ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

A Survey on Hybrid Caching Techniques to Reduce Latency in Large Language Model Systems

Dr. Chaitanya Udatha[1], Krithi Chippada[2], Satvik Dabbara[3]

[1,2,3] Information Technology, Mahatma Gandhi Institute of Technology (MGIT), Hyderabad, India.

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150500032

Received: 30 April 2026; Accepted: 05 May 2026; Published: 26 May 2026

ABSTRACT

Large Language Models (LLM) have vast applications in diverse fields such as text summarization and generation, generative and conversational Artificial Intelligence (AI) and Natural Language Processing tasks. However, generation of content for each real-time task causes high computational cost and latency in LLMs.

To address this drawback, the most effective solution proposed was - caching. Caching mechanisms were introduced to reuse a response instead of computing it for each redundant task. This survey explores various caching strategies from traditional key-value based techniques to the advanced hybrid strategies.

The paper highlights the effectiveness of caching techniques in improving the overall performance of LLM systems. Through this survey hybrid caching mechanism is found to be most useful with an estimate of 15-25% reduction in latency and 10-20% improvement compared to traditional caching mechanisms.

Keywords: Large Language Models, caching, semantic similarity, hybrid caching, query optimization, NLP.

INTRODUCTION

LLMs have become integral part of modern AI systems by enabling features such as chatbots, text generation, agentic workflow and virtual assistants. They use complex transformer architecture, which provides strong contextual understanding for the machines [1], but at the same time they are complex and increase computational costs and latency, making them resource-intensive for real time application. [2].

In real-world applications, users often input repetitive queries which are restructured and semantically similar. Computation for each redundant query leads to redundant computation resulting in inefficient use of resources. Hence, caching techniques are used to reuse the already computed and stored response when similar queries are encountered [3].

Initially, tradition techniques like key-value caching techniques were used that performed exact word to word match although they have been efficient, they failed to handle variations in natural languages queries. Hence recent research focused on semantic similarity to understand textual resemblance and perform better cache retrievals [4].

However, existing approaches still face challenges in balancing computational efficiency and similarity accuracy while handling paraphrased queries.

This research presents a comprehensive hybrid caching techniques as a solution for reducing the latency in LLM systems, focusing on analyzing their effectivness in handling semantically similar and paraphrased queries by proposing lightweight solutions such as multi-level n-gram similarity mechanisms while maintianing low computational cost.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Background

Large Language Models

LLMs are a type of deep learning models that are used for generating natural language texts. In order to do it, they use a transformer architecture, which relies on self-attention mechanisms to capture semantic relationships between words and their sequences. This architecture allows high performance on tasks such as text generation and chat systems [1].

Transformer architecture is computationally very expensive as the complexity increases with the length of the input. Often, in real-world scenario, input is very lengthy [2] therefore, to compute responses to such input LLMs scale in size and capabilities which causes latency. Optimizing their efficiency becomes important.

Caching in LLMs

Caching techniques are used in LLMs by storing most frequent inputs along with their response in a high-speed memory to reduce computation for redundant queries. Cache stores query along with its response in the memory, but it gets difficult identify redundant query that is paraphrased even though they are semantically similar [4]. To solve this issue semantic similarity techniques must be implemented to identify whether a pair of queries are similar [5] while ensuring that these techniques are not introducing new complexities.

LITERATURE SURVEY

The rapid advancement in LLM has improved NLP systems significantly. They are transformer-based architecture enables model to capture contextual relationship between words [1]. Even though it works effectively in generating human-like text, it introduces high computational cost as the complexity increases quadratically with input length which causes latency in the systems [2]. While transformer models provide strong contextual understanding, their high computational cost motivates the need for optimization techniques to improve efficieny.

To address these challenges many optimization techniques have been proposed. One such method is attention mechanisms and inference acceleration to reduce computational overhead. A component called Flash Attention speeds up attention computation by restructuring how attention is calculated which optimizes memory usage. Another component called speculative decoding predicts multiple tokens in parallel and verifies them efficiently, this reduces latency [6]. Although it improves efficiency, it does not eliminate redundant computations of similar user queries. This highlights the need for alternative optimization solutions. In contrast to model-level optimizations, caching based approaches aim to reduce redundant computations at application level, making them more effective for repeated queries.

Earlier caching solutions were introduced for storing and retrieving responses for queries. It is widely used in caching search engine queries which reduces response time by reducing repeated computations [4]. Further [7] showed that user queries have locality i.e., similar queries occur frequently. Earlier caching followed exact match, user query and the corresponding responses is hashed and stored together. Only when exact query was given as input, this approach worked, but real-world queries are often paraphrased, this limits the system's ability to handle variations in natural language queries. Compared to optimization techniques, caching reduces redundant computations, but exact matching methods fail to handle semantic variations in user queries.

To overcome this limitation, cache management strategies such as Least Recently Used (LRU) are used to improve storage and retrieval efficiency [8]. This method works by retaining most frequently used in storge and makes them easily accessible. Although, this method improved cache performance, it still does not address the issue that most often user queries are paraphrased and hence this system fails to handle it. Unlike basic caching methods, more advanced similarity-based techniques attempt to capture relationships between queries beyond exact matches.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

To address this limitation, user queries semantic nature must be captured and handled, hence techniques like embedding models are used which represent user query in a vector space and compute its similarity by measuring cosine distance. Models like BERT [9] and sentence BERT [10] are used to capture semantic meaning between two user queries and the cached result will be retrieved based on its cosine similarity. Although it has reduced computation for redundant queries, generating embeddings and performing similarity calculations using transformer models is computationally expensive. Compared to traditional caching methods, semantic caching methods improve accuracy but introduce significant compuational overhead, making them less suitable for real-time systems.

To reduce this computational overhead, hash-based similarity techniques were explored. Minhash was proposed by Border in 1997 [11] estimates the similarity between two queries using Jaccard similarity. It provided a lightweight alternative technique to compare similarity between two token sets. Another technique called Simhash proposed by Chairkar [12] improves efficiency by generating compact fingerprints that detect near duplicate texts quickly. While these methods capture surface level similarity, they fail to capture deep semantic relationships. Compared to semnatic approaches, hash-based methods offer lower computational cost but lack the ability to capture contextual meaning.

To bridge this gap between efficiency and semantic relationships, retrieval-based similarity methods such as FAISS were introduced. This method is used to quickly predict the nearest neighbors in a high-dimensional embedding space [13]. Another method called Dense Passage Retrieval (DPR) [14] uses dense vector representations to retrieve accurate similarity results efficiently. Models like Universal Sentence Encoder are lightweight models provide faster embedding computation compared to transformer models [15]. Recent research has explored efficient retrieval and similarity optimization techniques for LLM by focusing on improving scalability and reducing latency in large-scale deployments by enhancing similarity matching and retrival mechanisms [16]. Such adavancements highlight the growing importance of efficient caching and retrieval strategies in modern LLM applications. These models improve speed and scalability. However, they still cause computational overhead as they rely on embedding computation. Compared to hash-based methods, retrieval-based approaches improve semantic accuracy but continue to introduce computational overhead due to embedding dependency.

Caching systems like GPTCache are introduced to integrate embedding-based similarity mechanisms directly into caching mechanisms. It retrieves cached responses for semantically similar queries by using vector representations of queries, hence improving response time [3]. Computation overhead still exists as it works with embeddings and this remains a concern for large-scale systems. Compared to earlier methods, GPTCache improves semantic retrival and attempts to reduce the computational cost by minimizing unnessary semantic matching calls. However it still relies on embedding-based similarity and does not fully eliminate the associated computational overhead.

To address these limitations individually, a combination of techniques is proposed in MinCache. In this method the three progressive layers are introduced – exact matching compares word-to-word to retrieve cached response, if it fails resemblance between two queries is compared to retrieve cached response else, semantic comparison is performed. This significantly reduces the need for embedding computation as in most of the cases, exact and resemblance handle cached responses [17]. Compared to previous approaches, MinCache achieves a better balance between efiiciency and accuracy. However it still struggles with paraphrased queries due to limitations in resemblance matching techniques.

Overall, the existing approaches significantly differ based on their trade-off between computational efficiency and similarity accuracy. Traditional caching techniques offer fast retrival but fail to handle variations in the linguistics of natural languages. Semantic and retrival based methods improve similarity detection but introduce significant computational overhead due to the usage of embedding operations. Hash-based techniques provide efficient alternatives but lack deeper contextual understanding. Hence, the hybrid caching approaches attemps to combine these stengths. However it still struggles to handle paraphrased queries. This survey aims to highlight a need for a balanced resemblance-based techniques that can enhance similarity detection whie maintaing low computational cost.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Table 1. Comparison of Caching Techniques.

Technique Type	Method Used	Advantages	Limitations
Traditional Caching [4][7][8]	LRU, Exact Match	Fast, simple implementation	Cannot handle query variations
Semantic Caching [3][9][10][15]	BERT, Sentence-BERT	High accuracy, understands meaning	High computational cost
Hash-Based Methods [11][12][5]	MinHash, SimHash	Fast, scalable	Limited semantic understanding
Retrieval-Based [13][14]	FAISS, DPR	Efficient large-scale similarity	Requires embedding computation
Hybrid Caching [3][17]	Combined approaches	Balanced speed and accuracy	Threshold tuning, complexity issues

Table 1. summarizes the existing system and shows a clear comparison between various caching techniques for LLMs and their advantages and limitations. This concludes that out of all the existing methods; hybrid caching is the most effective and provides a balance by reducing the latency without introducing new computational complexity.

Lımıtatıon in Exıstıng System

MinCache has proposed the most effective cache solution to reduce latency in LLM system without introducing computational overhead. However, its resemblance matching layer only compares based on the number of similar words between two queries, it doesn't consider word orders. Most often user queries are a paraphrased, this results in missing the cache and calling semantic caching mechanism or it results in false positive cache hits which is not desirable.

Proposed Improvements

To address these limitations several improvements are proposed. Unlike the existing systems that heavily rely on either exact matching or computationally expensive semnatic matching that uses embeddings, the proposed improvements focus on enhancing resemblance-based techniques using lightweight and multi-level similarity mechanisms. This is the novelty of porposed approach as it focues on improving similarity detecion for paraphrased queires while maintaing lower computational costs.

One method is to introduce token based semantic features in the resemblance layer. Bigram shingling and multi-gram shingling can be used in resemblance layer to consider word order similarity to improve matches for structurally varied queries.

Positional penalty can be added to reduce false positive cache hits by penalizing differences in word order. Paraphrases-aware techniques such as synonym detection can be integrated to improve resemblance performance by identifying semnatically equivalent queries with different meaning.

Cache clustering techniques aim to improve retrival efficiency. Instead of treating each query indepenetly, this method groups similar queries into cluster based on their structural or semantic similarity. Clustering allows the system to organize cache entries into meaningful groups enabling faster lookup and reducing search space during retrieval. When a new query is recieved instead of matching it with every other query in the cache, they can be matched directly with queries in the closest cache cluster.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Additionally adaptive threshold methods are used to dynamically adjust cache hits based on various scenarios. Query normalization techniques can be used to improve cache efficiency by standardizing inputs.

Table 2. Analytical Estimation of Proposed Improvements.

Proposed Improvement	Purpose	Predicted Impact (%)
Bigram & Multigram Shingling	Capture word order and context	Potential 10–15% improvement in resemblance accuracy
Paraphrase Awareness (Synonym Detection)	Detect semantic equivalence	Potential 15–25% improvement in cache hit rate
Adaptive Threshold Selection	Dynamic similarity tuning	Potential 10–20% reduction in false matches
Query Normalization	Standardize input queries	Potential 5–10% improvement in matching consistency
Cache Clustering	Group similar queries	Potential 10–15% faster retrieval time

Table 2. presents and predicts the impact of the proposed improvement and their overall system performance. These improvements focus on similarity detection, accuracy, increase cache hit rates and reduce unnecessary computational overhead.

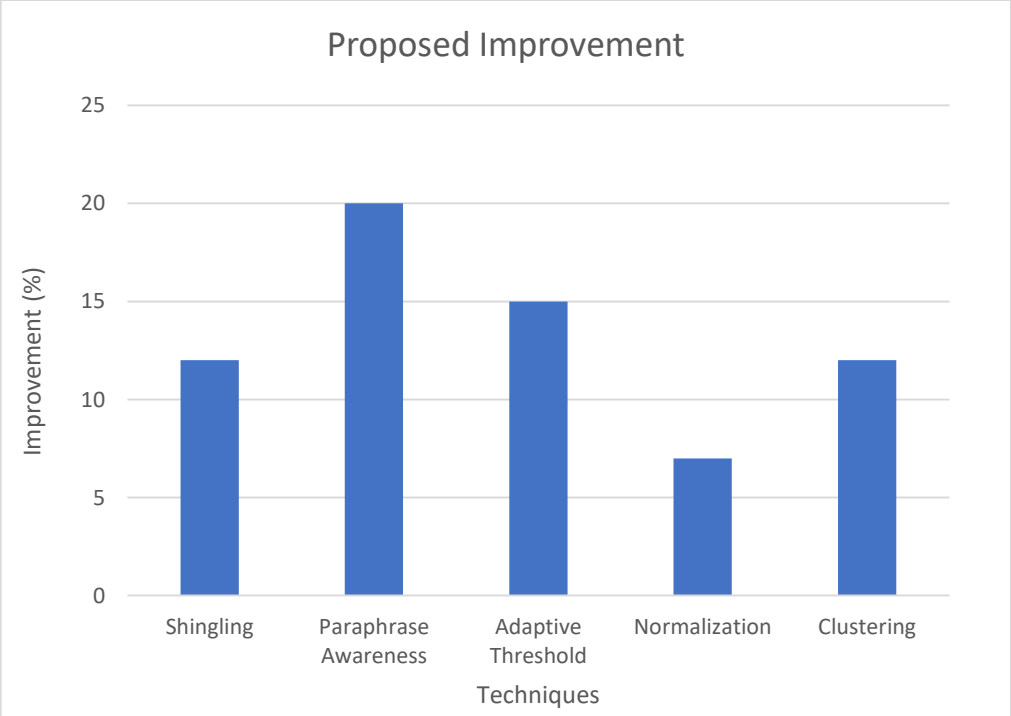

Fig 1. Predicted Performance Improvements of Proposed Caching Enhancements.

Fig 1. Illustrates the expected performance improvements of each of the proposed techniques. This bar graphs shows that the paraphrase awareness technique shows the highest possible improvement followed by adaptive threshold technique and shingling techniques. Query normalization and cache clustering also contribute to performance gains by improving matching consistency and retrieval speed. Overall, the figure highlights the effectiveness of combining multiple lightweight techniques to enhance similarity detection while maintaining low computational overhead.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Table 3. Predicted Computational Complexity of proposed techniques.

Proposed Improvement	Purpose	Estimated Computational Cost (%)
Bigram & Multigram Shingling	Capture word order and context	10–20% (Low)
Paraphrase Awareness (Synonym Detection)	Detect semantic equivalence	20–30% (Moderate)
Adaptive Threshold Selection	Dynamic similarity tuning	5–10% (Very Low)
Query Normalization	Standardize input queries	5–10% (Very Low)
Cache Clustering	Group similar queries	10–15% (Low)

Table 3. represents the estimated computational complexity of each proposed technique. It can be observed that methods such as adaptive threshold and query normalization are least complex, followed by cache clustering and bigram and multigram methods.

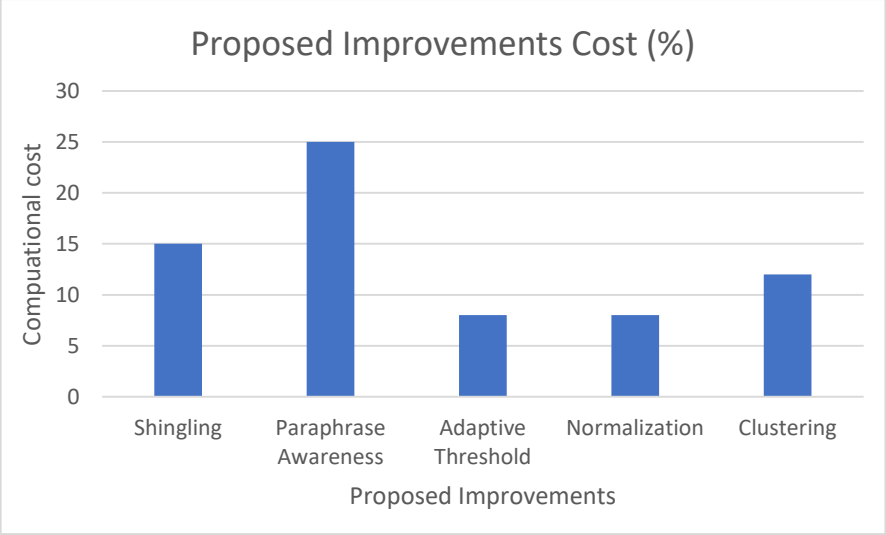

Fig 2. Computational Cost Comparison of Proposed Caching Improvements.

Fig 2. graph represents the estimated cost of each of the proposed improvement technique with adaptive threshold being the least complex compared to methods such as paraphrase awareness techniques.

Comparative Analysis and justification of Proposed Approach

The selection of proposed approaches is supported by the trade-off between their predicted performance improvements and computational cost as illustrated in fig 1 and 2. Paraphrase-aware similarity provide the highest improvements in cache hit rate but they also introduce relatively higher computational overhead. Approaches suchs as query normalization and adaptive threshold offer lower computational cost but their individual impact on performance is significantly low.

Shingling based techniques have a moderate improvement while maintaining low compuational complexity and simultaneously improving their performance in accurate paraphrase detection. Additionally cache clustering contributes to improved retrieval effiency with manageable overhead.

By comparing both the predicted performance improvements and their computational cost techniques, it is very evident that no single method provides the most optimal solution independently. Hence the research proposes a combination of shingling and cache clustering techniques must be used to provide a balanced solution to improve

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

the performance of LLMs while keeping the complexity low.

Architecture or Workflow of Proposed Approach

Based on this analysis, a lightweight hybrid workflow combining shingling and cache clustering techniques is proposed to improve similarity detection while maintaining low computational complexity.

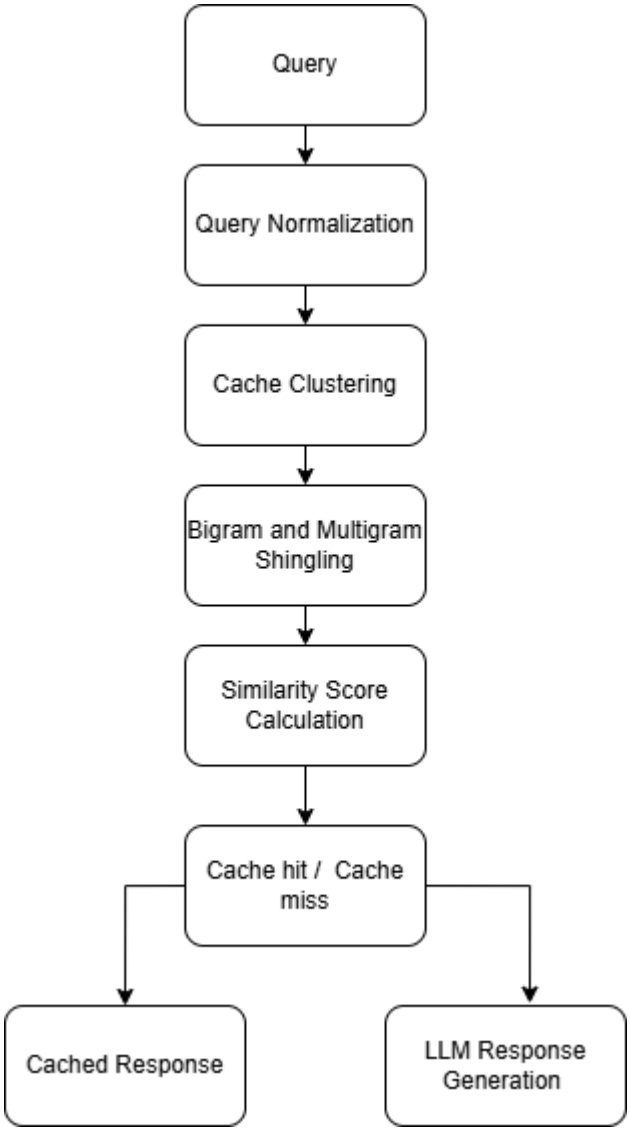

Fig 3. Proposed workflow for Hybrid Cache System for LLMs

Fig 3 illustrates the proposed workflow for a Hybrid Cache System for LLMs dervied from the comparitive analysis on the basis of effectiveness and computational efficieny. The system integrates query normalization, cache clustering and shingling methods to effectively identify similar queries while minimizing computational overhead. Based on the caclulated similarity scores the system will decide to retrieve the response form cache or LLM.

CONCLUSION

This survey of caching solutions for LLMs have shown that multiple caching solutions were proposed to reduce latency and improve speed, scalability and efficiency but, most of the earlier solutions failed to handle semantic relationships in queries to compare its similarity. Embedding based models were introduced as a solution to handle semantic relationships but they introduced computational overhead.

To reduce this hash-based method were proposed as a lightweight solution but they didn't capture deeper semantic relationships. Retrieval-based methods were introduced as another solution but they still relied on

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

embeddings which resulted in computational overhead. Hence hybrid caching mechanism is proposed as a balanced solution to capture semantic relationships only when it is necessary without introducing additional complexity, particularly shingling based similarity methods and cache clustering techniques are propsed as an effective solutions.

This combined approach demonstrates significant improvements by achieving 15-25% reduction in latency and 10-20% increase in cache hit rates compared to traditional semantic caching methods. To prevent false positive cache hits, additional optimization techniques can be integrated into hybrid caching mechanism to improve the overall performance of LLM systems.

Future Scope

Future work may include implementing the proposed hybrid caching techniques and experimentally validating it against benchmark datasets to evaluate their performance againts existing systems like GPTCache and MinCache. It can be further improved in parameters such as latency reduction, false positive reduction, cache hit rates and computational efficiency. It can also explore real-time deployment and scalability testing in production-scale environments.

REFERENCES

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
2. Zhou, Z., Ning, X., Hong, K., Fu, T., Xu, J., Li, S., ... & Wang, Y. (2024). A survey on efficient inference for large language models. arXiv preprint arXiv:2404.14294.
3. Bang, F. (2023, December). 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) (pp. 212-218).
4. Markatos, E. P. (2001). On caching search engine query results. Computer Communications, 24(2), 137-143.
5. Chen, G., Chen, G., Wu, D., Liu, Q., Zhang, L., & Fan, X. (2021, July). An improved Simhash algorithm based malicious mirror website detection method. In Journal of Physics: Conference Series (Vol. 1971, No. 1, p. 012067). IOP Publishing.
6. Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in neural information processing systems, 35, 16344-16359.
7. Xie, Y., & O'hallaron, D. (2001). Locality in search engine queries and its implications for caching (No. CMUCS01128).
8. Mookerjee, V. S., & Tan, Y. (2002). Analysis of a least recently used cache management policy for web browsers. Operations Research, 50(2), 345-357.
9. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers) (pp. 4171-4186).
10. Reimers, N., & Gurevych, I. (2019, November). Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP) (pp. 3982-3992).
11. Broder, A. Z. (1997, June). On the resemblance and containment of documents. In Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171) (pp. 21-29). IEEE.
12. Charikar, M. S. (2002, May). Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (pp. 380-388).
13. Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE transactions on big data, 7(3), 535-547.
14. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020, November). Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP) (pp. 6769-6781).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

15. Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Kurzweil, R. (2018). Universal sentence encoder. arXiv preprint arXiv:1803.11175.
16. Liu, Y., Wu, J., He, Y., Gong, R., Xia, J., Li, L., ... & Li, K. (2025). Efficient inference for large reasoning models: A survey. arXiv preprint arXiv:2503.23077.
17. Haqiq, K., Jahan, M. V., Farimani, S. A., & Masoom, S. M. F. (2025). MinCache: A hybrid cache system for efficient chatbots with hierarchical embedding matching and LLM. Future Generation Computer Systems, 170, 107822.