ContextCache: Context-Aware Semantic Cache for Multi-Turn Queries in Large Language Models

Jianxin Yan
Zhejiang University
Hangzhou, China
yanjianx666@gmail.com

Wangze Ni*
Zhejiang University
Hangzhou, China
niwangze@zju.edu.cn

Lei Chen
HKUST (GZ) & HKUST
Guangzhou, China
leichen@cse.ust.hk

Xuemin Lin
Shanghai Jiaotong University
Shanghai, China
xuemin.lin@gmail.com

Peng Cheng
Tongji University
Shanghai, China
cspcheng@tongji.edu.cn

Zhan Qin, Kui Ren
Zhejiang University
Hangzhou, China
{qinzhan,kuiren}@zju.edu.cn

ABSTRACT

Semantic caching significantly reduces computational costs and improves efficiency by storing and reusing large language model (LLM) responses. However, existing systems rely primarily on matching individual queries, lacking awareness of multi-turn dialogue contexts, which leads to incorrect cache hits when similar queries appear in different conversational settings. This demonstration introduces ContextCache, a context-aware semantic caching system for multi-turn dialogues. ContextCache employs a two-stage retrieval architecture that first executes vector-based retrieval on the current query to identify potential matches and then integrates current and historical dialogue representations through self-attention mechanisms for precise contextual matching. Evaluation of real-world conversations shows that ContextCache improves precision and recall compared to existing methods. Additionally, cached responses exhibit approximately 10 times lower latency than direct LLM invocation, enabling significant computational cost reductions for LLM conversational applications.

PVLDB Reference Format:
Jianxin Yan, Wangze Ni, Lei Chen, Xuemin Lin, Peng Cheng, and Zhan Qin, Kui Ren. ContextCache: Context-Aware Semantic Cache for Multi-Turn Queries in Large Language Models. PVLDB, 18(12): 5391 - 5394, 2025.
doi:10.14778/3750601.3750679

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

1 INTRODUCTION

Large Language Models (LLMs) like ChatGPT have become essential components in conversational systems and productivity tools due to advanced language understanding and generation capabilities. However, LLM inference demands substantial computational resources, leading to significant operational costs. To address this challenge, semantic caching has emerged as an optimization strategy [2], adapting database result caching principles to LLM workflows through vector-based indexing and similarity-based retrieval of previously generated responses.

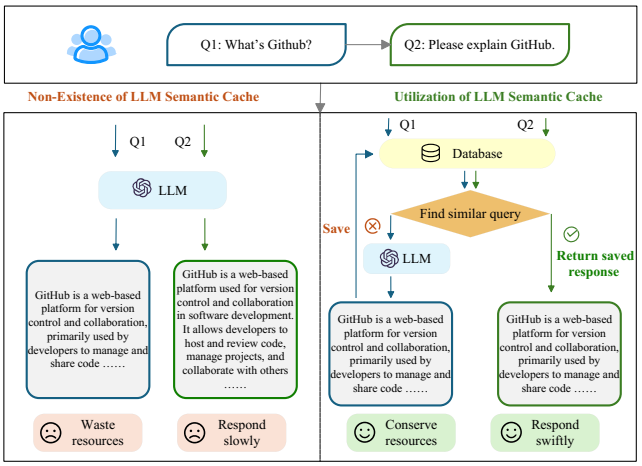


Figure 1: Optimizing LLM Responses with Semantic Caching

Figure 1 illustrates LLM semantic caching, which enables the retrieval of previously computed responses for semantically similar queries. The figure contrasts two processing approaches: without caching (left), each query requires a complete LLM inference cycle, consuming significant computational resources and increasing latency; and with caching (right), when a user submits query Q2 that is semantically similar to a previously cached query Q1, the system directly retrieves the existing response without invoking the LLM. This cached approach **reduces both response time and operational costs by eliminating redundant processing**. Previous empirical studies support its practical efficiency: approximately 33% of search engine queries are resubmitted [5], and 31% of ChatGPT interactions contain semantically similar queries [3].

Unlike traditional caching scenarios, **LLM interactions require consideration of dynamic conversational context rather than isolated requests**. Direct application of conventional database caching paradigms that match queries solely against stored embeddings achieves poor performance in LLM scenarios (as shown in Figure 3a, where GPTCache [2] exemplifies the conventional database

*Wangze Ni is also with The State Key Laboratory of Blockchain and Data Security; Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750679

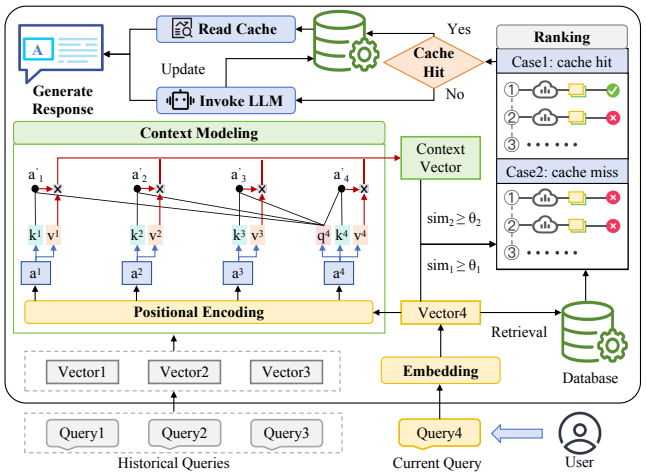
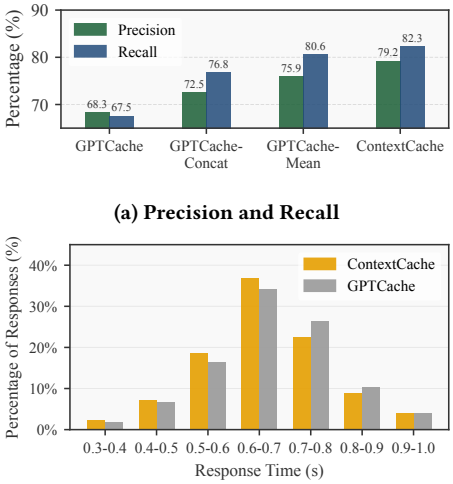


Figure 2: The Architecture Of ContextCache



(a) Precision and Recall

(b) Cache Hit Response Time

Figure 3: Performance Metrics: ContextCache vs. GPTCache

caching paradigm). Existing approaches [3] attempt to concatenate previous queries before applying lightweight pre-trained models or average turn embeddings to extract contextual representations, but face two limitations: Concatenation-based methods encounter attention dilution when self-attention mechanisms process long text sequences [7, 8], while embedding averaging causes representation flattening that obscures turn-specific semantic features [6, 8]. Both approaches exhibit insufficient semantic discrimination due to limited sentence-level supervision, resulting in precision degradation as conversation length increases. Therefore, the core challenge for conversational caching systems is **to develop mechanisms that model context while maintaining precise semantic relationship detection with minimal computational overhead**.

To address these challenges, we present ContextCache, a context-aware semantic caching framework for multi-turn conversational systems. Our approach overcomes existing limitations through three key innovations. First, our Dynamic Context Modeling implements a **hierarchical self-attention mechanism** that captures cross-turn semantic dependencies, mitigating attention dilution in concatenation while preserving turn-specific features lost through representation flattening. Second, our LLM-enhanced training employs **difficult negative sample mining** to improve matching precision, addressing insufficient semantic discrimination through robust sentence-level supervision. Finally, our **Two-Stage Dynamic Retrieval Architecture** combines efficient vector retrieval for candidate selection with precise attention-based contextual matching, balancing computational efficiency with semantic accuracy. Experimental evaluation shows ContextCache outperforms GPTCache, **improving precision and recall by 10.9% and 14.8%** while cache-served responses deliver approximately **10 times lower latency** compared to LLM invocation. Our contributions include:

- A Two-Stage Dynamic Retrieval Architecture combining vector retrieval with attention-based contextual matching.
- A prototype demonstrating our techniques in realistic conversational scenarios, reducing computational costs and latency.

2 BACKGROUND

Semantic caching improves upon traditional methods by operating on a query's meaning rather than its exact words. This allows the system to identify semantically equivalent questions, even when phrased differently, and reuse answers for greater efficiency.

Embedding generation. The core of the system is the embedding generator, which transforms queries into vector representations. It employs transformer-based architectures [8] to encode semantic information: $E(q) = f_\theta(q)$, where geometric proximity in the vector space corresponds to semantic similarity [4].

Similarity quantification. This step provides the decision mechanism for the cache. It uses cosine similarity to measure the angular proximity between two embedding vectors, formulated as: $\text{similarity}(E_1, E_2) = \frac{E_1 \cdot E_2}{||E_1|| \, ||E_2||}$. When this similarity value exceeds a predetermined threshold, the system classifies queries as equivalent and retrieves cached response instead of invoking the LLM.

3 SYSTEM OVERVIEW

3.1 Workflow

ContextCache optimizes large language model applications by leveraging conversational context for semantic caching. Figure 2 illustrates the system architecture with six key components: (1) **Query interception**: User queries are intercepted before transmission to the LLM. (2) **Context collection**: The system captures the current query and retrieves historical dialogue to construct a comprehensive conversational context. (3) **Semantic representation generation**: The system generates embedding vectors only for the current query while reusing historical dialogue embeddings from previous calculations. (4) **Two-tier retrieval**: The system first employs the current query's embedding for preliminary similarity search to identify potential matches. For filtered candidates, it then integrates embeddings from both the current query and historical dialogue, applying self-attention mechanisms to analyze inter-turn relationships and generate a unified contextual representation for precise matching. (5) **Response determination**: When a cache entry with matching semantics and context is found, the system returns the cached response; otherwise, it forwards the query to the LLM. (6) **Cache update**: For LLM-generated responses, the

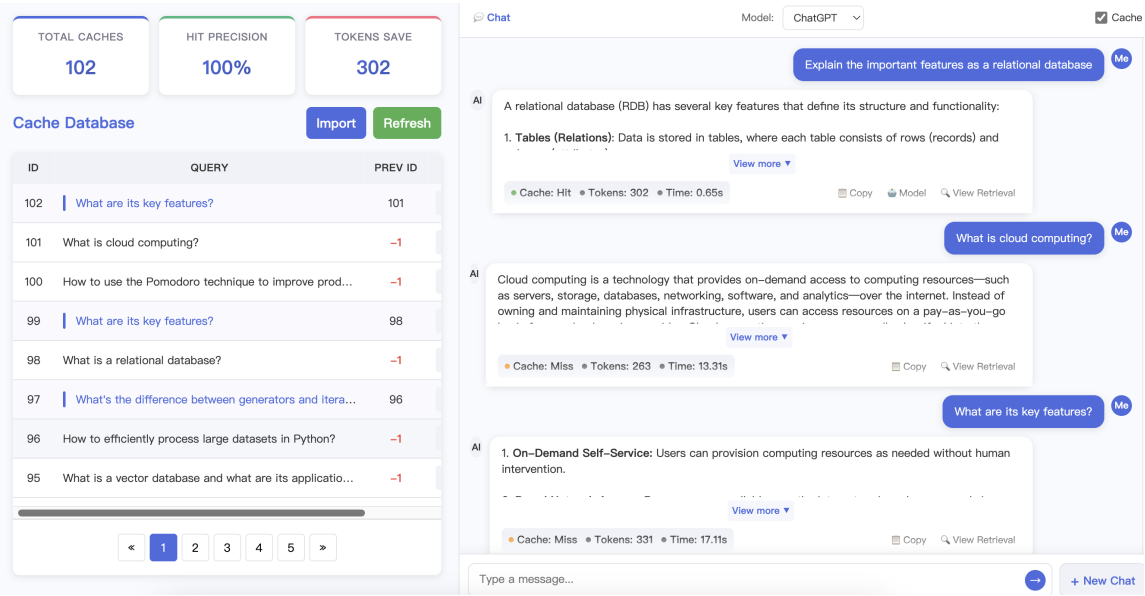


Figure 4: ContextCache User Interface And Demonstration Scenarios

system stores the response along with the query embedding and contextual representation before returning it to the user.

3.2 Implementation

We enhanced GPTCache [2] through an architecture that integrates conversational context:

- **Query preprocessing module** standardizes queries and incorporates conversation context. Historical dialogue records $H = \{h_1, h_2, ..., h_n\}$ establish the conversational context. The Albert [4] embedding model E generates semantic representation only for the current query $v_Q = E(Q)$, while historical dialogue vectors $V_H = \{E(h_1), E(h_2), ..., E(h_n)\}$ are reused from previous calculations, eliminating redundant computation.
- **Two-stage retrieval mechanism** improves cache hit accuracy through progressive refinement. The coarse-grained stage filters candidates $C_1 = \{c \mid \cos(v_Q, v_c) > \theta_1\}$ using cosine similarity threshold θ_1, where v_c represents the query vector of a cached entry. The fine-grained stage uses precomputed contextual representations stored in the cache. For each candidate, the global representation g_c is retrieved from previous cache updates. The system generates the current conversation's global representation $g_{current} = \text{SelfAttention}(V_{current})$ where $V_{current} = \{v_Q\} \cup V_H$. This self-attention mechanism analyzes inter-turn relationships, producing a unified contextual representation. Final similarity assessment uses cosine similarity: $S_c = \cos(g_{current}, g_c)$. The system selects the optimal match $C_{best} = \arg\max_{c \in C_1} S_c$ that exceeds the threshold θ_2, ensuring contextually appropriate responses while maintaining efficiency.
- **Cache update module** employs a dual-storage architecture. New responses R and associated metadata are stored as contextual tuples (Q, R) in a relational database, while query vectors v_Q and global representations $g_{current}$ are indexed in the vector database for efficient similarity search. This separation optimizes both storage and retrieval performance.

3.3 Core Technique And Evaluation

Our evaluation used 1,000 queries derived from the ShareGPT dataset [1]. We initialized the cache with 30% of original dialogue samples [3, 5], then generated the test queries by creating semantically equivalent variations through GPT-4-based paraphrasing to introduce linguistic diversity while preserving semantic intent. This methodology simulates real-world scenarios where users reformulate queries using different phrasings without altering the underlying conversational context.

Context integration. Our self-attention mechanism models semantic relationships across dialogue turns, generating unified contextual representations that capture inter-turn dependencies. Figure 3a demonstrates that our approach significantly outperforms concatenation and averaging methods by preserving turn-specific semantic features while modeling their contextual relationships. Compared to GPTCache, this context-aware approach **achieves 10.9% higher precision and 14.8% improved recall**, particularly in distinguishing between semantically similar queries that appear in different conversational contexts.

Efficiency optimization. Our system reduces computational overhead through the two-stage retrieval mechanism and dual-storage architecture. Compared to direct LLM invocation, this design delivers cache-served responses with approximately **10 times lower latency**, substantially reducing operational costs (Figure 5). Furthermore, as Figure 3b demonstrates, despite incorporating contextual processing that improves semantic precision, we achieve a **3% reduction in average cache hit time** compared to GPTCache.

4 DEMONSTRATION OVERVIEW

We present ContextCache, a system featuring a Vue-based interface that extends LLM query caching through contextual awareness. Similar to GPTCache [2], our implementation uses Albert [4] as the semantic encoder and employs a dual-database architecture combining SQL relational storage with FAISS vector indexing for

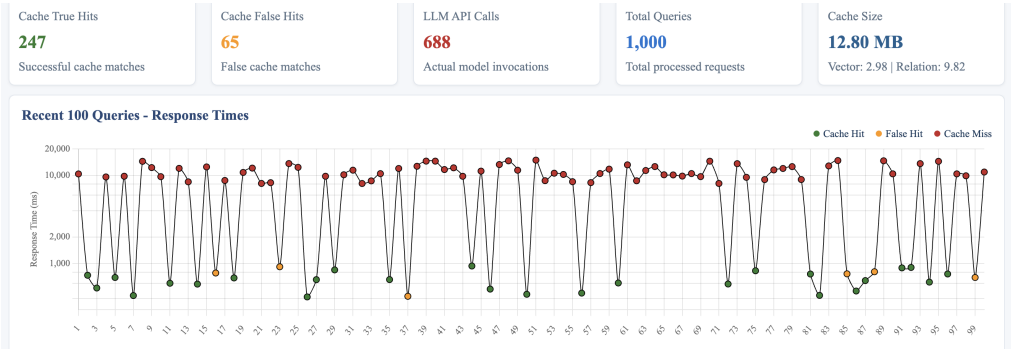


Figure 5: Demonstration Engagement Results

retrieval. Our demonstration showcases key scenarios that illustrate the system's capabilities and practical benefits, as shown in Figure 4.

Scenario 1: Cache Hit. *Our first demonstration evaluates the system's ability to identify similar queries despite linguistic variations.*

The system features a split-screen layout with database contents and interaction panel. We populate it with a pre-curated collection of 100 query-response pairs containing "What is a relational database?" and "What are its key features?". For testing, users submit "Explain the important features as a relational database"—semantically equivalent but lexically distinct from the cached query.

① When the test query is submitted, the system retrieves the cached response, confirmed by a green "Cache Hit" indicator.

② Performance metrics display showing response time (0.65 seconds) and computational savings (302 tokens).

By discerning the user's true intent within a conversational context, ContextCache matches semantically equivalent queries to reduce response latency to milliseconds without compromising the appropriateness of the answer.

Scenario 2: Cache Miss. *Our second demonstration addresses a key challenge: correctly handling similar queries that require different responses based on their conversation history.*

To demonstrate this capability, users initiate a conversation about cloud computing as a controlled test case. After entering "What is cloud computing?", they follow with "What are its key features?"—a query identical to the previously cached query about relational databases but requiring a context-appropriate response.

① When this query is submitted, the system recognizes the contextual difference and displays a "Cache Miss" indicator.

② Following response generation, the database panel updates to display the newly stored conversation with its preserved context.

To avoid semantic confusion and reduce false positives, ContextCache incorporates conversational history into its matching process. This allows it to successfully distinguish between similar queries that appear in different contextual settings.

Demonstration engagement. To quantify system effectiveness, we integrated an interactive dashboard (Figure 5) that visualizes ContextCache's operational metrics in real-time. The interface tracks key performance indicators, including Cache True/False Hits, LLM API Calls, Query Volume, and Memory Utilization. The execution log provides request-level performance data, enabling users to examine query processing paths and response timing. Through this log analysis, users can observe that cache-served responses deliver approximately **10 times lower latency** compared to LLM invocation, while maintaining high semantic matching precision in multi-turn dialogues.

5 CONCLUSION

Our work introduces ContextCache, a context-aware semantic caching system that reduces LLM inference costs, contributing to the wider adoption of LLM applications.

6 ACKNOWLEDGMENT

This research is supported in part by the National Key Research and Development Program of China (Grant No.2023YFF0725100, 2021YFB3100300, 2023YFB2904000); National Natural Science Foundation of China (NSFC Grant No.U22B2060, U2241211, 62441238, 62072395, U20A20178,U23A20306,62032021); the Hong Kong RGC (GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G); the Guangdong-Hong Kong Technology Innovation Joint Funding Scheme (Project No. 2024A0505040012); Key Areas Special Project of Guangdong Provincial Universities (No. 2024ZDZX1006); Guangdong Province Science and Technology Plan Project (No. 2023A0505030011); Guangzhou municipality big data intelligence key lab (No. 2023A03J0012); Zhujiang project (No. 2021JC02X170); Hong Kong ITC ITF grants (No. MHX/078/21, PRP/004/22FX); the Fundamental Research Funds for the Central Universities; Microsoft Research Asia Collaborative Research Grant; HKUST-Webank joint research lab; and the 2023 HKUST Shenzhen-Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund from Shui On Xintiandi and the InnoSpace GBA.

REFERENCES

[1] 2024. ShareGPT. https://sharegpt.com/ Accessed: 2025-03-20.
[2] Fu Bang. 2023. Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. 212–218.
[3] Waris Gill, Mohamed Elidrisi, et al. 2024. Privacy-Aware Semantic Cache for Large Language Models. *arXiv preprint arXiv:2403.02694* (2024).
[4] Zhenzhong Lan, Mingda Chen, et al. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).
[5] Evangelos P. Markatos. 2001. On caching search engine query results. *Computer Communications* 24, 2 (2001), 137–143.
[6] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
[7] Yi Tay, Mostafa Dehghani, Samira Abnar, et al. 2020. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006* (2020).
[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).