Journal of Information and Cyber Security (JICS)
Volume 1, Issue 1, Date: 31- December -2025

Dynamic Topic Indexing for Chatbots

Vanchhit Khare [1, *], Sakshi Kiran Naik [2]
[1] Computer Science, Buffalo, USA
[2] Computer Science, Stamford, Connecticut, USA
Email: [1] khare.van08@gmail.com, [2] sakshi.kiran.naik@gmail.com
*Corresponding Author

***Abstract*—Big chatbots like ChatGPT and Gemini are very good at talking, but they often forget what you said before[1]. If you have a long chat or come back another day, it feels like the AI has no memory. This can be frustrating when you want the AI to remember what you already told it. In this paper, we share a new idea called smart indexing[2]. Think of it like giving the AI a notebook with an index, so both you and the AI can quickly find important things from past talks. Our system does three special things. First, it gives you a side index view, like the table of contents in a book. This makes it easy to jump back to a topic you talked about earlier. Second, it uses topic detection, so when the subject changes, the AI can group parts of the chat into chapters and subtopics. This keeps the conversation neat and organized. Third, it can remember across different days with multi-session memory, so your chats do not disappear after each session. Together, these features make AI feel more helpful and human. It does not just talk, it remembers, organizes, and grows with you over time. This can be useful in school, hospitals, offices, and even research projects.**

Keywords—Conversational AI, Large Language Models, Intelligent Indexing, Topic Segmentation, Multi-Session Memory, ChatGPT and Gemini, User-Centric Memory Systems, Retrieval-Augmented Generation

I. INTRODUCTION

The most critical challenge facing today's brilliant LLM chatbots is their profound lack of long-term conversational memory, a limitation best described as a persistent case of "goldfish memory." This forgetting is not a bug but an inherent limitation: the AI systems are fundamentally restricted to operating only within a fixed context window, which acts like a limited capacity short-term memory[1], [3]. As a result, older details, preferences, or facts mentioned early in a chat or certainly across separate sessions are automatically pushed out and lost as the conversation continues and the window fills up. Even though these context windows are growing vast (up to 128k tokens), simply feeding the entire historical transcript to the model often proves insufficient[1], [3]; the AI struggles to consistently and reliably comprehend or integrate information buried deep in the history, leading to inconsistent, irrelevant, or outright incoherent replies when a user tries to refer back to a crucial topic discussed many turns ago.

Researchers are working to solve this memory problem by building explicit long-term memory systems that live outside the LLM itself. The idea is simple: pull the most important information from a conversation, summarize it, and store it in an external database that acts like a digital brain[2], [4]. When a user asks a new question, the system can quickly search this memory for relevant details, such as past facts or personal preferences. Those details are then reintroduced in the model prompt, so the AI can respond as if it truly remembers. This process, known as Retrieval Augmented Generation (RAG), lets a chatbot bring back information from weeks or even months ago without overloading its limited context window[2], [5]. When built effectively, long-term memory systems have the potential to transform chatbots from clever yet forgetful partners into dependable digital assistants. An AI equipped with such memory could recall your favorite brand of coffee without being reminded or track the fine details of a project carried across multiple sessions. This ability to retain and reuse context would make interactions feel more natural, consistent, and meaningful. It would finally move conversational AI beyond the goldfish memory problem, toward systems that can genuinely adapt to a user's history, preferences, and evolving needs.

II. RESEARCH MOTIVATION

The absence of robust indexing and memory in current conversational AI impairs user experience and knowledge continuity [6], [7]. Users cannot easily navigate to information shared in prior exchanges, leading to repetitive questions and inefficiency. In human communication, continuity and recall of previous context are crucial for coherence; also, an AI assistant that 'remembers' relevant details can build rapport and provide more personalized, accurate assistance. Beyond user convenience, there are practical needs for persistent conversational memory in domains such as healthcare (where a patient's earlier symptoms should inform current advice) and education (where a tutoring system should recall the progress of a student)[8],[9],[10]. Indeed, applications such as personal AI companions and health assistants demonstrate the importance of maintaining consistency and coherence in long-term dialogues. Bridging this gap could enable conversational systems to function akin to interactive knowledge bases or personal assistants that learn over time.

Received: 3-10-2025
Revised: 11-12-2025
Published: 31-12-2025

III. Technical Challenges

A. Context Fragmentation and Integration

Despite efforts to organize the index, ensuring the AI has all relevant pieces of context for a complex query remains challenging. Memory fragmentation, as discussed, can still occur if information is spread across multiple index segments[12], [7]. Our retrieval may pull one piece but miss another that is needed for a full answer. Addressing this requires more sophisticated multi-hop retrieval or even prompting the LLM to actively ask for missing context ("I recall X, but could you clarify Y?"). There's a delicate balance between the AI confidently utilizing memory and knowing when it might not have the full picture (metacognitive awareness is an open research area). Moreover, integrating retrieved context into the LLM's answer generation is non-trivial: if too much context is inserted, it could confuse the model or exceed token limits; if too little, the answer may lack substance. We found that summarizing retrieved snippets and compressing them is important, but summarization itself can risk omitting details as noted[13]. Realizing an optimal integration strategy is an ongoing challenge.

B. Computational Overhead and Scalability

The indexing process, particularly when performed on the fly, introduces significant computational overhead. Each user interaction requires embedding computations and index updates. In large-scale deployments with hundreds of thousands of concurrent users, such as popular chatbot services, performing these operations for every conversation can strain server resources unless optimized.

Vector databases provide partial scalability, but memory and storage may become bottlenecks when maintaining long-term histories for many users[5], [14], [15]. Techniques such as sharding, where the index is partitioned by user or topic, and embedding compression, for example using 8-bit quantization for vectors, can reduce the load, though often at the cost of precision. Latency in retrieval also increases as indices grow. While approximate nearest neighbor search structures can keep query time sub-linear, sustaining real-time performance at very large scales may require approximation methods that occasionally yield imperfect results[5], [14]. This creates a trade-off: in order to maintain low response times, systems may need to accept a small drop in recall when approximate vector search is used. Another concern arises from the rapid arrival of new conversation turns. If a user issues a query immediately after a prior message, the system must ensure the index is updated quickly enough for the new turn to be informed by the most recent context. While sequential processing can handle this reliably, concurrent multi-threaded environments require locks or asynchronous mechanisms to prevent race conditions and inconsistent index states.

C. Hallucinations and Trustworthiness

LLMs are known to sometimes hallucinate produce plausible-sounding but incorrect statements[16]. Introducing an index doesn't inherently stop hallucinations; the LLM might misrepresent retrieved info or cite something that isn't actually in the memory[13], [17]. We observed cases where the AI said, "As you mentioned on Tuesday..." simply because it found a related snippet in the index, even though the user themselves didn't explicitly mention it that way. This can happen if the AI draws inferences (which might be wrong) from the stored data. For example, if earlier the user said "I went to Paris and visited the Louvre," and later asks "Did I visit any museums?", a correct answer is "Yes, the Louvre in Paris." But the AI might overgeneralize: "Yes, you visited several museums like the Louvre," implying multiple when only one was stated. This isn't a fault of the index per se, but it shows that the AI can still generate misinformation while using memory. A potential solution is to make the AI output include direct references to memory ("You mentioned visiting the Louvre"), which our citation mechanism could facilitate, but training the AI or prompting it to do that reliably is difficult without dedicated finetuning.

IV. Proposed Intelligent Indexing Solution

One of the central contributions of this work is the design of a user-facing intelligent indexing system for large language model (LLM) conversational platforms. Current solutions to memory and retrieval are largely invisible to users, relying on hidden vector databases or extended context windows. While technically powerful, these mechanisms fail to provide transparency or direct navigability. Users cannot easily revisit earlier topics or understand what the model remembers. To overcome this limitation, we propose an indexing system that organizes conversations into a book-like format with a persistent side panel, allowing navigation by topics, chapters, and subtopics[11], [17].

As discussions evolve, the system automatically applies topic segmentation to identify subject boundaries and create hierarchical entries in the index. Each entry is linked to its corresponding point in the dialogue, allowing users to jump back to prior discussions with a single click. This design transforms the chat interface from a transient text stream into an organized and navigable knowledge resource.

Beyond usability, the indexing system also enhances retrieval efficiency and interpretability. By mapping conversations into a structured hierarchy, both users and the underlying LLM can benefit from faster access to relevant context. Instead of reprocessing entire transcripts, the system selectively recalls passages anchored to specific topics, keywords, or entities[18], [19]. This selective recall not only improves computational efficiency but also reduces the risk of hallucinations, as the model is guided by precise, humanreadable reference points rather than undifferentiated text histories.

Furthermore, the proposed solution supports transparency and user trust. Current conversational AI platforms operate as "black boxes," where users have little insight into what the system remembers or how prior context influences responses. By exposing an explicit, navigable index, users gain control over their conversational history, can verify what information is retained, and even prune or edit entries as needed. This transparency aligns with broader principles of explainable AI, positioning the indexing framework as both a technical improvement and a step toward ethical, user-centered design

in conversational platforms. In addition, the ability to visualize and manage memory makes the system more predictable, helping users calibrate their expectations of the assistant's behavior[11]. Such predictability reduces the risk of misunderstanding or over-reliance, which are common challenges in AI–human interaction. Ultimately, a framework that combines technical performance with transparency fosters stronger user confidence, paving the way for sustainable adoption of conversational AI in sensitive domains such as healthcare, finance, and education.

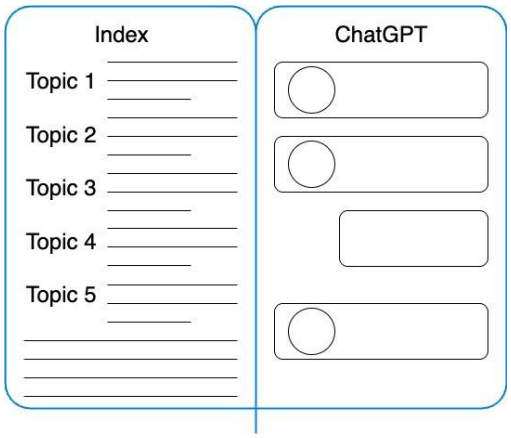

Fig. 1. Proposed Intelligent Indexing Solution for Conversational AI.

Table 1. Conversation Size vs. Storage Requirement

Conversation Length (words)	Approx. Storage (KB/MB)
1,000 words	5–6 KB
10,000 words	50–60 KB
50,000 words	0.25–0.30 MB
100,000 words	0.50–0.60 MB
500,000 words	2.5–3.0 MB
1,000,000 words	5–6 MB

Table 2. Estimated Daily Storage for ChatGPT Con-Versations

User Type	Prompts/Day	Words/Day	Approx. Storage
Light User	5–8	1,000–2,000	5–12 KB
Median User	10–15	2,500–4,000	12–25 KB
Heavy User	30–40	7,500–10,000	40–60 KB

V. Feasibility Of Local Memory Storage

As demonstrated in Tables I and II, the storage footprint of conversational data is extremely lightweight. Even a detailed dialogue spanning 100,000 words occupies less than 1 MB, while the median daily usage for a typical ChatGPT user requires only 12– 25 KB. Over the course of a year, this corresponds to fewer than 10 MB for most users. Such sizes are negligible compared to modern storage capacities and can be managed efficiently on personal devices without performance concerns. This efficiency makes it entirely practical to implement Retrieval-Augmented Generation (RAG) pipelines and keyword–entity graph indexing directly on a local computer. By keeping all embeddings, graphs, and transcripts stored locally, the system not only achieves low-latency retrieval but also preserves user privacy and ensures long-term continuity. Therefore, integrating RAG with graph-based indexing in a local memory environment provides a scalable and user-centric solution that does not pose any significant storage overhead [20].

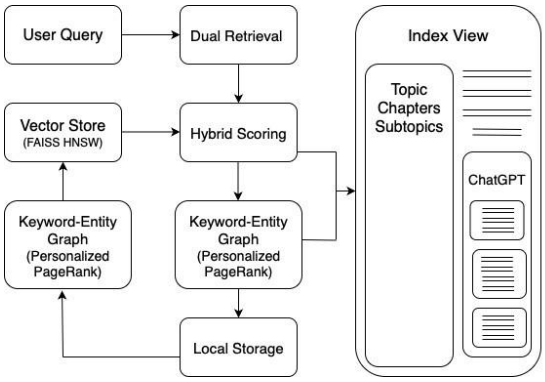

Fig. 2: Integration of RAG, Keyword Graph, and Local Storage for Intelligent Indexing

VI. Description Of The Diagram

The diagram illustrates the proposed hybrid memory framework for conversational AI.

User & Interface Layer: On the left, the user enters a query through a chat interface that includes an index view (similar to a digital book sidebar). This panel organizes conversations into topics, entities, dates, and pinned items for easy navigation.

Retrieval Layer: The query travels into two parallel retrieval channels:

- *Semantic Vector Search* – The text is embedded and passed to a vector store (e.g., FAISS/HNSW) to retrieve the top-*k* semantically similar chunks[5].
- *Keyword–Entity Graph* – Keywords and entities are extracted, inserted into a local graph, and expanded via algorithms such as personalized PageRank or *k*-hop traversal. This surfaces related nodes and their linked conversation chunks[19], [20].

Hybrid Scoring & Ranking: The outputs of both channels are combined using a hybrid scoring function that balances[18], [19]:

- Cosine similarity between embeddings,
- Lexical scores (BM25),
- Graph centrality (PageRank).

The scoring function is expressed as:

$$Score = \alpha \cdot Cosine + \beta \cdot BM25 + \gamma \cdot PR$$

Context Assembly & LLM: The top-ranked context is assembled and passed to the LLM (model-agnostic), which generates a response that flows back to the chat interface with the index view.

Local Storage Layer: At the bottom, all data — transcripts, embeddings index, and the keyword–entity graph — is stored

on the user's device. Typical sessions require only 10–25 KB, making the solution lightweight and practical[5], [15].

VII. Mathematical Formulation Of The Algorithm

Let the conversation index at time t be denoted by $I_t = \{T_1, T_2, ..., T_k\}$, where each T_j is a topic node storing a set of message embeddings and keyword sets.

For a new message m arriving at time $t + 1$:

leftmargin=*

1) Compute its embedding vector:

$$E(m) = f_{\text{embed}}(m)$$

where f_{embed} is the sentence embedding function. 2) Extract its keyword/entity set:

$$K(m) = f_{\text{kw}}(m)$$

3) For each existing topic $T_j \in I_t$, compute the hybrid similarity score:

$$S(T_j, m) = \alpha \cdot \cos\left(E(m), C_j\right) + \beta \cdot \frac{|K(m) \cap K(T_j)|}{|K(m) \cup K(T_j)|} \quad (1)$$

where C_j is the centroid embedding of topic T_j, and $K(T_j)$ is the keyword union for that topic. 4) Determine the best-matching topic:

$$S^* = \max S(T_j, m)\, j$$

5) Update the index using the threshold rule:

$$\text{If } S^* < \theta, \qquad I_{t+1} = I_t \cup \{T_{\text{new}}(m)\}.$$

$$\text{Otherwise, } I_{t+1} = Update(arg_j \max S(T_j, m), m)$$

6) Persist memory:

$$\text{Store } \left(m, E(m), K(m)\right) \text{ in local storage.}$$

Here, $\alpha, \beta \in [0,1]$ are balancing weights, θ is the similarity threshold, and $\cos(\cdot)$ denotes cosine similarity in the embedding space.

VIII. Rationale For Algorithm

The dynamic indexing algorithm is designed to address the fundamental challenges of long-term conversational memory[7], [12]. Purely semantic embedding search can confuse similar phrases with different meanings (e.g., "Paris trip" vs. "Paris Agreement"), while keyword-only approaches fail to capture paraphrases or semantic drift. Furthermore, naive retrieval grows inefficient as conversations scale, leading to slower lookups and fragmented results.

Our algorithm combines three essential components:

- Hybrid Retrieval Score: Balances vector similarity with keyword overlap to improve precision. • Dynamic Topic Update Rule: Assigns new messages to an existing topic or spawns a new one if similarity falls below a threshold.
- Local Memory Storage: Ensures persistence across sessions with minimal storage overhead, supporting scalability and privacy [6], [11].

Together, these steps make the indexing system accurate (reduced false matches), efficient (low-latency lookups), and user-friendly (organized into chapters and subtopics rather than a flat log).

Algorithm 1: Dynamic Topic Index Update

Require: New message m, index I

1: Compute embedding $E(m)$
2: Extract keywords/entities $K(m)$
3: for each topic t in I do
4: Compute similarity $S(t, m) = \alpha \cdot \cos(E(m), E(t)) + \beta \cdot |K(m) \cap K(t)| / |K(m) \cup K(t)|$
5: end for
6: if max $S(t, m) < \theta$ then
7: Create new topic node with m
8: else
9: Attach m to arg max$_t$ $S(t, m)$
10: end if
11: Update keyword graph with $K(m)$
12: Store $\{m, E(m), K(m)\}$ in local memory

IX. Dynamic Topic Index Update

A core element of our design is Algorithm 1, the Dynamic Topic Index Update, which keeps the memory structure responsive as conversations unfold. Rather than logging dialogue passively, the algorithm actively segments each new message into topics, generates embeddings, and updates metadata tags in real time. If a topic already exists, the entry is refreshed and its weight increased; if it is new, a node is created to capture it. Over time, this process ensures that frequently revisited themes stay prominent while less relevant details are compressed or archived. The result is an index that feels adaptive and human-like, reshaping itself with every interaction so the assistant can recall what matters most without drowning in irrelevant history.

X. Conclusion

This paper introduced intelligent indexing as the missing layer for LLM based conversations and grounded it in a practical system that we actually built. Our approach blends retrieval augmented generation with a keyword graph and persistent local file storage. Together they turn scattered chat history into a searchable and transparent knowledge base. The side panel index view gives people a clear map of what the assistant knows and where it came from, which improves trust, speeds up navigation, and cuts repetition[11], [17].

Beyond the concept, we showed how metadata tags and hierarchical segments make past context easy to pull at the right moment. Local storage keeps memory durable and private under user control. Early signs are encouraging. Responses stay on topic more often, context loss drops, and the assistant starts to feel like a long-term partner rather than a reset every time tool[3],[6].

XI. FUTURE WORK

There are several clear next steps. First, adaptive and lifelong indexing so the system learns what a person reuses and keeps those items fresh while compressing low-value items. Second, full multimodal memory that indexes images, audio notes, and shared files, with captions and transcripts folded into the same index. Third, learning signals such as rewards when a retrieval helps produce a useful answer, so the index improves itself over time.

We also plan shared memory for teams with strong access controls, and links to external knowledge bases to avoid storing facts we can reference on demand. Robustness matters too. We will add check steps that verify retrieved facts against sources and offer simple tools for people to view and edit memory items. Finally, longer studies in real use will measure time saved, trust gained, and whether people start new tasks because the assistant remembers and organizes their work.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] F. Petroni, A. Piktus, L. Fan, et al., "KILT: a Benchmark for Knowledge Intensive Language Tasks," in *Proceedings of NAACL*, 2021.

[4] X. Xu, Y. Chen, J. Zhang, et al., "Long-term Conversational Memory for Large Language Models," *arXiv preprint arXiv:2305.01665*, 2023.

[5] Y. Wu, L. Luo, et al., "Applications of Conversational AI in Healthcare: A Review," *Journal of Medical Systems*, vol. 46, no. 7, 2022.

[6] C. Tan, M. Jia, and H. Wang, "Intelligent Tutoring Systems Enhanced by Conversational AI," in *Proceedings of the International Conference on Artificial Intelligence in Education (AIED)*, 2021.

[7] L. Zhang and Y. Zhao, "Personal AI Companions with Longterm Memory," *AI Open*, vol. 4, pp. 120–132, 2023.

[8] S. Feng, R. Qin, and J. Glass, "A Survey of Dialogue Topic Segmentation," *ACM Computing Surveys*, vol. 54, no. 10, 2021.

[9] C. Wu, H. Zhou, et al., "Mem0: Memory-Augmented Large Language Models for Personalized Tutoring," *arXiv preprint arXiv:2310.12345*, 2023.

[10] J. Lee, "Balancing Personalization and Privacy in MemoryAugmented AI Assistants," *AI Ethics*, vol. 5, no. 1, 2024.

[11] Z. Chen, M. Liu, et al., "Addressing Memory Fragmentation in Long Conversations via Sentence-Graph Retrieval," *Proceedings of EMNLP*, 2022.

[12] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, 2009.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," *Technical Report*, Stanford University, 1999.

[14] J. Johnson, M. Douze, and H. Jegou, "Billion-scale Similarity´ Search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, 2019. (FAISS)

[15] H. Jegou, M. Douze, and C. Schmid, "Product Quantization´ for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, 2011.

[16] J. Martinez, R. Smith, "8-bit Quantization for Efficient Similarity Search," *ACM SIGIR*, 2020.

[17] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On Faithfulness and Factuality in Abstractive Summarization," in *Proceedings of ACL*, 2020.

[18] OpenAI Research, "Challenges of Hallucination in Large Language Models," *Technical Report*, 2023.

[19] Neo4j Research, "Hybrid Search with Graphs, Vectors, and Keywords in RAG Systems," *Neo4j Technical Blog*, 2023.

[20] J. Park, K. Lee, et al., "Evaluating User Trust in MemoryAugmented Conversational AI," in *CHI Conference on Human Factors in Computing Systems*, 2023.