The Impacts Of Embedding Models On Semantic Caching

Shervin Ghaffari
Amirkabir University of Technology
shervinghaffari79@gmail.com

Ali A. Noghabi
Amirkabir University of Technology
a.noghabi2002@gmail.com

Maryam Sadeghabadi
Sharif University of Technology
miryamsadeghi82@gmail.com

Abstract

Recent advancements in generative AI, particularly with large language models (LLMs) like GPT-4o, Claude AI Sonnet, and Gemini 1.5, have unlocked new opportunities across various fields. However, these models' stateless nature often leads to redundant computational costs and delays when responding to semantically similar queries. Caching systems can address this limitation by improving response times and reducing resource consumption. This paper presents a modular caching solution developed using GPTCache, an open-source Python library, to evaluate the performance of various embedding models. By implementing SQLite for caching, FAISS for vector storage, and SbertCrossencoder for similarity evaluation, we measured cache hit-and-miss ratios, token savings, and cost efficiency. Our results demonstrate the effectiveness of caching in enhancing LLM performance and highlight its potential in real-world applications like chatbot design and knowledge-sharing systems.

Introduction

Generative AI has witnessed significant progress due to the development of advanced large language models (LLMs), such as GPT-4o, Claude AI Sonnet, and Gemini 1.5. These models offer remarkable capabilities across a range of applications, from content creation to conversational agents. However, their stateless nature poses a critical challenge—processing semantically similar queries requires repeated computations, leading to increased latency and computational costs. Caching systems provide an effective solution by storing previously generated responses and efficiently managing query-response pairs. By leveraging embeddings to identify semantic similarities, caching can minimize redundant computations, thus improving LLM usability and cost-efficiency. This paper explores the implementation of a modular caching system using GPTCache, focusing on key metrics like cache hit-and-miss ratios, response latency, and token savings. Through extensive experimentation using the Quora Question-Paired dataset, we demonstrate the value of caching in optimizing LLM-based systems while discussing its broader implications for industry applications.

Methodology

1. **Data Manager Architecture**:
   - CacheBase
   - VectorBase
2. **Implementation Choices**:
   - **SQLite**: Selected for efficient query caching.
   - **FAISS**: Chosen for managing vector embeddings due to its robust vector similarity search capabilities.

- **SbertCrossencoder**: Used as the primary similarity evaluation model, with a similarity threshold of 0.7 for distinguishing between cache hits and misses.

3. **Eviction Policy**:
   - Employs least-recently-used (LRU) or first-in-first-out (FIFO) strategies to manage cache size and ensure relevance of cached data.
4. **Layered Caching Mechanism**:
   - The two-layer caching system enhances hit rates by leveraging multiple embedding models. The L1 cache stores initial embeddings, and if a miss occurs, L2 embeddings are queried before resorting to the LLM.

The full implementation of the caching system, including the use of GPTCache, SQLite, FAISS, and SbertCrossencoder, can be found on the project's GitHub repository.

Results

As a result, we successfully stored pairs of queries along with their respective responses in our caching system. After the cache became full, we provided the system with both uncached, semantically similar, and dissimilar queries to calculate cache hit-and-miss ratios. The caching system's effectiveness depends on the short response time latency, small vector dimensions, a large number of saved tokens, high cache hit-and-miss ratios for similar and dissimilar pairs, and low LLM costs for prompt and completion tokens.

Tables

The embedding models were evaluated based on these criteria, as detailed in the table below:

Conclusion

We look forward to enhancing industry trust in LLMs by implementing a system that accurately caches hits and misses, reducing unnecessary repetitive requests and leading to smoother, more cost-effective operations.