UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HENRIQUE LOPES NÓBREGA

# USO DE CACHE SEMÂNTICO PARA ECONOMIZAR RECURSOS EM FUNCIONALIDADES PROVIDAS POR LLMS

CAMPINA GRANDE - PB

2024

HENRIQUE LOPES NÓBREGA

# USO DE CACHE SEMÂNTICO PARA ECONOMIZAR RECURSOS EM FUNCIONALIDADES PROVIDAS POR LLMS

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.**

**Orientador : João Arthur Brunet Monteiro**

**CAMPINA GRANDE - PB**

**2024**

HENRIQUE LOPES NÓBREGA

# USO DE CACHE SEMÂNTICO PARA ECONOMIZAR RECURSOS EM FUNCIONALIDADES PROVIDAS POR LLMS

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.**

**Trabalho aprovado em: 15 de Maio de 2024.**

**CAMPINA GRANDE - PB**

## RESUMO

Modelos de Linguagem de Grande Escala (LLMs), como o ChatGPT, Claude e Llama 2, revolucionaram o processamento de linguagem natural, criando novos casos de uso para aplicações que utilizam esses modelos em seus fluxos de trabalho. No entanto, os altos custos computacionais desses modelos acarretam problemas de custo e latência, impedindo a escalabilidade de funcionalidades baseadas em LLM para muitos serviços e produtos, especialmente quando dependem de modelos com melhores capacidades de raciocínio, como o GPT-4 ou o Claude 3 Opus. Além disso, muitas consultas a esses modelos são duplicadas. O cache tradicional é uma solução natural para esse problema, mas sua incapacidade de determinar se duas consultas são semanticamente equivalentes leva a baixas taxas de cache hit.

Neste trabalho, propomos explorar o uso de cache semântico, que considera o significado das consultas em vez de sua formulação exata, para melhorar a eficiência de aplicações baseadas em LLM. Realizamos um experimento usando um conjunto de dados real da Alura, uma empresa brasileira de educação, em um cenário onde um aluno responde a uma pergunta e o GPT-4 corrige a resposta. Os resultados mostraram que 45,1% das solicitações feitas ao LLM poderiam ter sido atendidas a partir do cache usando um limiar de similaridade de 0.98, com uma melhoria de 4-10 vezes na latência. Esses resultados demonstram o potencial do cache semântico para melhorar a eficiência de funcionalidades baseadas em LLM, reduzindo custos e latência enquanto mantém os benefícios de modelos avançados de linguagem como o GPT-4. Essa abordagem poderia possibilitar a escalabilidade de funcionalidades baseadas em LLM para uma gama mais ampla de aplicações, avançando na adoção desses modelos poderosos em diversos domínios.

# Using semantic cache to spare resources of LLM-powered features

## ABSTRACT

Large Language Models (LLMs) such as ChatGPT, Claude and Llama 2 have revolutionized natural language processing, creating many new use cases for applications that use these models in their workflows. However, the high computational costs of these models lead to issues with cost and latency, preventing the scalability of LLM-based features to many services and products especially when they depend on models with better reasoning capabilities, such as GPT-4 or Claude 3 Opus. Additionally, many queries to these models are duplicated. Traditional caching is a natural solution to this problem, but its inability to determine if two queries are semantically equivalent leads to low cache hit rates.

In this work, we propose exploring the use of semantic caching, which considers the meaning of queries rather than their exact wording, to improve the efficiency of LLM-based applications. We conducted an experiment using a real dataset from Alura, a Brazilian EdTech company, in a scenario where a student answers a question and GPT-4 corrects the answer. The results showed that 45.1% of the requests made to the LLM could have been served from the cache using a similarity threshold of 0.98, with a 4-10x improvement in latency. These results demonstrate the potential of semantic caching to improve the efficiency of LLM-based features, reducing costs and latency while maintaining the benefits of advanced language models like GPT-4. This approach could enable the scalability of LLM-based features to a wider range of applications, advancing the adoption of these powerful models in various domains.

# Using semantic cache to spare resources of LLM-powered features

Henrique Lopes Nóbrega
Universidade Federal de Campina Grande
Campina Grande - PB
henrique.nobrega@ccc.ufcg.edu.br

João Arthur Brunet Monteiro
Universidade Federal de Campina Grande
Campina Grande - PB
joao.arthur@computacao.ufcg.edu.br

## ABSTRACT

Large Language Models (LLMs) such as ChatGPT, Claude and Llama 2 have revolutionized natural language processing, creating many new use cases for applications that use these models in their workflows. However, the high computational costs of these models lead to issues with cost and latency, preventing the scalability of LLM-based features to many services and products especially when they depend on models with better reasoning capabilities, such as GPT-4 or Claude 3 Opus. Additionally, many queries to these models are duplicated. Traditional caching is a natural solution to this problem, but its inability to determine if two queries are semantically equivalent leads to low cache hit rates.

In this work, we propose exploring the use of semantic caching, which considers the meaning of queries rather than their exact wording, to improve the efficiency of LLM-based applications. We conducted an experiment using a real dataset from Alura, a Brazilian EdTech company, in a scenario where a student answers a question and GPT-4 corrects the answer. The results showed that 45.1% of the requests made to the LLM could have been served from the cache using a similarity threshold of 0.98, with a 4-10x improvement in latency. These results demonstrate the potential of semantic caching to improve the efficiency of LLM-based features, reducing costs and latency while maintaining the benefits of advanced language models like GPT-4. This approach could enable the scalability of LLM-based features to a wider range of applications, advancing the adoption of these powerful models in various domains.

## Keywords

Semantic Caching, Large Language Models

## 1. INTRODUCTION

Large Language Models (LLMs) such as Mixtral [10], Claude [8], Llama [24] and ChatGPT [1] represent the forefront of advancements in natural language processing, especially in language understanding and generation. These models have been pivotal in transforming a wide array of applications across diverse sectors, enhancing applications from education [11,9] to health care [23, 4, 21].

**Motivation**. Despite their transformative potential, the use of LLMs is charged with challenges primarily due to the substantial computational resources they require. GPT-3, for example, has 175 billion parameters [2], requiring 326 GB of memory (considering that every parameter is a float16) to perform inference, which makes operational costs expensive. Advances have been made in making inference more efficient, such as employing Mixture of Experts (MoE) architectures that dynamically allocate computation across different 'expert' models based on the input [26] and improvements in the quantization of model parameters to reduce memory usage without compromising accuracy [12, 25], yet the use of these LLMs is still restricted due to their problems with cost and latency, especially at scale.

Alura, a Brazilian edtech company, faces a significant scale problem dealing with open-ended questions. These questions are a critical component of the learning process, allowing for a demonstration of a student's understanding in their own words. However, each student's unique phrasing of their answers to the same question necessitates individual assessment. Alura's current system, relying on GPT-4's advanced reasoning capabilities, demands high computational resources, as each distinct answer is evaluated for correctness, leading to considerable costs and latency. Over time, Alura observed that many answers are semantically similar to previously evaluated ones, suggesting a significant opportunity to reduce operational demands through caching.

**Problem.** Using cache to reduce resource consumption is a well-studied strategy [14]. However, traditional caching mechanisms predominantly rely on keyword matching, which often fails to capture the semantic relationships between similar, resulting in a low hit-rate. For example, similar queries such as "The answer is 4" and "answer is four" would typically result in a cache miss in traditional caches, leading to an unnecessary usage of the LLM even if they could be answered by the same LLM inference.

**Key insights and contributions.** In this paper, we propose the use of semantic cache to reduce the resources usage of LLM-powered features by leveraging the semantic understanding capabilities of LLMs to encode queries into vectors of floating numbers. By comparing these semantic vectors, our system can effectively recognize semantically similar queries, enhancing cache hit rates. Our collaboration with Alura demonstrated that implementing a semantic cache could prevent up to 45.1% of their LLM usage, reducing both cost and latency in the end-user.

Our research delineates the development and implementation of this semantic caching system, evaluates its performance against traditional caching mechanisms, and discusses its broader implications for the use of LLMs in various use cases. This approach not only mitigates the noted challenges but also enhances the scalability and efficiency of LLM applications, potentially transforming their economic and operational viability.

The remainder of this paper is organized as follows: Section 2 reviews related work, discussing various techniques for conserving resources in LLM-powered features and distinguishing our approach from existing methodologies. Section 3 describes our applied solution to the specific challenges faced by Alura. In Section 4, we detail the experimental setup used to test the efficacy of our semantic caching system. Section 5 presents the results of these experiments, analyzing the impact of our solution in terms of resource efficiency and response latency. Section 6 outlines potential future research directions to further enhance and expand the scope of semantic caching. Finally, we conclude the paper in Section 7 with a summary of our findings and their implications for the broader use of LLMs in educational and other sectors.

## 2. RELATED WORK

Ramírez et al. [19] and Stogiannidis et al. [22] saved LLM's resources by training smaller models on the output of powerful models using a teacher-student architecture. These small models progressively handle a larger share of requests, thus conserving computational resources. However, these studies focus primarily on classification tasks and do not address the semantic equivalences of queries as a mechanism to save resources.

Chen et al. [3] introduces the LLM Cascade technique, where queries pass through a sequence of LLMs (from smaller to larger) until a satisfactory answer is achieved. The verdict of whether an answer is satisfactory is made by a classification model trained on a labeled dataset. Despite its potential, our approach use case required us to directly use a powerful model like GPT-4 from the beginning, limiting our explorations of this method. Moreover, our approach does not require pre-labeled datasets, presenting a simpler yet effective solution to reduce usage resources from LLMs in real-world scenarios.

Zhu et al. [27] discusses a combined approach of caching and model multiplexing as a path to reduce costs on LLM usages by always picking the cheapest model that can answer the query (if the query was not previously cached). However, it assumes semantic caching as a solved problem, not exploring its complexities. In contrast, our work directly tackles the challenges of semantic caching, demonstrating its feasibility and effectiveness through an experiment with a real-world dataset.

Fu [6] introduces an open-source semantic cache framework, CacheGPT, that helps to use semantic cache in production-ready features. Similarly, Gill et al. [7] demonstrates how semantic cache can be used to reduce usages in a scenario privacy-oriented through the use of federated learning. Our work carves out a distinct niche within this landscape by applying semantic caching specifically within an educational setting, demonstrating that semantic caching can not only significantly reduce latency and operational costs but also improve the scalability and efficiency of educational applications powered by LLMs.

## 3. APPLIED SOLUTION

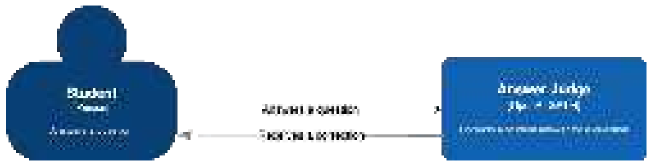

**Figure 1. The traditional workflow at Alura where a student submits an answer and receives a correction, with the Answer Judge (GPT-4) being the sole component for evaluation.**

We developed a semantic caching system to address the inefficiencies of traditional caching methods inside LLM-powered applications. This system, implemented in collaboration with Alura, takes advantage of the deep semantic understanding of LLMs to enhance caching efficiency, thus optimizing computational resources and reducing response times.

Figure 1 illustrates Alura's initial system, which primarily employs the Answer Judge component, utilizing GPT-4 - in particular the *gpt-4-1106-preview* [16] - to evaluate and correct student responses. While this method effectively assesses the accuracy and relevance of answers, it demands significant computational resources and introduces considerable latency, as each student's unique response necessitates individual processing.

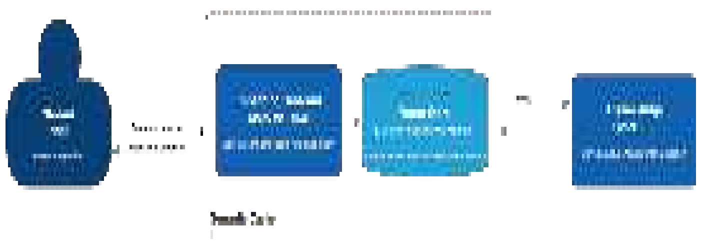

**Figure 2. Enhanced workflow with semantic caching usage.**

Our proposed enhancement, portrayed in Figure 2, introduces a semantic caching solution to optimize resource utilization and reduce response times. The solution contains the following components:

- **Embedding Generator.** This component generates a vector embedding of a student's answer using OpenAI *text-3-small model* [15]. The 1536-dimensional vector that it creates represents the semantic content of the answer.
- **Vector Store.** A storage for all the vector embeddings. It utilizes Postgres with PgVector [17] for its simplicity and HNSW indexes [13] to ensure fast searches within all embeddings. For each embedding, our vector store also keeps associated metadata, such as the original answer that originated the embedding and the correction for that answer. We employ cosine similarity [18] to calculate the similarity between vectors. This similarity goes from -1 to 1, with 1 indicating the most similarity possible between two vectors.

When it receives a new answer, our system proceeds as follows:

- The system generates an embedding for the incoming answer.
- This embedding is then matched against the Vector Store to locate the most similar existing answer to the same question.
- If a cached answer's similarity score meets or exceeds a predefined threshold, we consider it a cache hit. The Answer Judge's correction for the cached answer is then relayed back to the student.
- Should the similarity score fall short of the threshold, the Answer Judge is solicited to provide a fresh correction.
- Every answer, alongside its correction — whether retrieved from cache or newly generated — is recorded in the Vector Store for potential future use.

## 4. EXPERIMENT

### 4.1 Research Questions

In our experiment, we aimed to answer the following questions:

- **Reduction in LLM usage**: How many API calls to the LLM (Answer Judge) could be avoided by implementing our semantic caching system?

- **Latency improvement:** How does the introduction of a semantic cache impact the latency of corrections for students?

## 4.2 Experiment With Semantic Caching

To address the research questions posed, we implemented our applied solution as a prototype using JavaScript. We conducted a detailed evaluation of our semantic caching system using a production dataset consisting of 94,913 anonymized responses to 33 distinct questions. Each dataset entry included the question asked, the student's answer, and the correction provided by the Answer Judge. To simulate the operational conditions as accurately as possible, we processed the data sequentially to ensure the most realistic assessment. For each data point, we collected metrics on the operation's latency, whether a cache hit occurred, and the details of the most similar cached response along with its similarity score. For the semantic cache, we chose a similarity threshold of 0.98, indicating that if a response reached this level of similarity with a prior answer, it was considered a cache hit.

This threshold was selected to strike a balance between increasing the cache hit rate while ensuring the accuracy of the corrections applied from the cached responses. To validate this threshold, we manually analyzed 256 pairs of responses and their most similar matches, confirming that 87.5% of pairs with a similarity above this threshold were semantically equivalent.

To assess the baseline performance of traditional caching, we utilized a Redis instance hosted locally on our machine. In this experiment, we replayed the dataset sequentially once more, recording only the latency of each operation, the occurrence of cache hits, and the details of the cached answer if a hit occurred.

To find out the inherent latency of the LLM without the influence of any caching mechanism, we conducted an additional experiment. We randomly selected 50 entries from our dataset and processed each one without prior corrections, requiring a new invocation of the Answer Judge for each response. We chose to evaluate 50 entries as a sample size to balance a good representation of our system behavior with costs consideration, as each invocation of GPT-4 incurs financial costs. We established a baseline latency which serves as a benchmark against which the performance improvements provided by our semantic caching system can be measured.

## 5. RESULTS

## 5.1 Traditional Caching

Using the traditional caching as our baseline metrics, we observed a significant reduction in the number of necessary LLM invocations. Specifically, 30.38% of the requests could be avoided. This outcome indicates that even basic caching can reduce the load on computational resources considerably on LLM-powered features, depending on the specific use case.

To show the impact of the traditional caching in the feature, we plot the average latency to each percentile as follows:

| 90th percentile | 95th percentile | 99th percentile |
|---|---|---|
| 0.067 ms | 0.082 ms | 0.121 ms |

**Table 1. Latency distribution per percentile in the traditional cache setting.**

These results demonstrate that traditional caching can significantly reduce response times, with most cached requests being served in under a tenth of a millisecond. It yields a big impact because the average latency of our Answer Judge is 6.82 seconds, so using traditional caching can decrease our latency up to 6820x. Even if we consider a pessimistic scenario where our average latency to perform a cache search is 100 ms, it still marks an improvement of 68 times in the latency for the end-user.

## 5.2 Semantic Caching

Implementing our semantic caching system and varying the similarity threshold provided further insights into the balance between cache hit rate and the accuracy of the responses. The following table summarizes the percentage of cache hit at different thresholds:

| Threshold | Cache Hit (%) |
|---|---|
| 0.99 | 39.88% |
| 0.98 | 45.1% |
| 0.95 | 53.76% |

**Table 2. Cache hit percentage by threshold in the semantic cache system.**

As the threshold for similarity increases, the cache hit decreases because we increase the strictness of the semantic matching criteria. However, even if we use a high threshold, like 0.99, it still yields better results than the traditional cache. For example, a threshold at 0.99 has a cache hit of 39.88%, while the traditional cache has a cache hit of 30.38%.

Concerning latency, the recorded latencies at various percentiles represent the typical response times occurring with each cache access, irrespective of a hit or miss:

| 90th percentile | 95th percentile | 99th percentile |
|---|---|---|
| 672.11 ms | 784.55 ms | 1635.1 ms |

**Table 3. Latency distribution per percentile in the semantic cache system.**

Even while semantic cache latency is greater than the traditional cache latency, it still shows improvement from 4x to 10x in comparison with a system without cache at all.

Comparing the baseline results achieved with traditional caching to those obtained with semantic caching highlights a marked improvement in efficiency using our semantic approach. With a carefully chosen threshold of 0.98, the semantic caching system significantly outperformed the traditional model, demonstrating a higher percentage of cache hit. Specifically, the semantic cache saved up to 45.10% of requests, compared to 30.38% with traditional Redis caching. This substantial increase in cache hit highlights the effectiveness of semantic caching in understanding and leveraging similarities that extend beyond mere textual matches.

However, it is important to note that while semantic caching offers considerable advantages in terms of reducing the number of LLM invocations, it does come with higher latencies. The typical response times for semantic cache operations are considerably longer, with latencies reaching up to several seconds at higher percentiles. Despite these increased latencies, the trade-off is beneficial. The semantic cache's ability to drastically reduce the necessity for invoking the costly Answer Judge makes it a preferable solution. This trade-off highlights the strategic advantage of accepting higher cache operational latencies to significantly decrease the more substantial latencies and costs associated with LLM operations, enhancing overall system efficiency and user experience.

## 6. FUTURE WORKS

While our experiments employed a fixed threshold of 0.98 to balance cache hit rate and accuracy, future research could explore the impact of varying this threshold more dynamically. Testing a range of thresholds could provide deeper insights into the trade-offs between increasing cache hit rates and maintaining the accuracy of cached responses. This exploration might involve implementing an adaptive threshold mechanism that could adjust based on real-time feedback from the system's performance metrics, potentially leading to optimized resource usage tailored to the specific characteristics of the workload and user demands.

Another route for future work could involve experimenting with different embedding algorithms. Our current system uses the OpenAI text-3-small model to generate vector embeddings of text responses. However, different models may offer improvements in terms of the richness of the embeddings, the speed of computation, or even the granularity of semantic understanding. Future studies could evaluate the performance of alternative models such as closed models like Cohere Embed [5] and open-source ones like Sentence-BERT [20]. Comparing these models in the context of semantic caching could yield valuable insights into which models provide the best balance between performance and computational overhead.

## 7. CONCLUSION

In this paper, we demonstrated the implementation and effectiveness of semantic caching within Alura's educational platform, significantly reducing the frequency of LLM usage. Our approach achieved up to a 45.10% reduction in LLM usage with a semantic threshold of 0.98, markedly surpassing the 30.38% reduction offered by traditional caching methods.

This substantial decrease directly contributes to reduce costs and improved response times, enhancing the scalability and practicality of employing LLMs in educational settings. The reduction in the need for frequent LLM interactions not only mitigates the computational burden but also facilitates a more efficient and cost-effective usage of advanced LLMs. Although the approach introduces increased latency, the trade-off is validated by the significant decrease in LLM usage and the resultant improvement in overall system performance.

This study highlights the potential of semantic caching to significantly reduce reliance on intensive LLM usage, presenting a viable approach for enhancing the efficiency of LLM-powered applications. The outcomes of this research emphasize the practical benefits of semantic caching, demonstrating its ability to transform the operational dynamics of LLM applications across various domains.

## 8. REFERENCES

[1] Achiam, Josh, et al. "Gpt-4 technical report." arXiv preprint arXiv:2303.08774 (2023). https://arxiv.org/abs/2303.08774

[2] Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901. https://arxiv.org/abs/2005.14165

[3] Chen, Lingjiao, Matei Zaharia, and James Zou. "Frugalgpt: How to use large language models while reducing cost and improving performance." arXiv preprint arXiv:2305.05176 (2023). https://arxiv.org/abs/2305.05176

[4] Clusmann, Jan, et al. "The future landscape of large language models in medicine." Communications medicine 3.1 (2023): 141.

[5] Embed. Retrieved from: https://cohere.com/embed

[6] 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), pages 212–218, Singapore. Association for Computational Linguistics. https://aclanthology.org/2023.nlposs-1.24/

[7] Gill, Waris, et al. "Privacy-Aware Semantic Cache for Large Language Models." arXiv preprint arXiv:2403.02694 (2024). https://arxiv.org/abs/2403.02694

[8] Introducing the next generation of Claude. Retrieved from: https://www.anthropic.com/news/claude-3-family.

[9] Jeon, J., Lee, S. Large language models in education: A focus on the complementary relationship between human teachers and ChatGPT. Educ Inf Technol 28, 15873–15892 (2023). https://doi.org/10.1007/s10639-023-11834-1

[10] Jiang, Albert Q., et al. "Mixtral of experts." arXiv preprint arXiv:2401.04088 (2024). https://arxiv.org/abs/2401.04088

[11] Kasneci, Enkelejda, et al. "ChatGPT for good? On opportunities and challenges of large language models for education." Learning and individual differences 103 (2023): 102274.

[12] Ma, Shuming, et al. "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits." arXiv preprint arXiv:2402.17764 (2024). https://arxiv.org/abs/2402.17764

[13] Malkov, Yu A., and Dmitry A. Yashunin. "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs." IEEE transactions on pattern analysis and machine intelligence 42.4 (2018): 824-836.

[14] Markatos, Evangelos P. "On caching search engine query results." Computer Communications 24.2 (2001): 137-143.

[15] New embedding models and API updates. Retrieved from: https://openai.com/blog/new-embedding-models-and-api-updates

[16] New models and developer products announced at DevDay. Retrieved from: https://openai.com/blog/new-models-and-developer-products-announced-at-devday

[17] PgVector. Retrieved from: https://github.com/pgvector/pgvector

[18] Rahutomo, Faisal, Teruaki Kitasuka, and Masayoshi Aritsugi. "Semantic cosine similarity." The 7th international student conference on advanced science and technology ICAST. Vol. 4. No. 1. South Korea: University of Seoul, 2012.

[19] Ramírez, Guillem, et al. "Cache & distil: Optimising API calls to large language models." _arXiv preprint arXiv:2310.13561_ (2023). https://arxiv.org/abs/2310.13561

[20] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

[21] Sallam, Malik. "The utility of ChatGPT as an example of large language models in healthcare education, research and

practice: Systematic review on the future perspectives and potential limitations." MedRxiv (2023): 2023-02.

[22] Stogiannidis, Ilias, et al. "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). https://arxiv.org/abs/2310.13395

[23] Thirunavukarasu, Arun James, et al. "Large language models in medicine." Nature medicine 29.8 (2023): 1930-1940.

[24] Touvron, Hugo, et al. "Llama 2: Open foundation and fine-tuned chat models." arXiv preprint arXiv:2307.09288 (2023). https://arxiv.org/abs/2307.09288

[25] Wang, Hongyu, et al. "Bitnet: Scaling 1-bit transformers for large language models." arXiv preprint arXiv:2310.11453 (2023). https://arxiv.org/abs/2310.11453

[26] Zhou, Yanqi, et al. "Mixture-of-experts with expert choice routing." Advances in Neural Information Processing Systems 35 (2022): 7103-7114.

[27] Zhu, Banghua, et al. "On optimal caching and model multiplexing for large model inference." arXiv preprint arXiv:2306.02003 (2023). https://arxiv.org/abs/2306.02003