathbb{I}\{\tau_q(T) \geq s_0 + 1\}\tau_q(T) = \sum_{q:\tau_q(T)\geq s_0+1} \sum_{\tau_q\in[\tau_q(T)]} \mathbb{I}\{\tau_q < s_0 + 1\} + \mathbb{I}\{\tau_q \geq s_0 + 1\} < m(s_0 + 1) + m(e + 1)$. Therefore, we have $\mathbb{E}[\sum_{q\in\mathcal{Q}} \tau_q(T)] = \mathbb{E}[\sum_{q\in\mathcal{Q}} \mathbb{I}\{\tau_q(T) < s_0 + 1\}\tau_q(T) + \mathbb{I}\{\tau_q(T) \geq s_0 + 1\}\tau_q(T)] \leq m(s_0 + 1) + [m(s_0 + 1) + m(e + 1)] = m(2s_0 + e + 3) = O(m \log\log(T))$. Similarly, we can follow the same proof to show that $\mathbb{E}[\tau_p] \leq O(\log\log(T))$ by treating the arrival probability as a single arm with $m = 1$ that has total T observations. □

REFERENCES

[1] S. Samsi, D. Zhao, J. McDonald, B. Li, A. Michaleas, M. Jones, W. Bergeron, J. Kepner, D. Tiwari, and V. Gadepally, "From words to watts: Benchmarking the energy costs of large language model inference," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2023, pp. 1–9.

[2] B. Zhu, Y. Sheng, L. Zheng, C. Barrett, M. Jordan, and J. Jiao, "Towards optimal caching and model selection for large model inference," vol. 36, pp. 59 062–59 094, 2023.

[3] X. Liu, X. Dai, J. Zuo, S. Wang, C. Joe-Wong, J. Lui, and W. Chen, "Offline learning for combinatorial multi-armed bandits," *arXiv preprint arXiv:2501.19300*, 2025.

[4] W. Gill, M. Elidrisi, P. Kalapatapu, A. Ahmed, A. Anwar, and M. A. Gulzar, "Meancache: User-centric semantic cache for large language model based web services," *arXiv preprint arXiv:2403.02694*, 2024.

[5] F. Bang, "Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings," *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265607979

[6] I. Stogiannidis, S. Vassos, P. Malakasiotis, and I. Androutsopoulos, "Cache me if you can: An online cost-aware teacher-student framework to reduce the calls to large language models," *arXiv preprint arXiv:2310.13395*, 2023.

[7] J. Li, C. Xu, F. Wang, I. M. von Riedemann, C. Zhang, and J. Liu, "Scalm: Towards semantic caching for automated chat services with large language models," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[8] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in neural information processing systems*, vol. 35, pp. 30 318–30 332, 2022.

[9] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International conference on machine learning*. PMLR, 2023, pp. 38 087–38 099.

[10] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 274–19 286.

[11] B. Spector and C. Re, "Accelerating llm inference with staged speculative decoding," *arXiv preprint arXiv:2308.04623*, 2023.

[12] X. Miao, C. Shi, J. Duan, X. Xi, D. Lin, B. Cui, and Z. Jia, "Spotserve: Serving generative large language models on preemptible instances," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2024, pp. 1112–1127.

[13] Y. Fu, L. Xue, Y. Huang, A.-O. Brabete, D. Ustiugov, Y. Patel, and L. Mai, "{ServerlessLLM}:{Low-Latency} serverless inference for large language models," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024, pp. 135–153.

[14] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, A. Levskaya, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," *ArXiv*, vol. abs/2211.05102, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253420623

[15] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261697361

[16] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, D. Y. Fu, Z. Xie, B. Chen, C. W. Barrett, J. Gonzalez, P. Liang, C. Ré, I. Stoica, and C. Zhang, "High-throughput generative inference of large language models with a single gpu," in *International Conference on Machine Learning*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257495837

[17] I. Gim, G. Chen, S. seob Lee, N. Sarda, A. Khandelwal, and L. Zhong, "Prompt cache: Modular attention reuse for low-latency inference," *ArXiv*, vol. abs/2311.04934, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265067391

[18] G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, "Mobile edge intelligence for large language models: A contemporary survey," *ArXiv*, vol. abs/2407.18921, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:271534421

[19] X. Dai, J. Li, X. Liu, A. Yu, and J. C. S. Lui, "Cost-effective online multi-llm selection with versatile reward models," *ArXiv*, vol. abs/2405.16587, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270063595

[20] T. Feng, Y. Shen, and J. You, "Graphrouter: A graph-based router for llm selections," *ArXiv*, vol. abs/2410.03834, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:273185502

[21] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *International Conference on Machine Learning*. PMLR, 2013, pp. 151–159.

[22] Q. Wang and W. Chen, "Improving regret bounds for combinatorial semi-bandits with probabilistically triggered arms and its applications," in *Advances in Neural Information Processing Systems*, 2017, pp. 1161–1171.

[23] X. Liu, J. Zuo, S. Wang, C. Joe-Wong, J. Lui, and W. Chen, "Batch-size independent regret bounds for combinatorial semi-bandits with probabilistically triggered arms or independent arms," in *Advances in Neural Information Processing Systems*, 2022.

[24] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE transactions on Computers*, vol. 50, no. 12, pp. 1352–1361, 2001.

[25] H. Bahn, "Web cache management based on the expected cost of web objects," *Information and Software Technology*, vol. 47, no. 9, pp. 609–621, 2005.

[26] T. S. Salem, G. Özcan, I. Nikolaou, E. Terzi, and S. Ioannidis, "Online submodular maximization via online convex optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, 2024, pp. 15 038–15 046.

[27] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii, "Nearest-neighbor caching for content-match applications," in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 441–450.

[28] F. Chierichetti, R. Kumar, and S. Vassilvitskii, "Similarity caching," in *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2009, pp. 127–136.

[29] M. Garetto, E. Leonardi, and G. Neglia, "Similarity caching: Theory and algorithms," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 526–535.

[30] J. J. Pan, J. Wang, and G. Li, "Survey of vector database management systems," *The VLDB Journal*, vol. 33, no. 5, pp. 1591–1615, 2024.

[31] V. P. Il'ev, "An approximation guarantee of the greedy descent algorithm for minimizing a supermodular set function," *Discrete Applied Mathematics*, vol. 114, no. 1-3, pp. 131–146, 2001.

[32] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, "Provably efficient reinforcement learning with linear function approximation," in *Conference on Learning Theory*. PMLR, 2020, pp. 2137–2143.

[33] X. Dai, X. Liu, J. Zuo, H. Xie, C. Joe-Wong, and J. C. S. Lui, "Variance-aware bandit framework for dynamic probabilistic maximum coverage problem with triggered or self-reliant arms," *IEEE Transactions on Networking*, vol. 33, no. 3, pp. 982–993, 2025.

[34] X. Dai, X. Sun, J. Zuo, X. Liu, and J. C. Lui, "A unified online-offline framework for co-branding campaign recommendations," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '25, 2025, p. 427–438.

[35] X. Liu, J. Zuo, S. Wang, J. C. Lui, M. Hajiesmaili, A. Wierman, and W. Chen, "Contextual combinatorial bandits with probabilistically triggered arms," in *International Conference on Machine Learning*. PMLR, 2023, pp. 22 559–22 593.

[36] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[37] N. Apollonio and B. Simeone, "The maximum vertex coverage problem on bipartite graphs," *Discrete Appl. Math.*, vol. 165, p. 37–48, Mar. 2014. [Online]. Available: https://doi.org/10.1016/j.dam.2013.05.015

[38] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger, "Inequalities for the l1 deviation of the empirical distribution," *Hewlett-Packard Labs, Tech. Rep*, 2003.

[39] K. Dong, Y. Li, Q. Zhang, and Y. Zhou, "Multinomial logit bandit with low switching cost," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2607–2615.