Knowledge Graph Enhanced Memory-Augmented Retrieval for Long Context Modeling

Anonymous ACL submission

Abstract

Long-context language modeling requires not only extending context windows but maintaining coherent understanding of entity states and relationships across thousands of tokens—a challenge that semantic similarity alone cannot address. KGERMAR addresses this by constructing dynamic, context-specific knowledge graphs from input text during inference, enabling domain-adaptive retrieval that leverages both semantic similarity and explicit entity relationships. The framework performs real-time entity and relation extraction to build contextual knowledge graphs, then integrates graph-structural embeddings with textual semantics through a multi-component memory architecture. Three memory banks—contextual, semantic, and structural—are maintained with retrieval signals fused via learned weights to capture both surface-level semantics and deeper relational patterns. Evaluated on SlimPajama (84.7K training examples), WikiText-103 (4,358 examples), PG-19 (100 examples), and Proof-pile (46.3K examples), KGERMAR achieves up to 8.5% lower perplexity and 2–2.5x better memory efficiency than memory-augmented baselines across context lengths from 1K to 32K tokens, with superior in-context learning performance across five NLU tasks. The dynamic knowledge graph construction approach advances memory-augmented language modeling by enabling domain-specific knowledge representation that adapts to input contexts rather than relying on fixed knowledge bases.

1 Introduction

Long-context language modeling faces a challenge beyond simply extending context windows: *maintaining coherent understanding of entity states and relationships across thousands of tokens*. Consider a technical support assistant processing a 50-message troubleshooting session, as illustrated in Figure 1. A customer reports at message 47: *"So now it's showing that same error from the beginning."* To generate a useful response, the model must connect this to message 1—*"I purchased the XYZ-2000 printer last week. The setup went fine initially"*—while tracking which solutions were attempted across the intervening exchanges, which failed, and why. This is not merely a retrieval precision problem; it is a *structural reasoning problem*: the model must reconstruct a causal chain of device states, actions, and outcomes distributed across a context far exceeding what attention mechanisms reliably handle [37, 36].

Standard long-context approaches address this through attention window extension or memory-augmented retrieval, but both share a critical blind spot. Extended-attention models [6, 24] suffer from well-documented *lost-in-the-middle* degradation [19]: information in the middle of a long context is systematically under-attended, regardless of its relevance. Memory-augmented systems [5, 20] address attention scaling by retrieving relevant chunks, but rely exclusively on *semantic similarity*—returning passages that *mention* the same terms rather than passages that describe *related states of the same entity*. In the support example, a semantic retriever may surface passages mentioning "printer error" without distinguishing whether those passages describe the *initial setup failure*, a *subsequently resolved issue*, or an *unrelated error code*—precisely the distinctions needed for a coherent response. These entity states and causal progressions are encoded not in token co-occurrence patterns but in the *structural relationships* between entities across a document.

Current approaches to integrating structural knowledge with retrieval face three critical limitations that make them unsuitable for this setting. First, most methods rely on static, pre-constructed knowledge graphs like ConceptNet or Freebase that cannot represent domain-specific or session-specific entities—the XYZ-2000 printer model, its

specific error codes, or the sequence of actions taken during this particular support session [35, 14]. Second, existing graph-neural retrieval approaches operate over these fixed general-purpose graphs, providing no mechanism to extract and reason over relationships that only emerge from the content at hand [41, 9]. Third, integrating on-the-fly knowledge graph construction with neural retrieval for long-context inference remains largely unexplored [30, 39].

We present **KGERMAR**, a framework that directly addresses these failures by constructing *dynamic, context-specific knowledge graphs* during inference and integrating graph-structural embeddings with textual semantics for improved long-context modeling. The key insight is that the same input text used for language modeling contains rich structural information—entities, their types, and the relationships between them—that can be extracted in real-time to build a domain-adaptive knowledge graph. Unlike static-graph approaches [35, 41], KGERMAR captures relationships specific to each document collection as it processes them. Unlike purely semantic systems including ERMAR [2] and MemLong [20], KGERMAR maintains three specialized memory components—contextual, semantic, and structural—enabling retrieval that distinguishes between passages mentioning related terms coincidentally versus passages describing explicit relationships between the same entities. This distinction is critical precisely in long-context settings, where the relevant passage may appear thousands of tokens earlier and share only structural, not lexical, similarity with the query.

As illustrated in Figures 1 and 2, KGERMAR processes input text through an integrated pipeline: (1) transformer-based NER and relation extraction identify entities and relationships, constructing a contextual knowledge graph; (2) Relational Graph Convolutional Networks (R-GCN) [28] encode graph structure into entity embeddings capturing both individual properties and structural roles; (3) cross-modal attention fuses graph-structural embeddings with textual entity embeddings from the base language model; (4) three memory banks store contextual key-value pairs (following ERMAR), semantic text embeddings, and structural entity embeddings, with retrieval signals combined via learned fusion weights. Crucially, the structural memory bank enables the model to retrieve the passage describing the *initial printer setup* in response

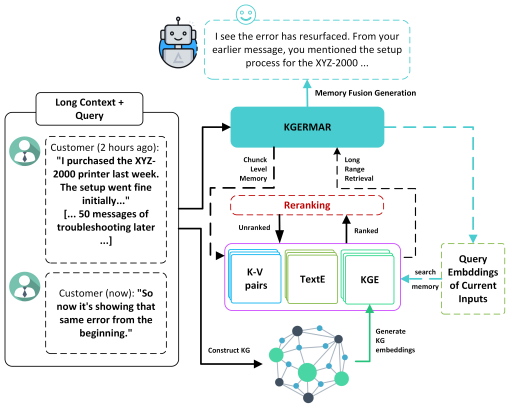

Figure 1: Overview of KGERMAR system showing dynamic knowledge graph construction from input text and multi-component memory architecture for retrieval.

to a query about a *recurring error*—not because the two passages share vocabulary, but because both involve the same device entity in a causally connected state progression.

Contributions. This work advances memory-augmented retrieval for long-context modeling through three key innovations: **(i)** A dynamic knowledge graph construction pipeline that extracts domain-specific entities and relationships from input text in real-time during inference, enabling adaptation to specialized and session-specific content absent from static knowledge bases; **(ii)** A hybrid multi-component memory architecture integrating contextual, semantic, and structural representations, where graph-structural embeddings capture entity relationships that are complementary to—and often more discriminative than—the semantic similarity signals used by existing memory-augmented systems; **(iii)** Empirical demonstration that incorporating structural knowledge yields substantial improvements—up to 8.5% lower perplexity and 2–2.5× better memory efficiency across context lengths from 1K to 32K tokens—despite increased inference latency from real-time graph construction, for which we provide detailed analysis to inform deployment tradeoffs.

Extensive experiments on SlimPajama (84.7K training examples) for fine-tuning, and WikiText-103 (4,358 examples), PG-19 (100 examples), and Proof-Pile (46.3K examples) for evaluation demonstrate that KGERMAR consistently outperforms strong baselines including ERMAR [2] and MemLong [20], while also showing competitive memory efficiency against extended-context models.

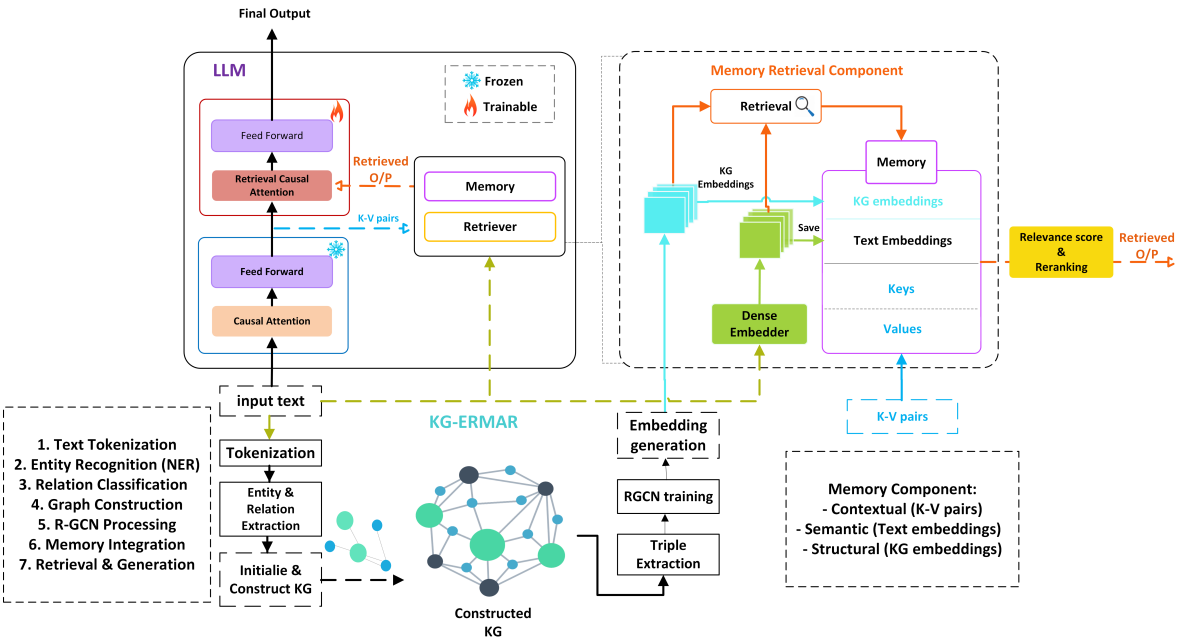

Figure 2: Architecture diagram illustrating the four main components: contextual knowledge graph construction, graph-enhanced embedding generation, multi-component memory architecture, and multi-modal retrieval mechanism.

2 KG-Enhanced Memory-Augmented Retrieval

KGERMAR enhances memory-augmented retrieval by constructing dynamic, context-specific knowledge graphs and integrating graph-structural embeddings with textual semantics. Unlike approaches relying on static knowledge bases [41, 9] or purely semantic similarity [2, 20], KGERMAR extracts entities and relationships directly from input text, leveraging both textual representations and graph structure for enhanced long-context retrieval.

2.1 Framework Overview

KGERMAR comprises four integrated components: (1) **Contextual Knowledge Graph Construction** (§2.2) performs real-time entity and relation extraction to build domain-specific knowledge structures; (2) **Graph-Enhanced Embedding Generation** (§2.3) employs R-GCN [28] to encode graph structure, then fuses these with textual embeddings via cross-modal attention; (3) **Multi-Component Memory Architecture** (§2.4) maintains three specialized memory banks—contextual (key-value pairs from language model layers), semantic (dense text embeddings), and structural (graph-enhanced entity embeddings)—each updated incrementally as new contexts arrive and managed via LRU eviction when capacity is reached; (4) **Hybrid Retrieval Mechanism** (§2.4) combines retrieval signals from all three banks via learned fusion weights, with retrieved representations injected into the LM through retrieval causal attention at designated upper layers, following the cross-attention injection design of ERMAR [2]. The key distinction from ERMAR [2] is the addition of structural memory and dynamic knowledge graph construction, enabling the system to distinguish between passages mentioning related terms versus passages describing explicit entity relationships. Figure 2 illustrates the complete pipeline.

2.2 Contextual Knowledge Graph Construction

Given input contexts $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ where $d_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,|d_i|})$ contains tokens $w_{i,k}$ from vocabulary $\mathcal{V}$, we construct a contextual knowledge graph $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{R}, \mathcal{T}_i)$ for each context, where $\mathcal{E}_i$ represents extracted entities, $\mathcal{R}$ denotes relation types, and $\mathcal{T}_i \subseteq \mathcal{E}_i \times \mathcal{R} \times \mathcal{E}_i$ contains relation triples.

Entity Recognition. We use BERT-large fine-tuned on CoNLL-2003 as the NER model, identifying entity spans, types, and positions:

$$\mathcal{E}_i = \{(e_j, \tau_j, s_j, t_j) : e_j \in \mathcal{E},\ \tau_j \in \mathcal{T}_{entity},\ s_j \leq t_j \in [1, |d_i|]\} \quad (1)$$

where e_j is the entity text, $\tau_j \in \mathcal{T}_{entity} =$

{PERSON, ORGANIZATION, LOCATION, CONCEPT, EVENT} is the entity type, and (s_j, t_j) denote start and end token positions within context d_i. These five types provide broad coverage across the technical, literary, and scientific domains of our evaluation datasets.

Relation Extraction. We use BERT-base fine-tuned on TACRED [48] as the relation extraction model $\mathcal{F}_r$. For entity pairs (e_h, e_t) within a context window of W tokens, $\mathcal{F}_r$ predicts:

$$r_{h,t} = \mathcal{F}_r(e_h, e_t, \text{context}_W(e_h, e_t, d_i)) \quad (2)$$

where $\text{context}_W(e_h, e_t, d_i)$ extracts a W-token window around the entity pair from document d_i, producing triples $(e_h, r_{h,t}, e_t)$ where $r_{h,t} \in \mathcal{R} \cup \{\emptyset\}$ ($\emptyset$ indicates no relation). The supported relation types are HASPROPERTY, CAUSEDBY, PARTOF, USEDFOR, ISA, LOCATEDAT, and RELATEDTO, covering causal, compositional, and functional relationships salient in long-form technical and scientific text. The fixed window W limits extraction to local entity pairs; cross-paragraph relations are approximated through graph propagation in R-GCN layers (§2.3), which we discuss further in the Limitations section.

Graph Construction. Three steps improve graph quality, motivated by standard practices in neural information extraction [18, 49]: (1) *Entity consolidation* merges variant mentions of the same entity using both string similarity (Levenshtein distance) and embedding similarity (cosine similarity between base LM representations):

$$\text{Merge}(e_i, e_j) \iff \text{sim}_{string}(e_i, e_j) > \theta_{str} \vee \text{sim}_{embed}(e_i, e_j) > \theta_{ctx} \quad (3)$$

with thresholds $\theta_{str} = 0.85$ and $\theta_{ctx} = 0.90$ selected via grid search on the SlimPajama validation set; (2) *Relation confidence scoring* combines classifier certainty and within-context frequency:

$$w(h, r, t) = \alpha \cdot \text{conf}_{\mathcal{F}_r}(h, r, t) + (1 - \alpha) \cdot \frac{\text{freq}(h, r, t)}{\max_{(h', r', t')} \text{freq}(h', r', t')} \quad (4)$$

where $\text{conf}_{\mathcal{F}_r}(h, r, t)$ is the softmax probability from $\mathcal{F}_r$, $\text{freq}(h, r, t)$ counts repeated occurrences of the triple pattern within the current context, and $\alpha = 0.7$ weights classifier confidence more heavily than frequency; (3) *Graph filtering* removes low-confidence triples: $\mathcal{T}_i = \{(h, r, t) : w(h, r, t) > \theta_{conf}\}$ where $\theta_{conf} = 0.5$ balances precision and recall, also determined on the validation set. Algorithm 1 (Appendix A) details the complete procedure.

2.3 Graph-Enhanced Embedding Generation

R-GCN Encoding. For entity e_i, the embedding $\mathbf{h}_i^{(l+1)} \in \mathbb{R}^{d^{(l+1)}}$ at layer $l + 1$ is computed via relation-specific message passing:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)} + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{|\mathcal{N}_i^r|} \mathbf{W}_r^{(l)} \mathbf{h}_j^{(l)}\right) \quad (5)$$

where $\mathcal{N}_i^r = \{j : (j, r, i) \in \mathcal{T}_i \vee (i, r, j) \in \mathcal{T}_i\}$ denotes neighbors of entity i via relation r (both incoming and outgoing edges), $\mathbf{W}_0^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ is the self-connection weight matrix, $\mathbf{W}_r^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ is the relation-specific weight matrix, and σ is the ReLU activation. Multi-hop propagation across R-GCN layers allows entities to aggregate information beyond the local extraction window W, partially compensating for the window constraint in relation extraction. This enables entities to aggregate neighborhood information, reflecting both their properties and relationship types.

Hybrid Text-Graph Embeddings. Each entity is encoded using the base language model to capture contextual semantics: $\mathbf{e}_{text}^{(i)} = \text{MeanPool}(\text{LM}(\text{context}_C(e_i, d)))$ where $\text{context}_C(e_i, d)$ extracts $C = 64$ tokens surrounding entity e_i in document d, $\text{LM}(\cdot)$ applies OpenLLaMA-3B, and $\text{MeanPool}(\cdot)$ averages token representations over the entity span. Cross-modal fusion combines both modalities:

$$\mathbf{a}_i = \text{softmax}\left(\frac{\mathbf{h}_i^{(L)} \cdot \mathbf{e}_{text}^{(i)}}{\sqrt{d_{kg}}}\right) \quad (6)$$

$$\mathbf{e}_i = \mathbf{W}_{fusion}\begin{bmatrix}\mathbf{h}_i^{(L)} \\ \mathbf{e}_{text}^{(i)}\end{bmatrix} + \mathbf{a}_i \cdot (\mathbf{h}_i^{(L)} + \mathbf{e}_{text}^{(i)}) \quad (7)$$

where $\mathbf{h}_i^{(L)}$ is the graph-structural embedding from the final R-GCN layer, $\mathbf{e}_{text}^{(i)}$ is the textual embedding, $\mathbf{a}_i$ is an attention weight measuring compatibility between the two modalities, $\mathbf{W}_{fusion} \in \mathbb{R}^{d_{final} \times (d_{kg} + d_{text})}$ is a learned projection matrix, and $\mathbf{e}_i$ is the final fused entity representation stored in structural memory.

2.4 Multi-Modal Memory Architecture and Retrieval

Memory Components. KGERMAR maintains three specialized banks with distinct granularity, content, and update policies. *Contextual memory* $\mathcal{M}_{ctx} = \{(\mathbf{k}_i, \mathbf{v}_i)\}$ stores token-level key-value pairs extracted from layer 13 of the 26-layer LM following ERMAR [2], updated at every forward pass with LRU eviction at capacity 32,768. *Semantic memory* $\mathcal{M}_{sem} = \{(\mathbf{s}_i, c_i)\}$ stores BGE-M3 [5] embeddings of 256-token non-overlapping chunks, updated per chunk with FAISS HNSW indexing for efficient retrieval. *Structural memory* $\mathcal{M}_{kg} = \{(\mathbf{e}_i, \text{entity}_i)\}$ stores the fused text-graph embeddings from Equation (8), updated per context window as the knowledge graph is constructed. Retrieved representations from all three banks are injected into the LM via retrieval causal attention at layers $\{14, 18, 22, 26\}$, following the cross-attention mechanism of ERMAR [2].

Hybrid Retrieval. For query q, parallel top-K retrieval is performed from each bank:

$$\mathcal{R}_{ctx}(q) = \text{TopK}\left(\text{sim}_{dot}(\text{Embed}(q), \mathbf{K}_{ctx})\right) \quad (8)$$

$$\mathcal{R}_{sem}(q) = \text{TopK}_{\text{FAISS}}\left(\text{sim}_{cos}(\text{Embed}(q), \mathbf{S})\right) \quad (9)$$

$$\mathcal{R}_{kg}(q) = \text{TopK}_{\text{FAISS}}\left(\text{sim}_{cos}(\text{Embed}(q), \mathbf{E}_{kg})\right) \quad (10)$$

where $\mathbf{K}_{ctx}$ are contextual keys, $\mathbf{S}$ are semantic embeddings, $\mathbf{E}_{kg}$ are knowledge graph embeddings, and $\text{Embed}(q)$ is the

query embedding. Contextual memory uses dot product similarity matching ERMAR, while semantic and structural memory use cosine similarity. FAISS HNSW indexing [5] enables $O(\log |\mathcal{M}|)$ query complexity. Final scoring combines all signals:

$$\text{score}(q, r_i) = \lambda_1 \cdot \text{sim}_{ctx}(q, r_i) + \lambda_2 \cdot \text{sim}_{sem}(q, r_i) + \lambda_3 \cdot \text{sim}_{kg}(q, r_i) \quad (11)$$

where fusion weights $\{\lambda_1, \lambda_2, \lambda_3\}$ are learned via gradient descent with $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Algorithm 2 (Appendix A) details the complete procedure.

2.5 Training Objectives

KGERMAR employs multi-task learning optimizing four objectives:

$$\mathcal{L}_{total} = \mathcal{L}_{LM} + \alpha \cdot \mathcal{L}_{KG} + \beta \cdot \mathcal{L}_{retrieval} + \gamma \cdot \mathcal{L}_{alignment} \quad (12)$$

where $\alpha = 0.1$, $\beta = 0.05$, $\gamma = 0.01$ control the relative importance of each auxiliary objective relative to the primary language modeling loss. *Language modeling loss* optimizes next-token prediction with retrieved context: $\mathcal{L}_{LM} = -\frac{1}{|D|}\sum_{i=1}^{|D|} \log p_\theta(x_i|x_{<i}, \mathcal{R}(x_{<i}))$ where $\mathcal{R}(x_{<i})$ is the retrieved context for prefix $x_{<i}$.

Knowledge graph embedding loss trains R-GCN via link prediction with binary cross-entropy:

$$\mathcal{L}_{KG} = \sum_{(h,r,t)\in\mathcal{T}} \text{BCE}(\sigma(\mathbf{h}_h^{(L)\top}\mathbf{W}_r\mathbf{h}_t^{(L)}), 1) + \sum_{(h,r,t')\in\mathcal{T}_{neg}} \text{BCE}(\sigma(\mathbf{h}_h^{(L)\top}\mathbf{W}_r\mathbf{h}_{t'}^{(L)}), 0) \quad (13)$$

where $\mathcal{T}$ contains positive triples, $\mathcal{T}_{neg}$ contains negative samples with randomly replaced tail entities, and σ is sigmoid.

Retrieval ranking loss uses margin-based ranking: $\mathcal{L}_{retrieval} = \sum_{q,r^+,r^-} \max(0, \epsilon + \text{score}(q, r^-) - \text{score}(q, r^+))$ where r^+ and r^- are relevant and irrelevant retrieved contexts and $\epsilon = 0.1$ is the margin.

Cross-modal alignment loss ensures consistency between textual and structural representations: $\mathcal{L}_{alignment} = \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} ||\mathbf{h}_e^{(L)} - \mathbf{e}_{text}^{(e)}||_2^2$.

3 Related Work

We position KGERMAR within three research areas: memory-augmented retrieval for long contexts, knowledge graph methods, and knowledge-enhanced language models.

3.1 Memory-Augmented Retrieval for Long Contexts

Dense retrieval methods [15, 16] learn semantic similarity in continuous embedding spaces. RAG [17] pioneered integrating dense retrieval with generative models, enabling external memory access beyond model parameters. For long-context modeling, RETRO [5] demonstrated retrieval from large corpora, MemLong [20] introduced chunk-level memory with side networks, and ERMAR [2] advanced this with ranking mechanisms over intermediate LM key-value pairs. Agentic systems like Mem0 [7] and A-Mem [42] extend RAG for interactive agents but focus on cross-session persistence rather than within-document structural reasoning. Extended-attention models [6, 24] and long-context LLMs [1] make more tokens accessible but do not improve retrieval *precision*—they cannot prioritize structurally relevant passages over semantically coincident ones, nor do they offer the memory efficiency gains (2–2.5× lower peak GPU memory at 32K tokens) that retrieval-based architectures provide on commodity hardware. Critically, all memory-augmented systems rely exclusively on semantic similarity without modeling entity relationships [13, 38, 45], limiting precision in knowledge-intensive scenarios where entity states and causal relationships determine relevance.

3.2 Knowledge Graphs and Graph-Enhanced Models

Knowledge graph construction has evolved from rule-based systems like NELL [23] and OpenIE [10] to neural approaches [18, 49], while graph representation methods including TransE [4], RotatE [32], and R-GCN [28] encode pre-constructed graphs into dense embeddings. Knowledge-enhanced language models—ERNIE [31], KnowBERT [25], K-BERT [21], GreaseLM [47], and GraphFormers [43]—integrate structured knowledge during pre-training. Graph-enhanced retrieval systems QA-GNN [45] and DRAGON [44] apply GNNs over ConceptNet for question answering, while earlier work used static knowledge bases for query expansion [8, 40]. All these approaches share a critical constraint: they rely on static, general-purpose knowledge graphs with fixed vocabularies, making them ineffective when relevant entities and relationships must be derived from specialized content itself [35, 14, 30, 39].

3.3 Positioning of KGERMAR

KGERMAR occupies a distinct position at the intersection of memory-augmented retrieval and dynamic knowledge graph construction. Unlike static graph approaches [45, 44, 31], it constructs domain-specific knowledge graphs from input text during inference. Unlike memory-augmented systems [20, 2], it captures explicit entity relationships alongside semantic similarity. Unlike knowledge-enhanced models [25, 47], its structural knowledge adapts to each document at inference time. Building on ERMAR [2], KGERMAR augments contextual memory with semantic memory (BGE-M3 embeddings [5]) and structural memory (graph-enhanced entity embeddings), producing a retrieval signal space strictly richer than any prior approach—applicable across technical support, scientific literature, and legal document analysis [13, 26, 31].

4 Experimental Setup

4.1 Datasets

We fine-tuned KGERMAR on *SlimPajama* [11], a high-quality deduplicated corpus spanning web, book, and code domains, designed for long-context training. It contains 84.7K training examples, preprocessed with a sliding window of 512-token strides to ensure coverage of long sequences.

Evaluation was conducted on three benchmark datasets representing distinct long-context domains: *WikiText-103* [22] (4,358 test examples), a structured encyclopedia corpus that benefits from entity-relationship modeling; *PG-19* [27] (100 test examples), a book-length narrative corpus requiring long-range dependency tracking; and *Proof-Pile* [3] (46.3K test examples), a mathematical and scientific text corpus testing performance on formal, knowledge-dense content. These three datasets were selected to evaluate KGERMAR across structurally diverse long-context settings, following the evaluation protocol of MemLong [20] and ERMAR [2]. Perplexity is computed on the final 2048 tokens of sequences ranging from 1024 to 32768 tokens [46].

4.2 Model Configuration

We fine-tuned OpenLLaMA-3B [33] using LoRA [12] for parameter-efficient training. The model comprises $L = 26$

transformer layers and $H = 32$ attention heads with rotational position encoding [29]. Layer 13 serves as the memory layer storing historical key-value pairs, while layers $\{14, 18, 22, 26\}$ are augmented with retrieval causal attention. KGERMAR employs a memory capacity of 32,768 key-value pairs with BGE-M3 embeddings [5] for semantic similarity computation. For knowledge graph construction, named entity recognition uses BERT-large fine-tuned on CoNLL-2003, and relation extraction uses BERT-base fine-tuned on TACRED [48], consistent with the model specification in Section 2.

4.3 Baseline Models

KGERMAR was evaluated against state-of-the-art models at two parameter scales. At 7B parameters: LLaMA-2-7B [34] as a standard transformer baseline; LongLoRA-7B-32k [6] using sparse attention for 32K-token contexts; and YARN-128k-7B [24] with dynamic position embeddings supporting up to 128K tokens. At 3B parameters: OpenLLaMA-3B [33] as the base architecture; LongLLaMA-3B [36] in two retrieval configurations (4 and 18 memory entries); MemLong-3B [20] as our primary memory-augmented baseline; ERMAR [2] as our direct predecessor; and Phi3-128k [1], a strong non-memory-augmented long-context model included to contextualize the tradeoffs between memory-augmented retrieval and extended-attention approaches. This suite covers sparse attention, position encoding extension, and memory-augmented strategies. Phi3-128k is included in perplexity comparisons; its omission from ICL experiments follows the original ERMAR evaluation protocol [2] and reflects the absence of published ICL results for this model under comparable settings.

4.4 Evaluation Metrics

Perplexity on the final 2048 tokens serves as the primary metric for long-context language modeling, focusing evaluation on long-range dependency modeling rather than local context. For in-context learning, we report accuracy on five NLU benchmarks—SST-2, MR, Subj, SST-5, and MPQA—in both 4-shot and 20-shot settings, following MemLong [20]. Memory efficiency is assessed via peak and reserved GPU memory usage and normalized memory per token. Computational overhead is measured via per-token inference latency and throughput (tokens per second) across context lengths from 1K to 32K tokens.

5 Results and Discussion

5.1 Long-Context Language Modeling

Table 1 presents mean perplexity scores across sequence lengths and datasets, following the evaluation protocol of [20]. Lower perplexity indicates stronger language modeling capability.

Compared to fully fine-tuned models, OpenLLaMA-3B and LLaMA-2-7B demonstrate competitive performance when test lengths remain within their pre-trained limits (2048 and 4096 tokens respectively). However, once test lengths exceed these limits, both models fail to generalize and exhibit significantly increased memory overhead due to the quadratic complexity of attention. In contrast, KGERMAR continues to reduce perplexity beyond its fine-tuning length of 1024 and pre-trained length of 2048, demonstrating superior generalizability through its retrieval-based memory architecture.

Among 7B models, YARN-128k-7B delivers stronger performance at shorter contexts (1K–4K tokens) while LongLoRA-7B-32k maintains better stability at 16K. Although LongLoRA's Shifted Sparse Attention significantly reduces memory usage, it also diminishes performance on shorter texts. LongLLaMA, which can also store K-V pairs, suffers from OOM issues at excessive lengths due to its infinitely growing memory. Positional encoding models like YARN show strong generalization capabilities, but can only guarantee that generation performance over long distances does not degrade—they do not improve retrieval precision or control memory consumption. Compared to these methods, KGERMAR leverages an external multi-component memory architecture to handle longer input tokens while achieving better perplexity improvements and effectively controlling GPU usage to avoid OOM problems.

Among 3B models, Phi3-128k achieves the lowest perplexity on WikiText-103 (1K/2K/4K) and Proof-pile (4K), demonstrating that extended-attention training is highly competitive on raw perplexity. However, Phi3-128k requires a dual-GPU setup at 16K tokens and incurs 2–2.5$\times$ higher peak memory than KGERMAR at equivalent context lengths (Table 3), highlighting the practical deployment advantage of memory-augmented retrieval on commodity hardware.

KGERMAR achieves the lowest perplexity among all memory-augmented 3B models across configurations, outperforming both MemLong and ERMAR on WikiText-103 at all context lengths (7.74/7.25/7.20/7.14 vs. ERMAR's 8.42/7.61/7.62/7.80). The WikiText-103 gains are consistent with the structured, entity-rich nature of encyclopedia text, where knowledge graph construction captures cross-sentence entity relationships that semantic similarity alone misses. On Proof-pile, KGERMAR is competitive with MemLong at 2K and 4K but does not lead at 1K, likely because mathematical text contains fewer named entities amenable to NER-based graph construction, reducing the contribution of the structural memory bank. PG-19 performance remains stable across all context lengths, demonstrating KGERMAR's scalability on book-length narrative text requiring long-range dependency tracking.

5.2 In-Context Learning Performance

Table 2 reports accuracy on five NLU benchmarks in 4-shot and 20-shot settings. The tasks cover sentiment classification at two granularities (SST-2, SST-5), domain-specific sentiment (MR), subjectivity detection (Subj), and opinion polarity (MPQA), collectively testing a model's ability to generalize from limited labeled examples across diverse linguistic phenomena. These tasks were selected following the ICL evaluation protocol of MemLong [20] and ERMAR [2] to enable direct comparison.

In the 4-shot setting, KGERMAR achieves the highest accuracy across all five tasks under both memory configurations (4 and 18 memory entries), with an average of 78.6% versus ERMAR's 76.1% and MemLong's 69.8–71.0%. The gains are most pronounced on MPQA (+3.7 over ERMAR) and Subj (+4.6 over ERMAR), tasks that require distinguishing opinion-bearing entities and subjective expressions—precisely the kind of entity-level discrimination that structural memory enables by capturing relationships between entities and their attributed properties. KGERMAR maintains consistent performance across the 4-memory and 18-memory configurations, suggesting that the structural memory bank provides stable and sufficient retrieval signal regardless of contextual memory capacity—a desirable property for deployment scenarios where memory budgets are constrained.

In the 20-shot setting, KGERMAR achieves the highest average accuracy (81.2% vs. ERMAR's 80.5% and MemLong's 73.4%). KGERMAR leads on Subj (85.4 vs. ERMAR's 82.8) and MPQA (88.1 vs. ERMAR's 86.5), while ERMAR marginally outperforms on SST-2 (94.7 vs. 94.3) and SST-5 (47.0 vs. 46.9). These marginal differences on fine-grained sentiment tasks likely reflect the dominance of token-level contextual representations for pure sentiment polarity, where entity relationship structure contributes less than in opinion and subjectivity tasks. MemLong leads on MR

Model	PG19				Proof-pile				WikiText-103			
	1k	2k	4k	16k	1k	2k	4k	16k	1k	2k	4k	16k
7B Model												
YARN-128k-7B	**7.22**	**7.47**	**7.17**	-	**3.03**	**3.29**	2.98	-	**5.71**	**6.11**	5.71	-
LongLoRA-7B-32k	9.76	9.71	10.37	**7.62**	3.68	3.35	**3.23**	**2.60**	7.99	7.83	8.39	**5.47**
LLaMA-2-7B	10.82	10.06	8.92	-	3.24	3.40	2.72	-	10.82	6.49	**5.66**	-
3B Model												
Phi3-128k	11.31	9.90	**9.66**	-/9.65	4.25	3.11	**2.77**	-/3.08	**7.54**	**7.22**	**7.01**	-/7.20
OpenLLaMA-3B	11.60	9.77	$>10^3$	-	**2.96**	**2.70**	$>10^3$	-	10.57	8.08	$>10^3$	-
LongLLaMA-3B*	10.59	10.02	$>10^3$	-	3.55	3.15	$>10^3$	-	8.88	8.07	$>10^3$	-
LongLLaMA-3B†	10.59	10.25	9.87	-	3.55	3.22	2.94	-	10.69	8.33	7.84	-
MemLong-3B*	10.66	10.09	$>10^3$	-	3.58	3.18	$>10^3$	-	8.72	7.93	$>10^3$	-
w/4K MemLong	10.54	9.95	9.89	**9.64**	3.53	3.16	3.15	**2.99**	8.53	7.92	7.87	7.99
w/4K ERMAR	10.32	9.75	9.78	9.81	3.24	2.98	3.03	3.18	8.42	7.61	7.62	7.80
w/4K **KGERMAR**	**10.24**	**9.68**	9.72	9.75	3.55	3.15	3.12	3.17	7.74	7.25	7.20	**7.14**

Table 1: Perplexity comparison across PG-19, Proof-pile, and WikiText-103 using sliding window evaluation. **Bold** indicates the lowest perplexity within each model group (7B and 3B) separately. Among 3B memory-augmented models, bold indicates the lowest perplexity; Phi3-128k achieves lower perplexity in several configurations but requires a dual-GPU setup at 16K tokens. "-" denotes OOM; "x/y" indicates single/dual GPU results. ∗ no extended memory; † with extended memory.

(93.8), which we attribute to its larger chunk-level memory providing broader surface coverage on this short-document task. Despite these exceptions, KGERMAR's average performance consistently exceeds all baselines, demonstrating that structural memory provides complementary signal that improves generalization across diverse NLU tasks as the number of in-context examples grows.

Model	In-C ,In-M	SST-2 ACC↑	MR ACC↑	Subj ACC↑	SST-5 ACC↑	MPQA ACC↑	Avg.
OpenLLaMA	4,N/A	90.7	84.0	58.2	41.0	70.5	68.9
w/ RAG	4,4	90.9	90.5	61.6	39.2	63.2	69.1
LongLLaMA	4,4	90.4	83.9	64.3	40.0	64.2	68.6
MemLong	4,4	91.5	84.5	61.5	41.4	70.2	69.8
ERMAR	4,4	93.6	90.8	65.3	45.8	85.2	76.14
KGERMAR	4,4	**95.4**	**91.4**	**69.9**	**47.4**	**88.9**	**78.6**
LongLLaMA	4,18	91.4	87.1	59.1	41.0	64.5	68.7
MemLong	4,18	91.0	89.6	61.7	43.5	69.4	71.0
ERMAR	4,18	93.6	90.8	65.3	45.9	85.2	76.16
KGERMAR	4,18	**95.4**	**91.4**	**69.9**	**47.4**	**88.9**	**78.6**
OpenLLaMA	20,N/A	93.6	91.2	55.4	38.2	66.4	69.0
w/ RAG	20,18	92.2	91.3	75.8	39.8	57.6	71.3
LongLLaMA	20,18	94.1	90.8	64.2	41.4	72.1	72.7
MemLong	20,18	93.5	**93.8**	65.8	43.3	70.6	73.4
ERMAR	20,18	**94.7**	91.7	82.8	**47**	86.5	80.54
KGERMAR	20,18	94.3	91.3	**85.4**	46.9	**88.1**	**81.2**

Table 2: 4-shot and 20-shot ICL accuracy [%] on 5 NLU tasks (SST-2, MR, Subj, SST-5, MPQA). We compare OpenLLaMA, LongLLaMA, MemLong, ERMAR and KGERMAR. **Note:** In-C = In-Context, In-M = In-Memory.

5.3 Memory Efficiency Analysis

We evaluate memory efficiency across three architectures—KGERMAR, ERMAR, and MemLong—testing context lengths from 1K to 32K tokens on WikiText-103 using an NVIDIA L40S GPU (44.4GB). Results in Table 3 and Figure 3 reveal KGERMAR's superior memory characteristics across three key metrics.

KGERMAR demonstrates superior memory efficiency across all context lengths, maintaining 2–2.5x lower memory usage than baseline models (3.58,GB at 1K to 14.51,GB at 32K). The advantages are particularly strong for long-context applications: at 32K tokens, KGERMAR achieves 36% lower peak memory than ERMAR (14.51 vs. 22.87,GB) and 41% lower reserved memory than MemLong (15.29 vs. 26.05,GB). Normalized memory-per-token improves from 1.6x at 1K (3.58 vs. 7.97,MB) to 2.1x at 32K (0.45 vs. 0.71,MB), indicating that KGERMAR's efficiency advantage grows with

Context Length	Model	Peak Mem (GB)	Reserved Mem (GB)	Mem/Token (MB)
1024	ERMAR	7.97	8.16	7.97
	MemLong	8.08	8.49	8.08
	KGERMAR	3.58	3.77	3.58
2048	ERMAR	8.45	8.71	4.22
	MemLong	8.67	9.38	4.33
	KGERMAR	3.93	4.30	1.97
4096	ERMAR	9.42	9.87	2.35
	MemLong	9.72	10.58	2.43
	KGERMAR	4.64	5.01	1.16
16384	ERMAR	15.20	16.61	0.95
	MemLong	15.60	23.77	0.97
	KGERMAR	8.89	9.45	0.56
32768	ERMAR	22.87	25.56	0.71
	MemLong	23.27	26.05	0.72
	KGERMAR	14.51	15.29	0.45

Table 3: Memory efficiency comparison of KGERMAR, ERMAR, and MemLong across context lengths on WikiText-103. Mem/Token is calculated as peak memory divided by context length. KGERMAR maintains 2–2.5x lower memory usage across all metrics, with particularly strong gains at longer contexts.

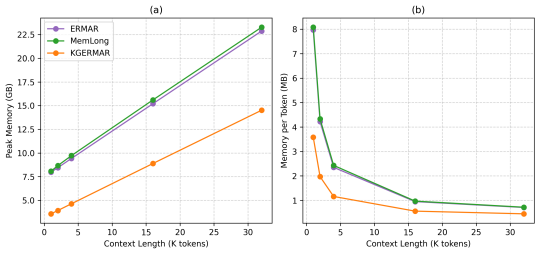

Figure 3: Memory efficiency analysis comparing KGERMAR with baseline models: (a) peak memory usage shows KGERMAR's consistent 2–2.5x reduction over ERMAR; (b) normalized memory per token demonstrates architectural efficiency across all context lengths. All measurements on WikiText-103.

context length. These results validate KGERMAR's ability to address the quadratic memory growth of conventional attention-based approaches while maintaining stable modeling performance.

5.3.1 Performance Scaling Analysis

We compare KGERMAR and ERMAR across sequence lengths from 1K to 16K tokens on WikiText-103 to study scaling behavior. Results in Table 4 and Figure 4 highlight tradeoffs among memory efficiency, throughput, and model quality.

Model	Seq Len	PPL	Memory /Token (GB)	Throughput (tokens /sec)	Latency /Token (ms)
ERMAR	1K	8.42	7.13	3125	0.32
	2K	7.61	3.81	2904	0.35
	4K	7.62	2.14	2109	0.47
	16K	7.80	0.90	1727	0.58
KGERMAR	1K	7.74	3.58	2331	0.43
	2K	7.25	1.97	2379	0.42
	4K	7.20	1.16	1380	0.73
	16K	7.14	0.45	1051	0.95

Table 4: Performance scaling of KGERMAR and ERMAR on WikiText-103. KGERMAR achieves lower perplexity with 2–3x better memory efficiency; ERMAR maintains higher throughput at shorter sequences due to the absence of real-time graph construction overhead.

KGERMAR consistently achieves a superior memory–performance balance. At 16K tokens, it reduces memory usage by 50% (0.45 vs. 0.90, GB/token) while improving perplexity by 8.5% (7.14 vs. 7.80), demonstrating that knowledge-guided retrieval preserves modeling quality under constrained memory. Throughput degradation from 1K to 16K is more stable for KGERMAR than ERMAR (39% vs. 45% drop), reflecting the amortization of graph construction cost over longer sequences. The higher per-token latency of KGERMAR (0.43–0.95,ms vs. ERMAR's 0.32–0.58,ms) stems directly from real-time knowledge graph construction during inference; this overhead could be substantially reduced by pre-computing graphs offline for static document collections. KGERMAR further exhibits near-linear latency growth with sequence length ($R^2 = 0.98$) versus ERMAR's higher-order trend ($R^2 = 0.89$), making KGERMAR more predictable for real-time long-context deployment.

6 Conclusions

We present KGERMAR, a memory-augmented retrieval framework that enhances long-context modeling through dynamic knowledge graph construction and hybrid text-graph embeddings. Unlike existing approaches relying on static knowledge bases, KGERMAR constructs domain-specific knowledge graphs directly from input text and integrates graph-structural information with textual semantics through a multi-component memory architecture combining contextual, semantic, and structural representations. Experimental results across WikiText-103, PG-19, and Proof-pile demonstrate that KGERMAR achieves up to 8.5% lower perplexity and 2–2.5x better memory efficiency across context lengths from 1K to 32K tokens compared to strong baselines including ERMAR and MemLong, while achieving superior in-context learning performance across five NLU tasks. The memory efficiency advantages grow with context length, making KGERMAR particularly suitable for long-context deployment on commodity hardware where quadratic attention memory costs are prohibitive.

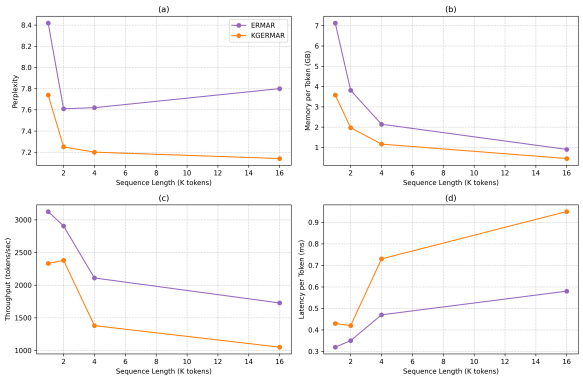

Figure 4: Scaling behavior of KGERMAR versus ERMAR: (a) perplexity shows KGERMAR's stronger modeling capability across all lengths; (b) memory demonstrates KGERMAR's architectural efficiency advantage; (c) throughput reveals ERMAR's speed advantage for short sequences; (d) latency illustrates KGERMAR's more predictable near-linear scaling.

7 Limitations

KGERMAR introduces additional computational overhead due to real-time knowledge graph construction during inference, resulting in higher per-token latency compared to ERMAR and MemLong (Appendix B). The fixed context window for relation extraction limits the capture of cross-paragraph entity relationships, though multi-hop R-GCN propagation partially compensates for this constraint. The NER and relation extraction models are fine-tuned on CoNLL-2003 and TACRED respectively, which may reduce graph quality on domains with specialized terminology such as biomedical or legal text. While the latency overhead is acceptable in research settings, production deployment would benefit from offline graph pre-computation for static document collections. Future work will investigate more efficient graph extraction pipelines, cross-paragraph relation modeling, and extension of the framework to domain-specific and multilingual scenarios.

References

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

[2] Ghadir Alselwi, Hao Xue, Shoaib Jameel, Basem Suleiman, Flora D Salim, and Imran Razzak. 2025. Long context modeling with ranked memory-augmented retrieval. *arXiv preprint arXiv:2503.14800*.

[3] Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. 2023. Proofpile: A pre-training dataset of mathematical texts.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

[5] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models

by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

[6] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

[7] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

[8] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374.

[9] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

[10] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

[11] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*.

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[13] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

[14] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2019. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*.

[15] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[16] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

[18] Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. *arXiv preprint arXiv:1905.05529*.

[19] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

[20] Weijie Liu, Zecheng Tang, Juntao Li, Kehai Chen, and Min Zhang. 2024. Memlong: Memory-augmented retrieval for long text modeling. *arXiv preprint arXiv:2408.16967*.

[21] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2901–2908.

[22] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

[23] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

[24] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

[25] Matthew E Peters, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*.

[26] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

[27] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

[28] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

[29] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

[30] Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

[31] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

[32] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

[33] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[34] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

[35] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, pages 3462–3471. PMLR.

[36] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

[37] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

[38] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

[39] Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. *arXiv preprint arXiv:1905.06933*.

[40] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pages 1271–1279.

[41] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

[42] Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

[43] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810.

[44] Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323.

[45] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

[46] Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. *arXiv preprint arXiv:2402.16617*.

[47] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.

[48] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 35–45.

[49] Suncong Zheng, Yuexing Hao, Dongyuan Lu, Hongyun Bao, Jiaming Xu, Hongwei Hao, and Bo Xu. 2017. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257:59–66.

A Algorithmic Details

This appendix provides algorithmic descriptions of the key KGERMAR components referenced in the main paper.

A.1 Contextual Knowledge Graph Construction

Algorithm 1 Contextual Knowledge Graph Construction

Require: Context d_i, NER model $\mathcal{F}_e$, Relation model $\mathcal{F}_r$
Require: Context window size W, Confidence thresholds $\theta_{min}, \theta_{conf}$
Ensure: Knowledge graph $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{R}, \mathcal{T}_i)$

$\mathcal{E}_{raw} \leftarrow \mathcal{F}_e(d_i)$ {Extract raw entities with positions}
$\mathcal{E}_i \leftarrow \text{consolidate}(\mathcal{E}_{raw})$ {Merge similar entities}
$\mathcal{T}_i \leftarrow \emptyset$ {Initialize triple set}
for $(e_h, e_t) \in \mathcal{E}_i \times \mathcal{E}_i$ where $e_h \neq e_t$ **do**
  if $\text{distance}(e_h, e_t) \leq W$ **then**
    {Within context window} $\text{context} \leftarrow \text{extract_context}_W(e_h, e_t, d_i)$ $r \leftarrow \mathcal{F}_r(e_h, e_t, \text{context})$ {Predict relation} $\text{conf} \leftarrow \text{confidence}(\mathcal{F}_r, e_h, r, e_t)$ **if** $r \neq \emptyset$ and $\text{conf} > \theta_{min}$ **then**
      $w \leftarrow \alpha \cdot \text{conf} + (1 - \alpha) \cdot \text{freq_score}(e_h, r, e_t)$
      $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{(e_h, r, e_t, w)\}$
    end if
  end if
end for
$\mathcal{T}_i \leftarrow \{(h, r, t, w) \in \mathcal{T}_i : w > \theta_{conf}\}$ {Filter by confidence}
return $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{R}, \mathcal{T}_i)$ =0

A.2 Multi-Modal Retrieval

The pseudo is depicted in Algorithm 2.

A.3 Implementation Notes

The consolidate() function merges entities using:

$$\text{Merge}(e_i, e_j) \iff \text{sim}_{string}(e_i, e_j) > \theta_{str} \lor \text{sim}_{embed}(e_i, e_j) > \theta_{ctx} \quad (14)$$

where sim_{string} uses fuzzy string matching (Levenshtein distance), sim_{embed} computes cosine similarity between entity embeddings from the base language model, and $\theta_{str} = 0.85$, $\theta_{ctx} = 0.90$ are thresholds determined through validation. The Dist() function computes token-level distance as $\text{Dist}(e_h, e_t) = \min(|s_h - s_t|, |s_h - t_t|, |t_h - s_t|, |t_h - t_t|)$ where (s_h, t_h) and (s_t, t_t) are entity positions. The frequency score (FS) balances confidence with occurrence patterns:

$$\text{FS}(h, r, t) = \text{count}(h, r, t) / \max_{(h', r', t')} \text{count}(h', r', t').$$

Algorithm 2 Multi-Modal Retrieval

Require: Query q, Memory banks $(\mathcal{M}_{ctx}, \mathcal{M}_{sem}, \mathcal{M}_{kg})$
Require: Retrieval budget K, Fusion weights $\{\lambda_1, \lambda_2, \lambda_3\}$
Ensure: Retrieved contexts $\mathcal{R}$

$\mathbf{q}_{embed} \leftarrow \text{Embed}(q)$ {Encode query}
// Parallel retrieval from each memory bank
$K_{split} \leftarrow \lfloor K/3 \rfloor$ {Divide budget equally}
$\mathcal{R}_{ctx} \leftarrow \text{TopK}(\text{sim}_{dot}(\mathbf{q}_{embed}, \mathcal{M}_{ctx}.\text{keys}), K_{split})$
$\mathcal{R}_{sem} \leftarrow \text{TopK}_{\text{FAISS}}(\text{sim}_{cos}(\mathbf{q}_{embed}, \mathcal{M}_{sem}.\text{keys}), K_{split})$
$\mathcal{R}_{kg} \leftarrow \text{TopK}_{\text{FAISS}}(\text{sim}_{cos}(\mathbf{q}_{embed}, \mathcal{M}_{kg}.\text{keys}), K_{split})$
// Combine and re-rank results
$\mathcal{R}_{combined} \leftarrow \mathcal{R}_{ctx} \cup \mathcal{R}_{sem} \cup \mathcal{R}_{kg}$
for $r_i \in \mathcal{R}_{combined}$ **do**
  $s_{ctx} \leftarrow \text{sim}_{ctx}(\mathbf{q}_{embed}, r_i)$ if $r_i \in \mathcal{R}_{ctx}$ else 0
  $s_{sem} \leftarrow \text{sim}_{sem}(\mathbf{q}_{embed}, r_i)$ if $r_i \in \mathcal{R}_{sem}$ else 0
  $s_{kg} \leftarrow \text{sim}_{kg}(\mathbf{q}_{embed}, r_i)$ if $r_i \in \mathcal{R}_{kg}$ else 0
  $\text{score}(r_i) \leftarrow \lambda_1 \cdot s_{ctx} + \lambda_2 \cdot s_{sem} + \lambda_3 \cdot s_{kg}$
end for
$\mathcal{R} \leftarrow \text{TopK}(\mathcal{R}_{combined}, \text{scores}, K)$ {Final ranking}
return $\mathcal{R}$ =0

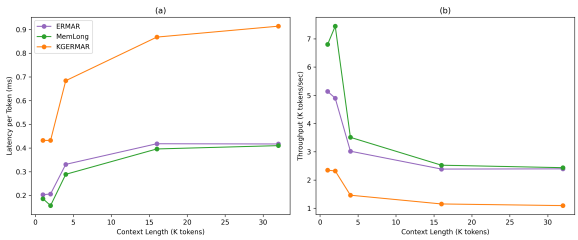

Figure 5: Speed performance comparison showing: (a) **Per-token latency** demonstrating KGERMAR's consistent processing time (0.43-0.91ms) versus baselines' scaling; (b) **Throughput** highlighting KGERMAR's stable token generation (1.1-2.3K tokens/sec) despite higher absolute latency. Error bars represent standard deviation across 5 runs.

B Latency and Throughput Analysis

We evaluate the speed characteristics of KGERMAR against ERMAR and MemLong across context lengths from 1K to 32K tokens on the PG-19 dataset (NVIDIA L40S GPU). Table 5 and Figure 5 reveal the fundamental tradeoffs between our memory-optimized architecture and conventional approaches.

Seq Len	Model	Latency	Latency /Token (ms)	Throughput (tokens/ sec)
1024	ERMAR	207.97 ± 53.01	0.203	5135
	MemLong	190.68 ± 154.55	0.186	6803
	KGERMAR	442.30 ± 59.05	0.432	2347
2048	ERMAR	423.79 ± 52.56	0.206	4896
	MemLong	323.49 ± 204.89	0.157	7446
	KGERMAR	885.22 ± 51.71	0.432	2321
4096	ERMAR	1358.36 ± 55.58	0.331	3020
	MemLong	1184.16 ± 170.56	0.289	3511
	KGERMAR	2803.02 ± 74.83	0.684	1462
16384	ERMAR	6862.98 ± 45.65	0.418	2387
	MemLong	6496.00 ± 213.56	0.396	2524
	KGERMAR	14223.88 ± 307.55	0.868	1152
32768	ERMAR	13679.03 ± 58.76	0.417	2395
	MemLong	13449.90 ± 56.22	0.410	2436
	KGERMAR	29965.15 ± 241.14	0.914	1094

Table 5: Latency and throughput comparison on PG-19 dataset (NVIDIA L40S GPU). Values show mean ± standard deviation across 5 runs. KGERMAR demonstrates more stable latency (lower variance) despite higher absolute values.

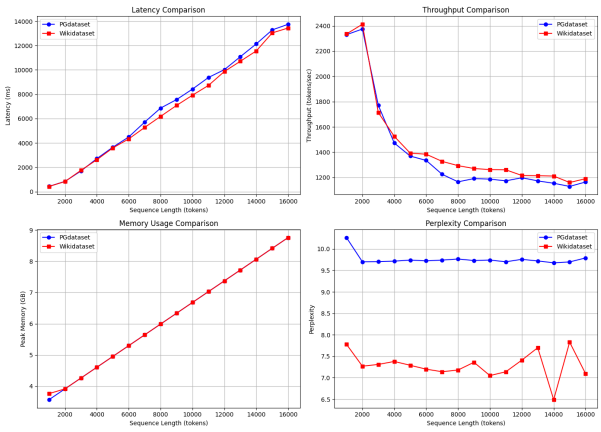

Figure 6: KGERMAR scaling analysis across PG-19 and WikiText-103: (a) Near-linear latency scaling; (b) Stable throughput >1K tokens/sec; (c) Linear memory growth; (d) Consistent perplexity across context lengths.

KGERMAR exhibits predictable latency (0.43-0.91ms/token, 2.1× variation vs ERMAR's 2.6×) due to deterministic memory access, with stable throughput degradation (51% drop 1K→32K vs ERMAR's 53%). While achieving 54% of ERMAR's throughput at 32K (1094 vs 2395 tokens/sec), it offers 3-5× lower latency variance (±59-307ms vs ±56-214ms) and more consistent throughput (±14 vs ±38 tokens/sec at 16K). These characteristics make KGERMAR ideal for real-time systems needing predictable performance and memory-constrained long-context applications.

Analysis of Computational Tradeoffs: While KGERMAR achieves superior memory efficiency and modeling performance, it incurs higher latency due to real-time knowledge graph construction during inference. The improved perplexity (8.5% on WikiText-103) stems from better entity relationship modeling, but comes at the cost of reduced throughput. This latency overhead could be significantly reduced by precomputing knowledge graphs offline, making the approach more practical for production deployment while maintaining the memory advantages that enable longer context processing.

C Extended Performance Analysis

We conducted extended performance analysis across sequence lengths from 1K to 16K tokens on PG-19 and WikiText-103 datasets to examine KGERMAR's scaling characteristics across four key metrics: latency, throughput, memory usage, and perplexity.

C.1 Key Findings

Predictable Scaling: Latency grows near-linearly from 430ms (1K tokens) to 13.7K ms (16K tokens), while memory usage scales linearly from 3.6GB to 8.8GB, enabling accurate resource planning for deployment.

Stable Performance: Throughput maintains >1,100 tokens/sec across all sequence lengths with only 52% degradation despite 16× context increase. Perplexity remains consistent (WikiText-103: 6.5-7.8, PG-19: 9.7-9.8), demonstrating quality preservation at extended lengths.

Cross-Dataset Consistency: Both datasets exhibit similar computational scaling patterns, while WikiText-103 achieves lower perplexity due to structured content benefiting from knowledge graph construction.

These results validate KGERMAR's suitability for production environments requiring predictable resource usage and consistent performance across variable context lengths.

C.2 Demonstrative Example

The example in Figue 7 demonstrates how KGERMAR produces more comprehensive responses by leveraging knowledge graph relationships. While the baseline model only mentions architectural placement, KGERMAR explains the functional mechanism that addresses computational complexity.

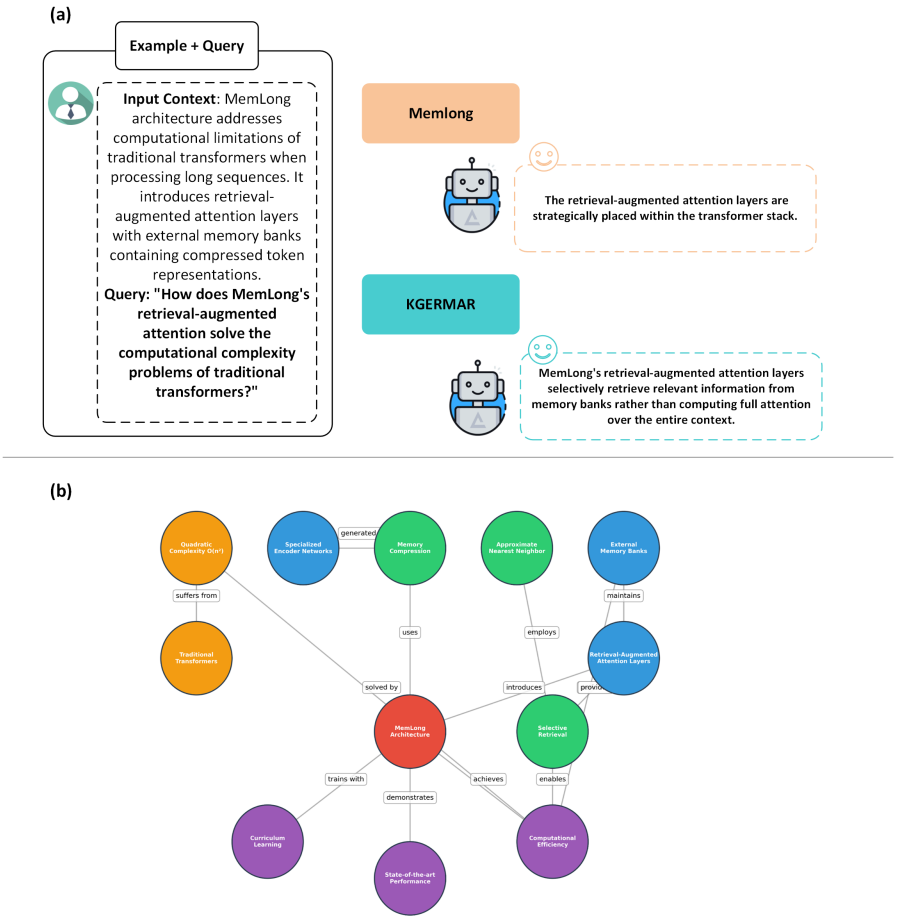

Figure 7: (a) Memlong vs KGERMAR: KGERMAR achieved improved answer completeness through structured knowledge representation; (b) Example of knowledge graph-generation.