Momentary Contexts: A Memory and Retrieval Approach for LLM Efficiency

"The world is the totality of facts, not of things."
— Ludwig Wittgenstein

Jaepil Jeong
Cognica, Inc.
Email: jaepil@cognica.io
Date: December 7, 2024

Abstract

Recent advancements in large language models (LLMs) have achieved remarkable breakthroughs in natural language processing and artificial intelligence [1]. However, current LLMs rely on cumulative context, which poses structural limitations. This approach often results in inefficient utilization of computational resources and diminished coherence and adaptability in extended interactions. This study proposes an innovative framework that eliminates cumulative context and introduces the concepts of **"memory"** and **"retrieval"** to establish **"momentarily reconstructed minimal contexts."**

In this framework, **memory** refers to the external storage of past interactions exactly as they occurred in a database. This raw data serves as a persistent repository that can be retrieved as needed. **Retrieval** is the process of selectively accessing relevant stored memory to dynamically reconstruct short-term, localized contexts tailored to specific queries or tasks. This aligns with retrieval-augmented generation methods [2] [3] but differs by emphasizing dynamic context reconstruction rather than cumulative accretion.

The proposed approach also highlights the potential of **memory** to function as an interface for external knowledge. By linking memory with domain-specific knowledge, it becomes possible to integrate expert knowledge dynamically into the reconstructed context at every moment, facilitating knowledge-intensive tasks [2]. This enables LLMs to produce not only general responses but also highly specialized and contextually accurate outputs, significantly enhancing their performance in diverse domains.

This study explores the design principles and implementation strategies of this memory-based system and empirically demonstrates its efficacy in improving both performance and efficiency when integrated into LLM architectures. By leveraging memory to incorporate external knowledge dynamically, this research offers a novel paradigm for LLM design, advancing AI models that emulate human memory systems while unlocking new possibilities for external data integration.

Keywords: Large Language Models, Memory, Retrieval, Context Reconstruction, Artificial Intelligence

Limitations of Cumulative Context in Attention Mechanisms

In traditional cumulative context models, the attention mechanism distributes weights across all input tokens, leading to:

- **Weight dilution**: As the input length (n) increases, important recent tokens receive smaller attention weights due to normalization.
- **Noise amplification**: Irrelevant or outdated tokens add unnecessary noise to the attention computation [1].

Attention Mechanism in Cumulative Context

The attention mechanism can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Where:

- Q: Query vector

- K: Key vector
- V: Value vector
- d_k: Dimension of the key vector

For an input sequence $X = [x_1, x_2, \ldots, x_n]$, the attention weight α_i for each token x_i is:

$$\alpha_i = \frac{\exp\left(\frac{qk_i^T}{\sqrt{d_k}}\right)}{\sum_{j=1}^{n} \exp\left(\frac{qk_j^T}{\sqrt{d_k}}\right)} \tag{2}$$

Where:

- α_i: The attention weight for token i.
- q: Query vector.
- k_i: Key vector for token i.
- n: Total number of tokens in the input sequence.

As n grows larger:

- The denominator increases, causing **weight dilution**, where even important tokens have diminished weights.
- Outdated or irrelevant tokens contribute to **noise amplification** in the attention calculation.

Retrieval-Based Context Reconstruction

Retrieval-based approaches dynamically reconstruct the input context X' with a subset of relevant tokens x'_i, reducing the effective input length m $(m \ll n)$. Such approaches relate to retrieval-augmented generation methods [2] [3]. The attention weight in this reduced context is:

$$\alpha'_i = \frac{\exp\left(\frac{qk'^T_i}{\sqrt{d_k}}\right)}{\sum_{j=1}^{m} \exp\left(\frac{qk'^T_j}{\sqrt{d_k}}\right)} \tag{3}$$

Key advantages:

- **Weight concentration**: Smaller m increases the attention weight for critical tokens.
- **Noise reduction**: Irrelevant tokens are excluded, improving focus and coherence.

Computational Efficiency

The computational cost of the attention mechanism is proportional to the square of the input length:

- **Cumulative context**: $O(n^2 \cdot d_k)$
- **Retrieval-based context**: $O(m^2 \cdot d_k)$, where $m \ll n$

Attention Focus and Entropy Analysis

Understanding Attention Focus

In attention mechanisms, the focus on specific tokens is determined by the distribution of attention weights α_i. These weights are calculated as:

$$\alpha_i = \frac{\exp\left(\frac{qk_i^T}{\sqrt{d_k}}\right)}{\sum_{j=1}^{n} \exp\left(\frac{qk_j^T}{\sqrt{d_k}}\right)} \tag{4}$$

Where:

- α_i: The attention weight for token i.
- q: Query vector.
- k_i: Key vector for token i.
- n: Total number of tokens in the input sequence.

The focus of attention is more effective when the weights are concentrated on relevant tokens. However, in cumulative context, the number of tokens (n) grows larger, which can dilute attention weights and reduce focus.

Measuring Attention Focus with Entropy

Entropy is a statistical measure that quantifies the dispersion of a probability distribution. In the context of attention, entropy can be used to measure how evenly attention weights (α_i) are distributed:

$$H = -\sum_{i=1}^{n} \alpha_i \log(\alpha_i) \tag{5}$$

Where:

- H: Entropy of the attention weights.
- α_i: Attention weight for token i.

Interpretation:

1. **High Entropy**:
    - Indicates that attention weights are spread out across many tokens.
    - This often occurs in cumulative context, where irrelevant or outdated tokens are included.
2. **Low Entropy**:
    - Indicates that attention weights are concentrated on fewer, more relevant tokens.
    - This is desirable and more achievable in reconstructed context.

Comparing Cumulative and Reconstructed Contexts

1. Cumulative Context:

In cumulative context, the input size (n) increases as more tokens are accumulated. This leads to:

- Higher entropy $(H_{\text{cumulative}})$ as attention weights are distributed over a larger set of tokens.
- Decreased focus on critical information, which can dilute model performance.

2. Reconstructed Context:

In reconstructed context, only the most relevant tokens $(m \ll n)$ are retained. This results in:

- Lower entropy $(H_{\text{reconstructed}})$ as attention weights are concentrated on a smaller set of meaningful tokens.
- Enhanced focus, improving both computational efficiency and output quality.

Quantitative Comparison

Entropy values for the two contexts can be summarized as:

$$H_{\text{reconstructed}} \ll H_{\text{cumulative}} \tag{6}$$

This indicates that reconstructed context enables a more efficient and effective attention mechanism, concentrating computational resources on relevant information.

Conclusion

By analyzing entropy, we demonstrate that retrieval-based context reconstruction reduces the dispersion of attention weights, leading to:

- Greater focus on relevant tokens.
- Improved computational efficiency.
- Higher overall performance of the model.

This supports the argument that attention mechanisms operate more efficiently in reconstructed contexts compared to cumulative contexts.

Reconstructed Context: Definition and Construction

The reconstructed context is a dynamically selected subset of memory, designed to enhance the efficiency and relevance of attention mechanisms in large language models (LLMs). Additionally, to maintain conversational coherence, the **most recent memory** is always included as an exception, ensuring the model retains immediate context for seamless responses, reflecting natural turn-taking in conversation [4]. The construction of the reconstructed context involves the following steps:

1. Memory Retrieval Using HNSW and BM25

Memory retrieval involves identifying the most relevant past interactions or knowledge from the memory store. Two techniques are employed:

- **HNSW (Hierarchical Navigable Small World Graph)**:
  - Used for fast approximate nearest neighbor search [5] [6].
  - Identifies relevant embeddings based on semantic similarity.
- **BM25**:
  - A traditional lexical matching algorithm, scoring relevance based on term frequency and inverse document frequency [7].

We will discuss how to integrate BM25 and HNSW in a separate section.

2. Inclusion of the Most Recent Memory

To ensure conversational continuity, the most recent memory m_{recent} is **always included**, regardless of its relevance score to reflect the fundamental nature of human dialogue and turn-taking [4]. This guarantees that the model can seamlessly reference the immediate prior interaction.

This design choice is motivated by the fundamental nature of human dialogue. In human-to-human interactions, the meaning and coherence of an ongoing conversation heavily rely on the most recent exchange. Ignoring or omitting the immediately preceding statement disrupts the natural flow of communication, causing confusion and a sense of incongruity. By contrast, ensuring that the most recent turn is consistently preserved mirrors the innate human tendency to continuously build upon the latest conversational cues.

From a cognitive perspective, humans instinctively reference the last utterance to maintain topic continuity and contextual relevance. This habitual pattern is so ingrained that any deviation—such as responding without regard to the previous speaker's last contribution—strikes us as awkward and artificial. In other words, retaining the most recent memory at all times ensures a smoother, more natural interaction. It guarantees that the model's responses remain anchored in the evolving discourse, thereby enhancing both the realism and coherence of extended conversations.

3. Top-K Selection

From the retrieved memory, we select the top K items based on their relevance scores, excluding m_{recent} since it is already guaranteed to be included:

$$\text{Top-K} = \{m_1, m_2, \ldots, m_K\}, \quad \text{where } m_i \text{ has the highest relevance scores and } m_i \neq m_{\text{recent}}. \tag{7}$$

4. Context Length Restriction

The reconstructed context is subject to the model's maximum context length $L_{\max}$. Memory items are truncated if their combined length exceeds this limit, prioritizing:

1. The most recent memory m_{recent}.
2. Relevant memory items $m_1, \ldots, m_K$ in descending order of relevance.

$$\text{Context Length: } \text{len}(m_{\text{recent}}) + \sum_{i=1}^{K} \text{len}(m_i) \leq L_{\max} \tag{8}$$

5. Time-Ordered Sorting

After truncating items to fit within $L_{\max}$, the remaining memory items, including m_{recent}, are sorted in chronological order. This ensures logical flow and temporal coherence.

Final Definition

The **reconstructed context** is formally defined as:

$$\text{Reconstructed Context} = \{m_{\text{recent}}\} \cup \{m_i \in \text{Memory} : \text{Relevant and Chronologically Sorted, Subject to } L_{\max}\}. \tag{9}$$

Advantages of Reconstructed Context with Recent Memory

1. **Relevance**: Ensures that the model focuses on the most pertinent information by leveraging advanced retrieval algorithms.
2. **Continuity**: Guarantees seamless conversational flow by always including the most recent memory [4].
3. **Efficiency**: Avoids exceeding the model's maximum context length, reducing computational overhead.
4. **Temporal Coherence**: Chronological sorting enhances the logical sequence of information.

By incorporating the most recent memory as an exception, this framework ensures conversational continuity while maintaining the benefits of relevance, efficiency, and coherence.

Hybrid Retrieval Strategy: BM25 + HNSW

BM25 Formula and Characteristics

BM25 is a classic keyword-based scoring function [7] widely used in information retrieval. It leverages term frequency, inverse document frequency, and document length normalization to produce a stable and interpretable relevance score.

BM25 Scoring Function:

$$\text{BM25}(D, Q) = \sum_{q_i \in Q} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \left(1 - b + \frac{b|D|}{\text{avgdl}}\right)} \tag{10}$$

- Strengths: Stable recall, clear interpretability, and strong keyword matching.
- Limitations: Lacks semantic understanding or synonym recognition.

Cosine Similarity and HNSW

Cosine similarity measures the angle between embedding vectors, enabling semantic matching beyond exact keywords. Given two vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\text{Cosine Similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \tag{11}$$

This focuses on the direction rather than magnitude, making it suitable for comparing embeddings where length scales may differ.

HNSW (Hierarchical Navigable Small World) graphs facilitate efficient approximate nearest neighbor (ANN) searches [5] in high-dimensional embedding spaces. HNSW provides semantic-rich retrieval, uncovering conceptually related documents missed by keyword-only methods. However, it may suffer from unstable recall due to approximation and is sensitive to parameters. Combining BM25 with HNSW merges stable lexical recall with semantic depth [3] [6].

Instability in Embedding Models

Beyond HNSW's approximate nature, embedding models themselves can introduce unpredictability:

- Different models or model versions may yield slightly varying embeddings for the same input.
- Embeddings retrained on evolving corpora can shift semantic landscapes, making results less stable over time.
- The complexity and **"black-box"** nature of embeddings mean small changes in vector space can disproportionately affect similarity rankings.

Such instability demands a hybrid and flexible approach.

Balancing BM25 and HNSW with Dynamic Weighting

We combine BM25 and HNSW scores into a weighted sum:

$$\text{Final Score} = w_{BM25} \cdot \text{BM25 Score}_{norm} + w_{HNSW} \cdot \text{HNSW Similarity}_{norm} \tag{12}$$

Weights w_{BM25} and w_{HNSW} are adjustable in real-time, allowing the system to respond to query characteristics, user preferences, or performance trends. For instance, increasing w_{BM25} stabilizes recall in uncertain conditions, while raising w_{HNSW} emphasizes semantic richness.

Extensibility with Multiple Embedding Models

This framework extends naturally to multiple embedding models:

$$\text{Final Score} = w_{BM25} \cdot \text{BM25 Score}_{norm} \ + \ \sum_{j=1}^{M} w_{HNSW}^{(j)} \cdot \text{HNSW Similarity}_{norm}^{(j)} \tag{13}$$

- Tailor retrieval to domains or languages by integrating specialized embeddings.
- Dynamically adjust weights for each model, leveraging multiple semantic perspectives.
- Scale and evolve the pipeline by adding or updating embedding models as needed.

Integrating Reciprocal Rank Fusion (RRF) for Score Aggregation

In addition to using weighted sums, we can consider **Reciprocal Rank Fusion (RRF)** [8] as an alternative or complementary method for combining ranked lists produced by BM25, HNSW, and multiple embedding models. RRF is a simple yet effective ensemble method often used in Information Retrieval (IR) research.

Reciprocal Rank Fusion Formula:

$$\text{RRFScore}(d) = \sum_{j=1}^{M} \frac{1}{k + \text{rank}^{(j)}(d)} \tag{14}$$

- M: Number of ranking methods.
- $\text{rank}^{(j)}(d)$: Rank position of document d by the j-th method.
- k: A small constant (e.g., 60 or 100) to dampen the influence of lower-ranked results.

Why RRF?

- Robustness to Score Variations: RRF relies on rank positions, mitigating the impact of disparate score scales or instability in individual methods.
- Simplicity and Effectiveness: RRF requires minimal tuning and often yields strong baseline fusion performance.
- Flexible Integration: RRF can be applied as a final step or combined with weighted scoring, allowing the system to experiment with different fusion strategies.

By incorporating RRF, we add a rank-based aggregation layer that can further enhance stability and fairness across multiple retrieval methods and embedding models.

Expected Benefits

1. **Enhanced Coherence and Focus**:
   Anchoring responses in the most recent turn and filtering irrelevant history fosters more natural, human-like interactions.
2. **Stable, Semantically Enriched Results**:
   Combining BM25's stable recall [7] with HNSW's semantic depth [5] counters the unpredictability of embedding-based retrieval [6], supported by adaptive weighting. Incorporating RRF further stabilizes final rankings by focusing on rank positions rather than raw scores.
3. **Adaptive and Extensible**:
   Real-time weight adjustments, the ability to integrate multiple embedding models, and the option to apply RRF for final fusion empower the system to adapt dynamically to changing data, user needs, and domain-specific vocabularies.
4. **Continuous Optimization**:
   Through iterative testing and monitoring, weight distributions, model combinations, and fusion strategies (weighted sums, RRF [8]) can be refined over time for better performance.

Conclusion

Our unified memory-based context reconstruction framework offers a solution to the pitfalls of cumulative context usage in LLM interactions. By always including the most recent memory [4], employing retrieval-based minimal context assembly [2] [3], blending BM25 [7] with HNSW [5], integrating multiple embedding models [6], and considering RRF [8] for final fusion, we create a robust, versatile, and semantically enriched retrieval environment.

This adaptive strategy enhances efficiency, accuracy, and adaptability in LLM-driven conversational systems, resonating with deeper philosophical themes. It provides a strong foundation for future research in retrieval-augmented generation [2] [3] and dynamic context management, capable of evolving alongside advancements in embeddings, ANN techniques, and natural language understanding [1].

References

1. Bender, E. M., & Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 5185–5198. ↩ ↩ ↩

2. Lewis, P., Oguz, B., Rinott, R., Riedel, S., & Stenetorp, P. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems (NeurIPS)*. ↩ ↩ ↩ ↩ ↩

3. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ↩ ↩ ↩ ↩ ↩

4. Sacks, H., Schegloff, E. A., & Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4), 696–735. ↩ ↩ ↩ ↩

5. Malkov, Y. A., & Yashunin, D. (2018). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10), 2567–2580. ↩ ↩ ↩ ↩

6. Ni, J., Flor, M., & Alfonseca, E. (2021). Sentence-level representations from multilingual BERT: A study of semantic equivalence tasks. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 471–481. ↩ ↩ ↩ ↩

7. Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 232–241). ↩ ↩ ↩ ↩

8. Cormack, G. V., Clarke, C. L., & Buettcher, S. (2009). Reciprocal Rank Fusion outperforms Condorcet and individual Rank Learning Methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 758–759. ↩ ↩ ↩