Evaluating Semantic Caching in Practice: A Study on a LLM-Driven Distributed Application in a Brazilian EdTech

Henrique Lopes Nóbrega[1], **David Candeia Medeiros Maia**[2], **João Brunet**[3]

[1]Alura – São Paulo – SP – Brazil

[2]Coordenação da Área de Informática – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campina Grande – PB – Brazil

[3]Departamento de Sistemas e Computação – Universidade Federal de Campina Grande Campina Grande – PB – Brazil

`henrique.nobrega@alura.com.br, david.maia@ifpb.edu.br`

`joao.arthur@computacao.ufcg.edu.br`

Abstract. *Large Language Models (LLMs) support various business functions, such as Alura's use of GPT-4 to assess students' answers. However, high computational and financial costs, along with response time issues, limit scalability. While caching mechanisms offer an alternative, traditional cache fails to evaluate queries semantically, leading to low hit rates. This work evaluates semantic caching on an Alura's dataset of* $94,913$ *answers from* $20,639$ *students. Results show that* 45.1% *of LLM requests could be served from the cache, significantly reducing costs and improving response times by 4–12*$\times$ *for cache hits.*

1. Introduction

Large Language Models (LLMs), such as Mixtral [Jiang et al. 2024], Claude [Ânthropic 2024], Llama [Touvron et al. 2023], and GPT [Achiam et al. 2023], have advanced natural language processing tasks like content generation, summarization, and translation. Alura, one of Brazil's largest EdTech companies, faces a scalability challenge with open-ended questions — which are vital for learning as they let students express their understanding — since each unique phrasing requires individual assessment.

Alura's system uses GPT-4's reasoning to evaluate student answers and provide feedback. Over time, the company noticed that many answers were semantically similar to previously evaluated ones, differing only in structure and vocabulary. For instance, in the analyzed dataset, two answers to the same question - 'a callback function' and 'callback' - suggest that caching GPT-4 evaluations could reduce operational costs.

Caching is a well-established strategy for reducing resource consumption [Markatos 2001]. However, traditional caching relies on keyword matching, which overlooks semantic relationships between similar texts, lowering cache hit rates. Given that LLMs inherently handle semantics, semantic caching is a natural solution.

This paper evaluates semantic caching in a real-world industry application, leveraging LLMs to optimize resource use. Analyzing $94,913$ answers from $20,639$ students, our solution reduced LLM requests and costs by 45.1%, while improving average response times by 12X compared to the current implementation. Additionally, a manual inspection of 256 random answer pairs confirmed 87.5% as semantically similar.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 describes our solution; Section 4 presents the experimental results; and Section 5 summarizes our findings and discusses possibilities of future work.

2. Related Work

To mitigate the high computational, operational, and latency costs of using LLMs, various strategies have been proposed. One approach employs models of different sizes to generate answers using smaller, less expensive models [Ramírez et al. 2023, Chen et al. 2023]. However, these approaches were unsuitable for our context: [Ramírez et al. 2023] focuses on classification tasks and lacks semantic query equivalence, while the models in [Chen et al. 2023] were insufficiently powerful for our needs. Additionally, our method does not rely on pre-labeled datasets.

Another approach involves using caching strategies to reduce LLM usage costs. Zhu et al. [Zhu et al. 2023] investigated a combined approach of caching and model multiplexing. However, they assumed an oracle to group similar queries and their evaluation focused on cache management strategies.

Other studies evaluated semantic caching solutions [Bang 2023, Gill et al. 2024, Regmi and Pun 2024], employing embedding generators, storage, and cosine similarity techniques, solutions also used in our work. They used synthetic datasets of web page content [Bang 2023, Gill et al. 2024] or question-answer pairs [Regmi and Pun 2024] and assessed storage size [Gill et al. 2024], cache hits/misses [Bang 2023, Gill et al. 2024, Regmi and Pun 2024], API call reduction [Regmi and Pun 2024], and latency/response time [Bang 2023, Gill et al. 2024, Regmi and Pun 2024]. In contrast, we evaluated semantic caching on a real educational dataset from Alura, focusing on latency, cache hits, and cost savings.

3. Methods

3.1. Applied Solution

Based on the solutions presented in [Bang 2023, Gill et al. 2024], Figure 1 illustrates our developed solution. An embedding generator extracts semantic information from student's answers and the resulting vector is used to search for similar answers that have been previously evaluated by GPT-4. In the event of a cache hit, the feedback is immediately provided to the student. If a cache miss occurs, the student's answer is submitted to GPT-4 - specifically the *gpt-4-1106-preview* [OpenAI 2023] - for evaluation and correction. The new evaluation is then stored in the vector store for future reference.

The embedding generator creates a vector representation of a student's answer using OpenAI *text-3-small model* [OpenAI 2024]. This model was chosen based on its cost, the team's experience with the solution, and its established use in both state-of-the-art and practice. The vector store serves as the repository for all generated vector embeddings. It uses Postgres with PgVector[1] for its simplicity, and HNSW indexes [Malkov and Yashunin 2018] to enable fast searches across embeddings. For each embedding, the vector store also retains associated metadata, such as the original answer and the corresponding correction (obtained from GPT-4).

[1]`https://github.com/pgvector/pgvector`

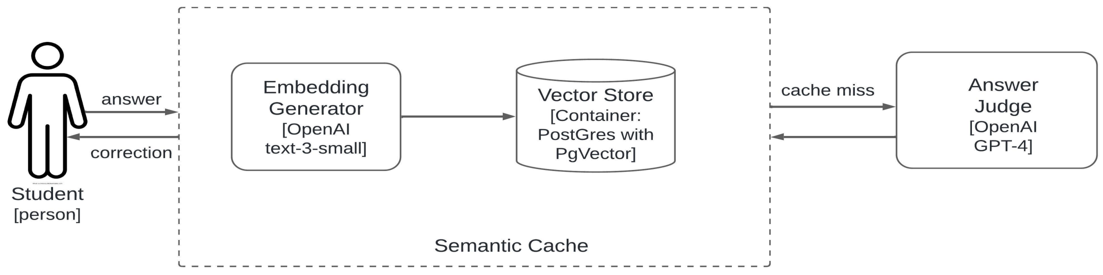

Figure 1. System workflow with a semantic cache

As proposed by [Bang 2023, Gill et al. 2024] we also use cosine similarity [Rahutomo et al. 2012] to measure the similarity between vectors. If the similarity of a cached answer meets or exceeds a predefined threshold, we consider it a cache hit; otherwise, the GPT-4 model is invoked to provide a fresh correction.

3.2. Experiments

To explore semantic caching in our scenario, we formulated the following research questions: i) How many LLM API calls (answer judge) can be avoided with semantic caching?; ii) What cost savings result from reduced LLM usage?; iii) How does semantic caching impact response time for student corrections?

To answer these questions, we conducted experiments using a production dataset obtained from Alura. The dataset consists of $94,913$ answers from $20,639$ anonymized students, covering 33 distinct questions proposed by the platform. These questions cover programming topics such as programming logic, Java, object orientation, JavaScript/CSS, GitHub, database manipulation, Scrum, and UI design. Of these questions, 12 require students to analyze a given code snippet and either explain its output or correct an error, while the remaining questions ask students to explain a concept or command based on the provided context. The dataset represents a subset of students' interactions with the system from December 4, 2023 to February 2, 2024. Each entry includes the question asked, the student's answer, and the correction provided by the GPT-4 model.

We considered three scenarios in the experiments: one without any caching mechanism (actual Alura's system); one with traditional caching; one with semantic caching. For the first scenario, 50 random dataset entries were submitted to the GPT-4 model, establishing a baseline response time for comparison. In the second scenario, we implemented traditional caching using a locally hosted Redis[2] instance - an open-source caching solution known for its popularity, simplicity, and low-latency performance. Finally, for the third scenario, we developed a prototype of the semantic caching system in Javascript (according to Figure 1). For each student answer submitted to Alura's system, the caching evaluation is performed using the *text-3-small* embedding generator, a vector store with PgVector and cosine similarity (as described in Section 3.1).

For each dataset entry experimented, we collected metrics on the operation's response time, whether a cache hit occurred, and details of the most similar cached response

[2]`https://redis.io/`

along with its similarity score. Native JavaScript resources were used to measure the response times of the caching system operations.

4. Results and Discussion

In our experiments, we first tested similarity thresholds of 0.7 [Bang 2023] and 0.83 [Gill et al. 2024], but a manual inspection revealed cache hits even when more specific feedback was needed. We then tested thresholds of 0.95, 0.98 and 0.99, with hit rates of 53.76%, 45.1% and 39.88%, respectively. A second manual inspection of 256 response pairs confirmed 87.5% of pairs above 0.98 were truly equivalent. Thus, we chose 0.98 as the optimal threshold, balancing cache hit rate, cost reduction and feedback quality.

In our analysis, we assume that each query to the GPT-4 model for generating feedback costs $\$0.01$. Given these configurations, Table 1 summarizes the observed response times, cache hit rates, and cost estimates.

Table 1. Observed values in experiments of each scenario

Scenario	**Cache hit ($\%$)**	95% **CI for Average Response Time (ms)**	**Cost Estimate**
Without Cache	-	$[6,029.08, 7,625.22]$	$\$949.13$
Traditional Caching	30.38%	$[0.0559, 0.0571]$	$\$660.79$
Semantic Caching	45.1%	$[531.56, 535.01]$	$\$521.07$

Regarding response times, users experienced average values of 6.827 seconds without caching, while caching reduced these times to at least half a second. For traditional caching, the 99th percentile was 0.121ms, with 283 queries exceeding 1ms and a maximum observed time of 11.10ms. In contrast, semantic caching had a 99th percentile of $1,635.1$ms, with most times below 800ms.

Considering the average response time in the scenario without caching, and the 99th percentile in the scenario with semantic caching, the response time decreased by up to 4.17X (without accounting for additional network latencies) and up to 3.93X (when adding additional 100ms of network latency). These improvements can be even greater when comparing average values, with a decrease of times up to 12X.

Comparing the results of traditional and semantic caching, we observe a marked improvement in efficiency with the semantic approach. With a carefully chosen threshold of 0.98, the semantic approach outperformed the traditional model in terms of cache hits. The semantic cache saved up to 45.1% of queries, compared to 30.38% with traditional Redis caching. This substantial increase in cache hits highlights the effectiveness of semantic caching in understanding and leveraging similarities that extend beyond mere textual matches. Although a reduction of $\$139.72$ in the cost of using GPT-4 may seem modest in the tested scenarios, it's important to remember that, in the massive daily use of the GPT-4 model in Alura's system, a 14.72% decrease in queries (from traditional to semantic caching) submitted to LLM can contribute to significant cost savings.

However, it's important to note that while semantic caching significantly reduces the number of LLM invocations compared to traditional caching, it comes at the cost of higher response times. The response times for semantic cache operations are noticeably longer, with observed times reaching several seconds at higher percentiles.

When comparing the $99th$ percentile of both approaches, traditional caching reduces response times by up to $13,513$X. When comparing average response times, the reduction reaches $9,428$X. Additionally, when calculating a weighted average for each cache strategy - considering the proportion of queries served and not served by the cache in each strategy - traditional cache achieved a response time of $3,988.54$ms, while semantic cache achieved a time of $4,752.97$ms. In this case, the reduction factor is up to 1.19X, making the overall experienced response times of both approaches more comparable.

Despite the increased response times, the trade-off is beneficial. The semantic cache's ability to reduce the need for invoking the costly GPT-4 model — especially given the massive daily use in the context of Alura — makes it a preferable solution. This trade-off highlights the strategic advantage of accepting higher cache operational processing times to significantly decrease the more substantial times and costs associated with LLM operations, thereby enhancing overall system efficiency and user experience.

5. Conclusions and Future Works

In this paper, we evaluated the use of a semantic caching solution within the context of Alura's educational platform. To assess its effectiveness, we conducted experiments on a real dataset from Alura comparing three scenarios: the company's existing solution without caching, a traditional caching system, and the semantic caching solution.

Using a semantic similarity threshold of 0.98, we observed a 45.1% reduction in LLM queries - significantly exceeding the 30.38% reduction achieved by traditional caching in our dataset. Given the cost associated with each LLM query, this reduction translates into substantial cost savings. Moreover, while the semantic cache reduces overall response times by a factor between 3.93 and 12 compared to the non-caching system, its operations exhibit higher response times than those of a traditional cache.

This work has limitations that suggest opportunities for future work. While cosine similarity is used in the literature [Bang 2023, Gill et al. 2024, Regmi and Pun 2024], exploring alternative similarity metrics - such as Manhattan distance, Euclidean distance, Jaccard similarity or Minkowski distance - could further enhance caching efficiency and cost savings. We used a fixed 0.98 threshold to balance cache hit and accuracy. However, exploring a broader range of values and implementing an adaptive threshold mechanism based on real-time performance metrics are promising directions. Additionally, evaluating different embedding algorithms and testing semantic caching on larger Alura datasets, as well as datasets from other online learning platforms, could provide valuable insights.

Acknowledgments

This research was partially sponsored by IBM and Flextronics as a result of a collaboration between the companies and Universidade Federal de Campina Grande.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bang, F. (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)*, pages 212–218.

Chen, L., Zaharia, M., and Zou, J. (2023). Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

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

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024). Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Markatos, E. P. (2001). On caching search engine query results. *Computer Communications*, 24(2):137–143.

OpenAI (2023). New models and developer products announced at devday. `https://openai.com/blog/new-models-and-developer-products-announced-at-devday`. Accessed in September 05, 2024.

OpenAI (2024). New embedding models and api updates. `https://openai.com/blog/new-embedding-models-and-api-updates`. Accessed in September 05, 2024.

Rahutomo, F., Kitasuka, T., Aritsugi, M., et al. (2012). Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1. University of Seoul South Korea.

Ramírez, G., Lindemann, M., Birch, A., and Titov, I. (2023). Cache & distil: Optimising api calls to large language models. *arXiv preprint arXiv:2310.13561*.

Regmi, S. and Pun, C. P. (2024). Gpt semantic cache: Reducing llm costs and latency via semantic embedding caching. *arXiv preprint arXiv:2411.05276*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Zhu, B., Sheng, Y., Zheng, L., Barrett, C., Jordan, M. I., and Jiao, J. (2023). On optimal caching and model multiplexing for large model inference. *arXiv preprint arXiv:2306.02003*.

Ânthropic (2024). Introducing the next generation of claude. `https://www.anthropic.com/news/claude-3-family`. Âccessed in September 05, 2024.