Segment-Graph Retrieval Memory: Extending Context Windows for Large Language Models with Graph-Structured Memory

Alakh Sharma

July 6, 2025

Abstract

Large Language Models (LLMs) are fundamentally limited by their fixed-size context windows, which restrict their ability to utilize long-term historical information. We propose Segment-Graph Retrieval Memory (SGRM), a modular, mathematically rigorous system that employs graph-based data structures to dynamically store, index, and retrieve context segments far exceeding the LLM's native input size. This paper details the mathematical formulation of every stage—chunking, embedding, graph construction, context retrieval, summarization, and generation—and includes a thorough theoretical sanity check of the approach.

1 Introduction

Modern LLMs, such as GPT-style transformers, possess a fixed input context window (e.g., 2,048 tokens), impeding applications that demand persistent long-term memory or knowledge retrieval. Our method seeks to break this barrier using data structures, enabling:

- Efficient chunking and embedding of arbitrarily long input streams.
- Graph-based storage and retrieval using both semantic similarity and chronology.
- Summarization and re-insertion of relevant historical information into the prompt.

The approach is modular, scalable, and amenable to extension with more sophisticated memory systems.

2 Pipeline Overview

Given an input text D, we perform the following steps:

(1) **Chunking**: $D \to \{c_1, \ldots, c_n\}$

(2) **Embedding**: $c_i \to \mathbf{e}_i \in \mathbb{R}^d$

(3) **Graph Construction**: $G = (V, E),\ V = \{v_i\}$

(4) **Query Embedding**: $q \to \mathbf{e}_q$

(5) **Context Retrieval**: $\mathcal{R}_q \subseteq V$

(6) **Summarization/Serialization**: $\mathcal{S}(\mathcal{R}_q)$

(7) **Prompt Generation & LLM Decoding**

3 Mathematical Formulation

3.1 Chunking

Given document D (tokenized into T tokens), split into $n = \lceil \frac{|T|}{C} \rceil$ chunks $c_1, \ldots, c_n$ of size at most C:

$$c_i = \text{decode}(T[(i-1)C + 1 : \min(iC, |T|)])$$

3.2 Embedding

Each chunk c_i is mapped via an embedding function f_{emb} (e.g., a sentence-transformer):

$$\mathbf{e}_i = f_{emb}(c_i), \quad \mathbf{e}_i \in \mathbb{R}^d$$

All embeddings stored in matrix $E \in \mathbb{R}^{n \times d}$.

3.3 Graph Construction

Vertices. Each chunk c_i is represented as node v_i.

Edges. Two edge types:

- *Chronological edges:* $(v_{i-1}, v_i),\ i = 2, \ldots, n.$
- *Similarity edges:* For each v_i, compute cosine similarity with all $j < i$:

$$s_{ij} = \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\|}$$

  and create directed edges (v_i, v_j) for the top-k highest s_{ij}.

Let $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$.

3.4 Query Embedding

Given a new query q (may be a question or new text), encode as $\mathbf{e}_q = f_{emb}(q)$.

3.5 Seed Node Selection

Find seed node v_s:

$$s = \arg\max_i \left(\frac{\mathbf{e}_q \cdot \mathbf{e}_i}{\|\mathbf{e}_q\| \|\mathbf{e}_i\|} \right)$$

3.6 Context Retrieval

From v_s, select a set $\mathcal{R}_q$ of nodes by traversing up to L shortest paths (BFS/DFS) or by similarity ranking:

$$\mathcal{R}_q = \text{Top}_m\{v_j \mid \text{reachable from } v_s\}$$

This set is then ordered (chronologically, or by similarity).

3.7 Summarization and Prompt Construction

Summarize or truncate retrieved texts:

$$\tilde{c}_j = f_{summ}(c_j),\ \forall v_j \in \mathcal{R}_q$$

The prompt for the LLM is:

$$P = [\tilde{c}_{j_1};\ \ldots;\ \tilde{c}_{j_m};\ q]$$

where ; denotes concatenation with clear separators.

3.8 LLM Generation

Let f_{llm} denote the LLM's decoding function:

$$y = f_{llm}(P)$$

where y is the generated continuation or answer.

4 Implementation Details

Chunking. We use HuggingFace tokenizers for robust, language-aware chunking.

Embedding. Sentence-Transformer models, such as `all-MiniLM-L6-v2`, are employed for efficient, semantically meaningful chunk embeddings.

Graph Management. NetworkX is used for a directed, dynamically growing graph structure. Similarity computations utilize NumPy or efficient libraries such as FAISS for scaling.

Summarization. HuggingFace summarization models (`facebook/bart-large-cnn`, `t5-base`) compress retrieved context to fit within the LLM's token window.

Generation. The prompt, composed of (summarized) context and query, is fed to an LLM (e.g., GPT-2 or larger models).

5 Mathematical and Theoretical Sanity Checks

A. Uniqueness and Coverage Every input token t is assigned to a single chunk c_i, and all c_i are processed—no information is dropped at this stage.

B. Embedding Representation Each c_i is mapped to $\mathbf{e}_i \in \mathbb{R}^d$:

- Embeddings must be sufficiently expressive; if d is too small, chunk similarity may be inaccurate.
- The use of pre-trained sentence transformers ensures semantic similarity is preserved.

C. Graph Connectivity

- The graph is always weakly connected (via chronological links).
- Similarity edges add shortcut connections; as k *uparrow*, retrieval recall increases but at the cost of more graph edges.

D. Retrieval Correctness

- Given a well-trained embedder and sufficient k, the set $\mathcal{R}_q$ will, in expectation, contain the most semantically relevant history to q.
- Limiting $|\mathcal{R}_q|$ controls the prompt size.

E. Computational Complexity

- Chunking:
  $mathcalO(n)$
- Embedding:
  $mathcalO(nd)$
- Similarity search:
  $mathcalO(nk)$ for each new chunk (brute force),
  $mathcalO(n$
  $logn)$ with approximate methods
- Retrieval (shortest-path):
  $mathcalO(|E| + |V|)$
- Summarization/LLM: bounded by model complexity and prompt length

F. Edge Cases

- If all chunk embeddings are nearly orthogonal, only chronological context is retrieved.
- If k is too low, long-range semantic retrieval fails; if too high, the graph becomes dense and costly to search.
- Overly aggressive summarization may discard important detail.

6 Experimental Protocol

We recommend the following protocol:

- Input: Use a long, multi-topic document (see Appendix for example).
- Query: Pose questions requiring both local and long-range context.
- Metrics:
  - Retrieval recall@k: Fraction of relevant chunks (by ground-truth) present in $\mathcal{R}_q$.
  - QA accuracy: LLM answer correctness with and without segment-graph augmentation.
- Ablation: Vary k, chunk size C, and summarizer.

7 Conclusion

The Segment-Graph Retrieval Memory (SGRM) framework provides an effective, scalable, and mathematically principled approach to extending the context of LLMs. Theoretically, the system guarantees retrieval coverage and semantic recall given a sufficiently expressive embedding space and a well-structured graph. Practically, this enables LLMs to operate on input scales far beyond their fixed context window, unlocking new real-world applications.

Appendix: Example Input

See the provided `input.txt` for a sample multi-section text on electric vehicles.