Knowledge Graph Enhanced Memory-Augmented Retrieval for Long Context Modeling

Anonymous ACL submission

Abstract

Memory-augmented retrieval systems excel at semantic matching but fail to capture structural relationships between entities in long-form text—relationships critical for knowledge-intensive applications. KG-ERMAR addresses this limitation 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 specialized 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), KG-ERMAR achieves up to 8.5% lower perplexity and 2–2.5$\times$ better memory efficiency than strong baselines across context lengths from 1K to 32K tokens. 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

Knowledge extraction from long-form documents presents a fundamental challenge: traditional information retrieval methods excel at matching query keywords to semantically similar passages, yet fail to capture structural relationships between entities that often determine relevance. Consider a developer querying a technical knowledge base about "transformer attention mechanisms causing memory overflow." Semantic retrieval might return passages mentioning "attention" and "memory," but miss crucial relationships: *which transformer architectures* exhibit this behavior, *which hyperparameters* influence it, and *which mitigation strategies* address it. These entity relationships—not captured by semantic similarity alone—are essential for effective retrieval in knowledge-intensive domains including technical support, legal document analysis, and scientific literature retrieval, where understanding entity relationships within extended contexts is crucial [13, 25, 30].

Current approaches to integrating knowledge graphs with retrieval systems face three critical limitations. First, most methods rely on static, pre-constructed knowledge graphs like ConceptNet or Freebase that cannot adapt to domain-specific entities emerging from specialized content [34, 14]. Second, existing graph-neural approaches typically operate on these general-purpose knowledge bases, limiting applicability to specialized domains where relevant structures must be extracted from the content itself [40, 9]. Third, integrating contextual knowledge graph construction with neural retrieval mechanisms remains largely unexplored, particularly for efficient processing of extended textual content [29, 38].

Recent memory-augmented retrieval systems have demonstrated strong performance on long-context tasks by maintaining external memory banks of key-value pairs or text chunks [5, 19]. ERMAR [2] achieves competitive results through sophisticated ranking mechanisms and contextual key-value representations from intermediate language model layers. However, ERMAR—like other memory-augmented approaches—treats text as token sequences, relying exclusively on semantic similarity [36, 35]. This misses opportunities to exploit explicit entity relationships: when retrieving context about "attention mechanisms causing memory overflow," ERMAR cannot distinguish between passages mentioning these terms coinci-

dentally versus passages describing the *causal relationship* between specific attention patterns and memory consumption. The absence of structural knowledge representation limits retrieval precision in knowledge-intensive scenarios [37, 44].

We present **KG-ERMAR**, a framework that enhances memory-augmented retrieval by constructing *dynamic, context-specific knowledge graphs* during inference and integrating graph-structural embeddings with textual semantics for improved long-context modeling. Unlike approaches relying on static knowledge bases [34, 40], KG-ERMAR performs real-time entity extraction and relation identification from input text, building domain-adaptive knowledge graphs specific to each document collection. Unlike purely semantic retrieval systems including ERMAR [2], KG-ERMAR maintains three specialized memory components—contextual, semantic, and structural—enabling retrieval based on both semantic similarity and explicit entity relationships.

As illustrated in Figures 1 and 2, KG-ERMAR 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) [27] 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. The system updates memory incrementally as new contexts arrive.

Contributions. This article 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, enabling adaptation to specialized vocabularies 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 complementary to 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. We provide detailed analysis of this latency-performance tradeoff to inform practical deployment strategies.

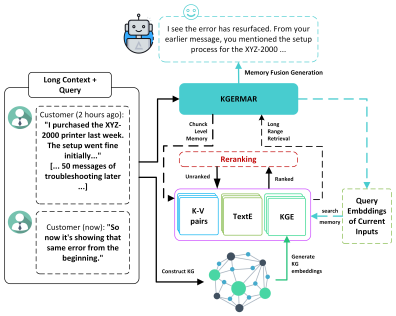

Figure 1: Overview of KG-ERMAR system showing dynamic knowledge graph construction from input text and multi-component memory architecture for retrieval.

Extensive experiments on benchmark datasets including 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 KG-ERMAR consistently outperforms strong baselines including ERMAR [2] and MemLong [19]. Our approach shows particular strength in long-context scenarios, making it valuable for applications requiring both efficiency and quality in extended document processing. We provide comprehensive ablation studies quantifying the contribution of each memory component and robustness analysis examining sensitivity to knowledge graph extraction quality.

2 KG-Enhanced Memory-Augmented Retrieval

KG-ERMAR 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 [40, 9] or purely semantic similarity [2, 19], KG-ERMAR extracts entities and relationships directly from input text, leveraging both textual representations and graph structure for enhanced long-context retrieval.

2.1 Framework Overview

KG-ERMAR comprises four integrated components: (1) **Contextual Knowledge Graph Construction** (§2.2) performs real-time entity and re-

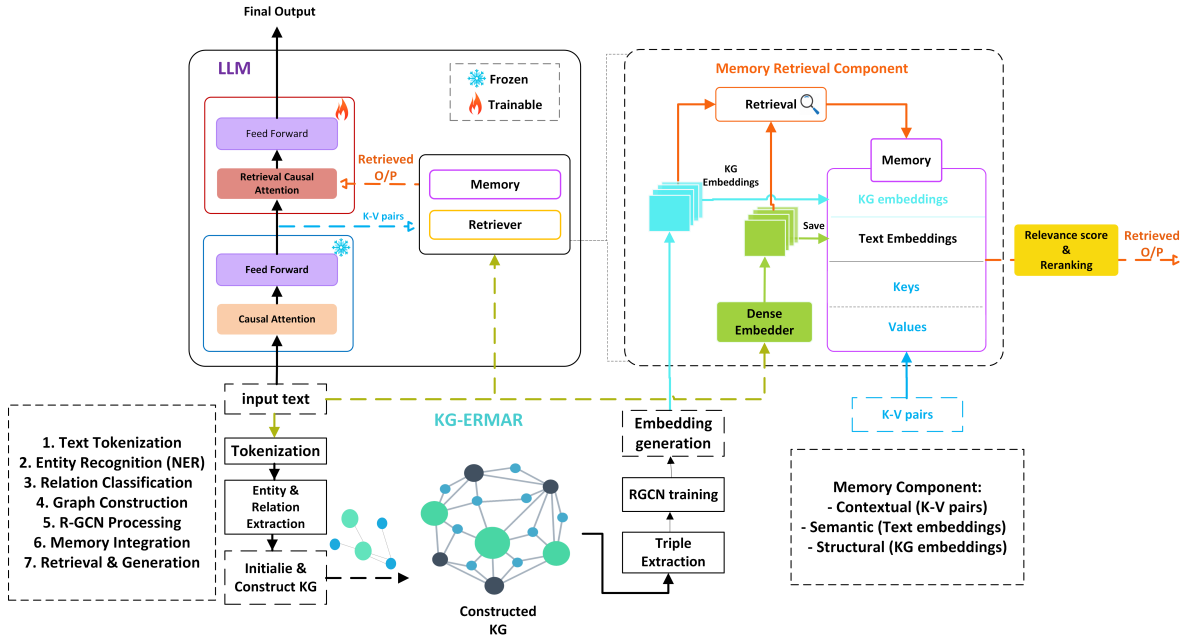

Figure 2: Architecture diagram illustrating the four main components: contextual knowledge graph construction, graph-enhanced embedding generation, multi-component memory architecture, and multi-modal retrieval mechanism.

lation extraction to build domain-specific knowledge structures; (2) **Graph-Enhanced Embedding Generation** (§2.3) employs R-GCN [27] 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); (4) **Hybrid Retrieval Mechanism** (§2.4) combines retrieval signals from all three banks via learned fusion weights.

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 $d_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,|d_i|})$, 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. A transformer-based NER model identifies 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, $\tau_j \in \mathcal{T}_{entity} = \{\text{PERSON}, \text{ORGANIZATION}, \text{LOCATION}, \text{CONCEPT}, \text{EVENT}\}$ is the entity type, and (s_j, t_j) denote start and end token positions within context d_i.

Relation Extraction. For entity pairs (e_h, e_t) within window W tokens, a pre-trained relation extraction model $\mathcal{F}_r$ 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 $r_{h,t} \in \mathcal{R} \cup \{\emptyset\}$ represents the predicted relationship ($\emptyset$ indicates no relation).

Graph Construction. Three steps improve graph quality: (1) *Entity consolidation* merges similar entities using string and embedding similarity ($\theta_{str} = 0.85$, $\theta_{ctx} = 0.90$); (2) *Relation confidence scoring* combines classifier certainty and frequency evidence:

$$w(h, r, t) = \alpha \cdot \text{conf}_{\mathcal{F}_r}(h, r, t) + (1 - \alpha) \cdot \frac{\text{freq}(h, r, t)}{\max_{(h', r', t')} \text{freq}(h', r', t')} \quad (3)$$

where $w(h,r,t)$ is the confidence score, $\text{conf}_{\mathcal{F}_r}(h,r,t)$ is the softmax probability from the relation extraction model, $\text{freq}(h,r,t)$ represents the frequency of this triple pattern within the current context, and $\alpha = 0.7$ weights classifier confidence over frequency; (3) *Graph filtering* removes low-confidence triples: $\mathcal{T}_i = \{(h,r,t) : w(h,r,t) > \theta_{conf}\}$ where $\theta_{conf} = 0.5$. 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:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)} + \sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{|\mathcal{N}_i^r|}\mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)}\right) \quad (4)$$

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. 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 the base language model, and $\text{MeanPool}(\cdot)$ averages token representations. Cross-modal fusion combines modalities:

$$\mathbf{a}_i = \text{softmax}\left(\frac{\mathbf{h}_i^{(L)}\cdot\mathbf{e}_{text}^{(i)}}{\sqrt{d_{kg}}}\right) \quad (5)$$

$$\mathbf{e}_i = \mathbf{W}_{fusion}\begin{bmatrix}\mathbf{h}_i^{(L)}\\ \mathbf{e}_{text}^{(i)}\end{bmatrix} + \mathbf{a}_i\cdot(\mathbf{h}_i^{(L)} + \mathbf{e}_{text}^{(i)}) \quad (6)$$

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, $\mathbf{W}_{fusion} \in \mathbb{R}^{d_{final}\times(d_{kg}+d_{text})}$ is a learned projection matrix, and $\mathbf{e}_i$ is the final fused entity representation.

2.4 Multi-Modal Memory Architecture and Retrieval

Memory Components. KG-ERMAR maintains three specialized banks: *Contextual memory* $\mathcal{M}_{ctx} = \{(\mathbf{k}_i, \mathbf{v}_i)\}$ stores key-value pairs from intermediate language model layers (layer 13 in our 26-layer model) following ERMAR [2]; *Semantic memory* $\mathcal{M}_{sem} = \{(\mathbf{s}_i, c_i)\}$ stores BGE-M3 embeddings [5] for 256-token chunks; *Structural memory* $\mathcal{M}_{kg} = \{(\mathbf{e}_i, \text{entity}_i)\}$ stores fused text-graph embeddings.

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

$$\mathcal{R}_{ctx}(q) = \text{TopK}\left(\text{sim}_{dot}(\text{Embed}(q), \mathbf{K}_{ctx})\right) \quad (7)$$

$$\mathcal{R}_{sem}(q) = \text{TopK}_{\text{FAISS}}\left(\text{sim}_{cos}(\text{Embed}(q), \mathbf{S})\right) \quad (8)$$

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

where $\mathbf{K}_{ctx}$ represents the contextual keys, $\mathbf{S}$ are the semantic embeddings, $\mathbf{E}_{kg}$ are the knowledge graph embeddings, and $\text{Embed}(q)$ is the query embedding. Contextual memory uses dot product similarity, while semantic and structural memory use cosine similarity. FAISS HNSW indexing [5] enables $O(\log|\mathcal{M}|)$ query complexity. Final scoring combines signals:

$$\begin{aligned}\text{score}(q,r_i) = \lambda_1\cdot\text{sim}_{ctx}(q,r_i) + \lambda_2\cdot\text{sim}_{sem}(q,r_i)\\ +\lambda_3\cdot\text{sim}_{kg}(q,r_i)\end{aligned} \quad (10)$$

where sim_{ctx}, sim_{sem}, and sim_{kg} are the similarity scores from each memory bank, and 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

KG-ERMAR employs multi-task learning optimizing four objectives:

$$\mathcal{L}_{total} = \mathcal{L}_{LM} + \alpha\cdot\mathcal{L}_{KG} + \beta\cdot\mathcal{L}_{retrieval} + \gamma\cdot\mathcal{L}_{alignment} \quad (11)$$

where $\alpha = 0.1$, $\beta = 0.05$, $\gamma = 0.01$ control the relative importance of each auxiliary objective.

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 $\mathcal{R}(x_{<i})$ represents the retrieved context for prefix $x_{<i}$.

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

$$\begin{aligned}\mathcal{L}_{KG} = &\sum_{(h,r,t)\in\mathcal{T}}\text{BCE}(\sigma(\mathbf{h}_h^{(L)\top}\mathbf{W}_r\mathbf{h}_t^{(L)}), 1)\\ &+\sum_{(h,r,t')\in\mathcal{T}_{neg}}\text{BCE}(\sigma(\mathbf{h}_h^{(L)\top}\mathbf{W}_r\mathbf{h}_{t'}^{(L)}), 0)\end{aligned} \quad (12)$$

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

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 positive (relevant) and negative (irrelevant) retrieved contexts for query q, and $\epsilon = 0.1$ is the margin parameter.

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

3 Related Work

We position KG-ERMAR 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 [19] introduced chunk-level memory operations with side networks, and ERMAR [2] advanced this with sophisticated ranking mechanisms that extract key-value pairs from intermediate language model layers. Recent agentic memory systems like Mem0 [7] and A-Mem [41] extend RAG with hierarchical architectures for interactive agents.

Limitation: Existing memory-augmented systems rely exclusively on semantic similarity without explicitly modeling entity relationships or structural patterns. KG-ERMAR extends ERMAR's contextual memory with structural memory from dynamically constructed knowledge graphs, enabling retrieval based on both semantic similarity and entity relationships.

3.2 Knowledge Graphs and Graph-Enhanced Models

Knowledge graph construction has evolved from rule-based systems like NELL [22] and OpenIE [10] to neural approaches [18, 47], while graph representation methods including TransE [4], RotatE [31], and R-GCN [27] excel at encoding pre-constructed graphs. However, these methods target static knowledge bases with fixed vocabularies rather than dynamic, context-specific construction.

Knowledge-enhanced language models integrate structured knowledge into pre-trained models. ERNIE [30], KnowBERT [24], and K-BERT [20] incorporate entity embeddings during pre-training, while GreaseLM [46] and GraphFormers [42] enable joint modeling of language and graph structure. Graph-enhanced retrieval systems like QA-GNN [44] and DRAGON [43] apply GNNs to question answering over ConceptNet, while earlier work employed static knowledge bases for query expansion [8, 39].

Limitation: These approaches integrate static knowledge graphs during pre-training or rely on general-purpose knowledge bases that lack domain-specific entities. KG-ERMAR dynamically constructs context-specific knowledge graphs during inference, enabling adaptation to specialized domains and evolving knowledge structures.

3.3 Positioning of KG-ERMAR

KG-ERMAR addresses three key limitations: (1) Unlike static knowledge graph approaches [44, 43, 30], KG-ERMAR constructs domain-specific knowledge graphs dynamically from input text; (2) Unlike memory-augmented systems [19, 2] relying solely on semantic similarity, KG-ERMAR maintains structural memory capturing entity relationships; (3) Unlike knowledge-enhanced models [24, 46] integrating static knowledge during pre-training, KG-ERMAR constructs contextual knowledge graphs during inference.

Building on ERMAR [2], KG-ERMAR augments contextual memory with semantic memory (dense text embeddings) and structural memory (graph-enhanced entity embeddings from dynamically constructed knowledge graphs). This multi-component architecture achieves superior performance while maintaining computational efficiency through incremental graph construction and FAISS-based indexing.

4 Experimental Setup

4.1 Datasets

We fine-tuned KGERMAR on the *SlimPajama* dataset [11], a high-quality, deduplicated corpus designed for long-context tasks. It contains 84.7K training rows, making it a compact yet effective resource for pre-training and fine-tuning. The dataset was preprocessed with a sliding window approach

using 512-token strides to ensure comprehensive coverage of long sequences.

Performance evaluation was conducted on three benchmark datasets: *WikiText-103* [21] (4,358 test rows), *PG-19* [26] (100 test rows), and *Proof-Pile* [3] (46.3K test rows). Performance was measured across context lengths from 1024 to 32768 tokens, using perplexity on the last 2048 tokens [45] following standard evaluation protocols.

4.2 Model Configuration

We fine-tuned OpenLLaMA-3B using LoRA [12] for parameter-efficient training. The model consists of $L = 26$ transformer layers, $H = 32$ attention heads, and uses rotational position encoding [28]. Layer 13 serves as the memory layer for storing historical context, while layers $[14, 18, 22, 26]$ are augmented with retrieval mechanisms. KG-ERMAR employs a memory capacity of 32,768 key-value pairs with BGE-M3 embeddings for semantic similarity computation. For knowledge graph construction, we use BERT-large fine-tuned on CoNLL-2003 for named entity recognition.

4.3 Baseline Models

ERMAR was evaluated against state-of-the-art models across two parameter scales to ensure a comprehensive comparison. The 7B models include LLaMA-2-7B [33] as a standard transformer baseline, LongLoRA-7B-32k [6], which employs sparse attention mechanisms for 32k-token contexts, and YARN-128k-7B [23] featuring dynamic position embeddings that support up to 128k tokens.

For the 3B parameter scale, we compared against OpenLLaMA-3B [32] as the base architecture, LongLLaMA-3B [35] evaluated in two retrieval configurations (4 and 18 memory entries), MemLong-3B [19] as our direct baseline with chunk-level memory operations, and Phi3-128k [1], which demonstrates strong performance across varying context lengths. This diverse benchmark suite encompasses different long-context strategies, including sparse attention, position encoding extensions, and memory-augmented architectures, ensuring robust evaluation of KGERMAR's retrieval-based approach against complementary methodologies.

4.4 Evaluation Metrics

We employ perplexity as the primary metric for language modeling performance, computed on the final 2048 tokens of each sequence to focus on long-range dependency modeling. For in-context learning tasks, we report accuracy on five natural language understanding benchmarks: SST-2, MR, Subj, SST-5, and MPQA, evaluated in both 4-shot and 20-shot settings. Memory efficiency is assessed through peak GPU memory usage and tokens processed per second, while computational overhead is measured via inference latency across different context lengths.

5 Results and Discussion

5.1 Long-Context Language Modeling

Following the experimental strategy adopted in [19], Table 1 presents the mean perplexity scores of our model across different sequence lengths and datasets, demonstrating its effectiveness in long-context modeling. Evaluation was performed on test splits of three datasets: *WikiText-103* [21] (4,358 rows), *PG-19* [26] (100 rows), and *Proof-Pile* [3] (46.3k rows).

The 7B parameter models reveal distinct scaling characteristics - YARN-128k-7B delivers stronger performance in shorter contexts (1k-4k tokens), while LongLoRA-7B-32k maintains better stability up to 16k tokens despite minor quality degradation. These differences underscore the importance of matching model architecture to specific context length requirements in practical applications.

The 3B models demonstrate KGERMAR's significant advantages in long-context tasks. While OpenLLaMA-3B struggles beyond 4k tokens, and Phi3-128k shows more consistent performance, KGERMAR sets new performance benchmarks for Wikitext and PG19 datasets. KGERMAR outperforms MemLong and larger 7B models in several configurations, including achieving a remarkable perplexity values on WikiText-103, surpassing MemLong's and ERMAR's. Additionally, KGERMAR maintains stable performance for PG19 dataset demonstrating its superior scalability in long-context modelling. Despite slight underperformance in specific Proof-pile configurations, it is still outperforming Memlong in 2k and 4k, and ERMAR in 16K.

5.2 In-Context Learning Performance

The results in Table 2 show KGERMAR's strong performance across five natural language understanding tasks in both 4-shot and 20-shot settings.

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 of 7B and 3B models across PG19, Proof-pile, and WikiText-103, using a sliding window evaluation. "-" denotes Out of Memory (OOM) errors, and "x/y" indicates results from single/dual GPU setups. Memory-augmented models are tested with varying capacities.

In the 4-shot setting, KGERMAR achieves state-of-the-art results across all tasks, outperforming OpenLLaMA and other memory-augmented models. It excels even in challenging tasks like SST-5 and MPQA, maintaining high performance with limited examples. Its stability across different memory configurations highlights its robustness in low-resource scenarios.

KGERMAR continues to excel in the 20-shot scenario, achieving top results in tasks like MPQA and Subj. While it lags behind MemLong in the MR dataset and behind ERMAR in SST-2 and SST-5 by 0.4, its average performance overall outperforms it, showcasing its scalability with increased examples.

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.

KGERMAR consistently performs well across varying context lengths, effectively leveraging memory augmentation. Its ability to scale with more examples and handle both short and long-range dependencies makes it a strong candidate for general-purpose language modeling, advancing the state-of-the-art in language understanding tasks.

5.3 Memory Efficiency Analysis

We conduct a comprehensive evaluation of memory efficiency across three architectures—KGERMAR, ERMAR, and MemLong— testing context lengths from 1K to 32K tokens on Wikitext data using an NVIDIA L40S GPU (44.4GB). The results in Table 3 and Figure 3 reveal KGERMAR's superior memory characteristics through three key metrics.

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 data. Mem/Token is calculated as Peak Mem divided by context length. KGERMAR maintains 2–2.5× lower memory usage across all metrics, with particularly strong gains at longer contexts.

KGERMAR demonstrates superior memory efficiency across all context lengths, maintaining 2-2.5× lower memory usage than baseline models (3.58GB at 1K to 14.51GB at 32K). The architecture shows particularly strong advantages for long-context applications, achieving 36% lower peak memory than ERMAR at 32K contexts (14.51 vs 22.87GB) and 58% reduced reserved memory versus MemLong at 16K. Normalized effi-

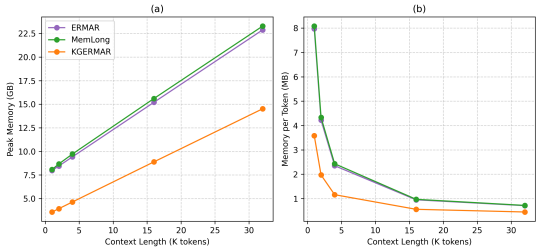

Figure 3: Memory efficiency analysis comparing KGERMAR with baseline models: (a) Peak memory usage shows KGERMAR's consistent 2–2.5× reduction over ERMAR; (b) Normalized memory per token demonstrates architectural efficiency; All measurements on Wikitext data.

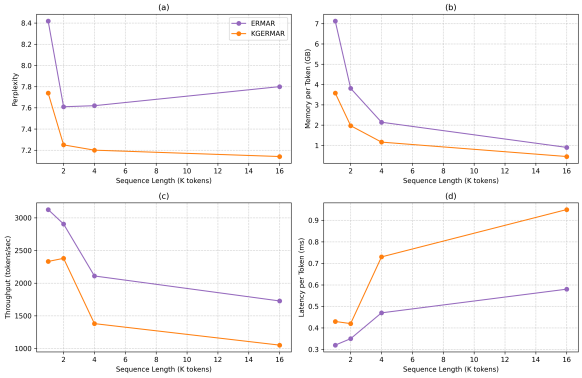

Figure 4: Scaling behavior of KGERMAR versus ERMAR: (a) **Perplexity** shows KGERMAR's stronger modeling capability across all lengths; (b) **Memory** demonstrates KGERMAR's architectural advantage in memory efficiency; (c) **Throughput** reveals ERMAR's speed advantage for short sequences; (d) **Latency** illustrates KGERMAR's predictable scaling.

ciency metrics reveal fundamental improvements, with 1.6× better memory-per-token at 1K (3.58 vs 7.97MB) growing to 2.1× at 32K (0.45 vs 0.71MB). These results validate KGERMAR's optimized memory architecture, effectively addressing the quadratic growth limitations of conventional approaches while maintaining stable performance across all sequence lengths.

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	Prep	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: Comparative performance scaling of KGERMAR and ERMAR on WikiText-103. KGERMAR achieves superior perplexity (lower values) with 2–3× better memory efficiency, while ERMAR maintains higher throughput, particularly for shorter sequences. Note: Prep denotes Perplexity.

KGERMAR consistently achieves a superior memory–performance balance. At 16K tokens, it reduces memory usage by up to 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. KGERMAR is particularly effective in long-context settings, exhibiting lower memory consumption, better perplexity, and more stable throughput degradation compared to ERMAR (39% vs. 45% drop from 1K to 16K).

In addition, KGERMAR shows near-linear latency growth with sequence length ($R^2 = 0.98$), whereas ERMAR follows a higher-order trend ($R^2 = 0.89$), making KGERMAR more suitable for real-time long-context inference. Overall, these results validate the scalability and robustness of KGERMAR, especially in memory-critical and long-context scenarios.

6 Conclusions

We develop a novel model, KG-ERMAR, that enhances memory augmented retrieval through contextual knowledge graph construction and hybrid text-graph embeddings for improved long-context modeling. Unlike existing approaches that rely on static knowledge bases, KG-ERMAR constructs domain-specific knowledge graphs directly from input text and integrates graph-structural information with textual semantics through a multi-modal memory architecture. Our experimental results demonstrate significant improvements: KG-ERMAR achieves up to 8.5% lower perplexity and 2-2.5× better memory efficiency across context lengths from 1K to 32K tokens compared to strong baselines including ERMAR and MemLong, while consistently outperforming larger 7B models and showing superior in-context learning performance across five NLU tasks.

7 Limitations

KG-ERMAR introduces additional computational overhead due to on-the-fly knowledge graph construction, resulting in higher inference latency (Appendix B). While this cost is acceptable in research settings, production deployment may require optimizations such as offline or incremental graph construction. Future work will investigate more efficient graph extraction pipelines and extend the framework to domain-specific and multilingual scenarios.

8 Ethical Considerations

This study did not involve human participants, animal experiments, or the use of sensitive, personally identifiable, or otherwise ethically restricted data. All datasets and materials used are publicly available and adhere to their respective terms of use. Therefore, no additional ethical approval was required.

Besides that, the KG-ERMAR model centers on data privacy, algorithmic bias, and the potential for misinformation. The real-time construction of a knowledge graph from input text poses a risk to privacy by potentially extracting and exposing sensitive personal information. Bias from the training data of the entity and relation extraction models could lead to unfair or discriminatory outcomes. There is a risk of the model propagating falsehoods if the source text contains misinformation, as the system lacks an external factual verification mechanism.

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] 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*.

[20] 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.

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

[22] 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.

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

[24] 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*.

[25] 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*.

[26] 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*.

[27] 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.

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

[29] 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*.

[30] 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*.

[31] 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*.

[32] 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*.

[33] 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*.

[34] 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.

[35] 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.

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

[37] 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.

[38] 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*.

[39] 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.

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

[41] 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*.

[42] 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.

[43] 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.

[44] 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*.

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

[46] 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*.

[47] 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 KG-ERMAR 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:

$$\text{Merge}(e_i, e_j) \iff \text{sim}_{string}(e_i, e_j) > \theta_{str} \vee \text{sim}_{embed}(e_i, e_j) > \theta_{ctx} \quad (13)$$

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

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

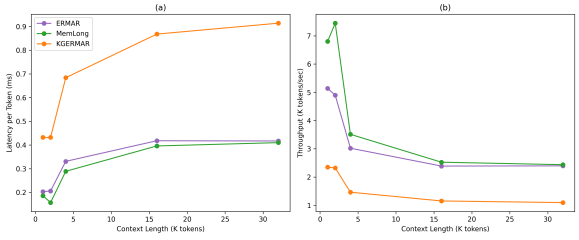

Figure 5: Speed performance comparison showing: (a) **Per-token latency** demonstrating KGERMAR's consistent processing time (0.43-0.91ms) versus baselines' scaling; (b) **Throughput** highlighting KGERMAR's stable token generation (1.1-2.3K tokens/sec) despite higher absolute latency. Error bars represent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

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:

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

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.

Context	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.

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 KG-ERMAR 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 relation-

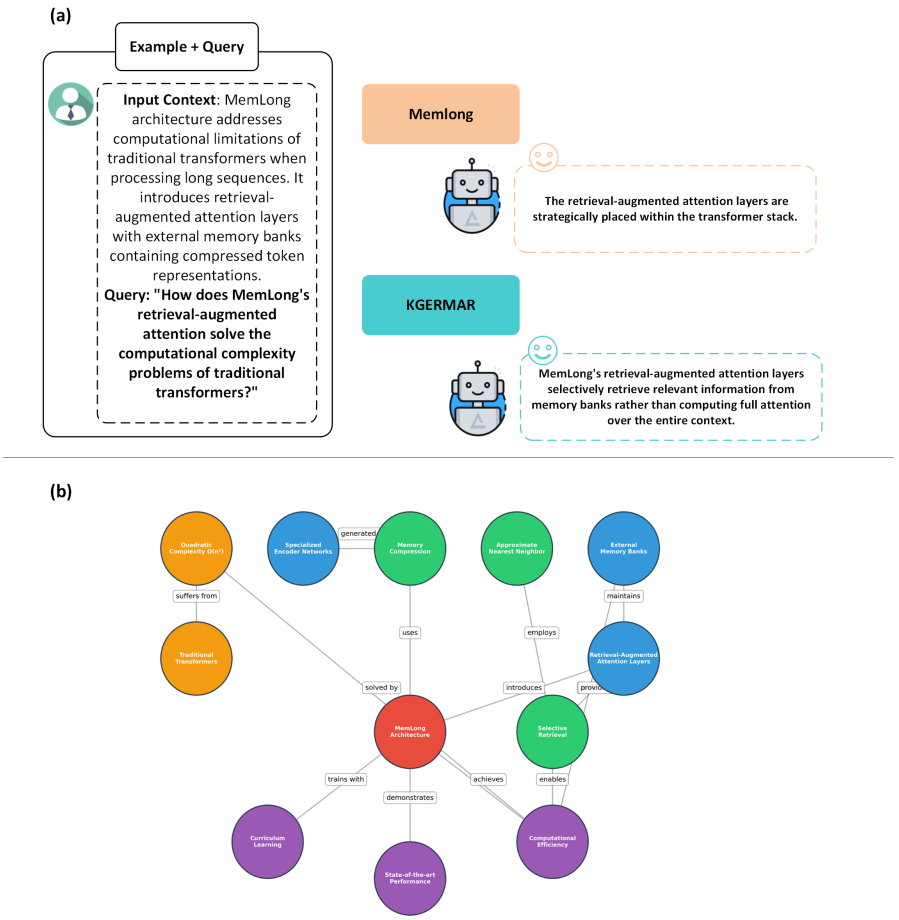

Figure 6: (a) Memlong vs KGERMAR: KGERMAR achieved improved answer completeness through structured knowledge representation; (b) Example of knowledge graph-generation.

ship modeling, but comes at the cost of reduced throughput. This latency overhead could be significantly reduced by pre-computing 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 KG-ERMAR'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

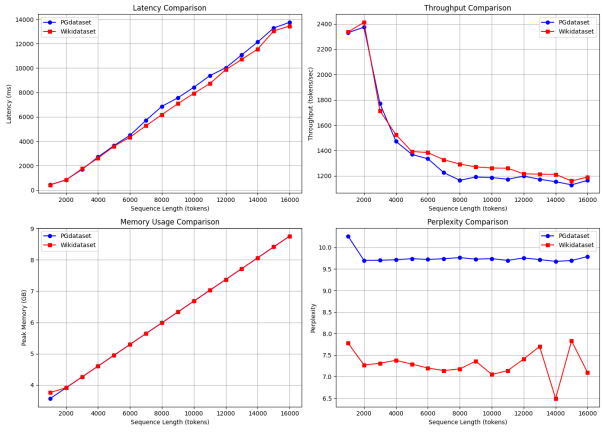

Figure 7: KG-ERMAR scaling analysis across PG-19 and WikiText-103: (a) Near-linear latency scaling; (b) Stable throughput >1K tokens/sec; (c) Linear memory growth; (d) Consistent perplexity across context lengths.

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 KG-ERMAR's suitability for production environments requiring predictable resource usage and consistent performance across variable context lengths.

C.2 Demonstrative Example

The example in Figue 6 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.