SAG: Enhancing Domain-Specific Information Retrieval with Semantic-Augmented Graphs

Carol-Luca Gasan and Vasile Păiș
Research Institute for Artificial Intelligence "Mihai Drăgănescu", Romanian Academy
gasancarolluca@gmail.com, vasile@racai.ro

Abstract

Retrieval-Augmented Generation (RAG) systems rely on high-quality embeddings to retrieve relevant context for large language models. This paper introduces the Semantic-Augmented Graph (SAG), a new architecture that improves domain-specific embeddings by capturing hierarchical semantic relationships between text segments. Inspired by human information processing, SAG organizes content from general to specific concepts using a graph-based structure. By combining static embeddings with dynamic semantic graphs, it generates context-aware representations that reflect both lexical and conceptual links. Experiments on text similarity and domain-specific question answering show that SAG consistently outperforms standard embedding methods within RAG pipelines.

1 Introduction

Retrieval-Augmented Generation (RAG) enhances large language models by integrating relevant external information during response generation. This process relies on embedding models to convert text into vector representations for efficient similarity-based retrieval. While general-purpose embeddings perform well across many domains, they often fall short in specialized fields like medicine, law, or technical writing, where semantic structures are more complex and hierarchical. To address this gap, we propose the Semantic-Augmented Graph (SAG), a novel architecture designed to improve domain-specific embeddings by modeling how humans organize knowledge. SAG converts unstructured text into a directed graph, with nodes representing semantic units and edges encoding contextual relationships. These graphs are further optimized into subtrees that reflect domain-relevant knowledge clusters, capturing both broad concepts and detailed information.

SAG is particularly suited for environments with limited computational resources, where large-scale models are impractical. By enhancing static embeddings with graph-based structure, SAG achieves an effective balance between efficiency and representational power. The approach aligns with ongoing efforts to reduce the energy and compute costs of NLP systems, offering a scalable solution for domain-adapted retrieval and reasoning.

The rest of the paper is structured as follows: Section 2 reviews related work on embeddings and RAG methods. Section 3 outlines the conceptual foundation of our approach. Section 4 details the SAG architecture and processing pipeline. Section 5 presents experimental results on semantic similarity and question answering. Section 6 concludes with future directions.

2 Related Work

Similarity search in high-dimensional vector spaces is a core component of information retrieval and knowledge discovery. Faiss (Johnson et al., 2019), developed by Meta's AI research team, is a widely adopted library that enables scalable similarity search and clustering. It supports both CPU and GPU execution and implements efficient algorithms using techniques like product quantization and inverted indexing. Faiss allows rapid retrieval from large datasets while supporting multiple distance metrics, making it a foundational tool for retrieval-augmented systems.

Learning effective graph representations is essential for tasks such as node classification, link prediction, and clustering. Node2Vec (Grover and Leskovec, 2016) addresses this by generating node embeddings through biased random walks that capture various neighborhood structures. By preserving network proximity in the embedding space, Node2Vec generalizes earlier methods and provides reliable performance across multiple graph-based applications.

*Proceedings of the 14th Joint Conference on Lexical and Computational Semantics (*SEM 2025)*, pages 362–371

Evaluating retrieval-augmented generation systems in specialized domains like medicine requires robust benchmarks. MIRAGE (Rajpurkar et al., 2020) offers a comprehensive suite of medical question-answering datasets and an evaluation framework designed to reflect realistic use cases. Some of them are BioASQ (Tsatsaronis et al., 2015), MedMCQA (Pal et al., 2022), MedQA-USMLE (Marro et al., 2023), and MMLU-MED (Hendrycks et al., 2021). Recent advances in biomedical retrieval have leveraged contrastive learning to improve semantic representations. MedCPT (Yuan et al., 2022) introduces a transformer model trained with contrastive objectives to support zero-shot retrieval in the biomedical domain. It achieves state-of-the-art results on standard benchmarks by generating robust sentence embeddings even in the absence of annotated training pairs.

Knowledge graphs provide structured representations of domain-specific information, supporting complex reasoning and query answering. In the medical field, the traditional static knowledge graphs are being replaced by adaptive systems that incorporate large language models and domain-specific retrieval tools. These systems automatically extract entities, infer relationships, and build query-specific graphs. Frameworks like AMG-RAG (Wang et al., 2023) use these adaptive knowledge graphs in retrieval-augmented pipelines, combining evidence retrieval and reasoning to improve accuracy and interpretability in medical question answering.

3 Model's Philosophy

SAG is inspired by the hierarchical structure of human cognition, where knowledge is processed from general principles to specific details. This structure is especially prominent in expert domains, where reasoning involves traversing semantic layers—from broad categories to precise instances. We define information specificity as "the range of knowledge from which a sentence is derivable". General statements apply broadly, while specific ones are valid in narrower contexts. Consider this spectrum from the biomedical domain:

- General: "Medications are substances used to treat medical conditions."
- Intermediate: "Anticoagulants prevent blood clot formation."
- Specific: "Warfarin inhibits vitamin K epoxide reductase, preventing activation of clotting factors II, VII, IX, and X."

SAG operationalizes this cognitive structure as a directed graph: nodes represent text chunks, and edges encode general-to-specific relationships. This graph captures domain-specific knowledge hierarchies, which we leverage to produce semantically rich embeddings tailored for RAG systems.

4 SAG's Pipeline

4.1 Data and Static Word Embeddings

We begin with unstructured domain-specific text, segmented into semantically coherent chunks (hereafter called paragraphs). Tokenization is handled using a standard English tokenizer with lowercasing; we found no substantial benefit from domain-specific tokenizers in our setup. Initial embeddings are drawn from pre-trained models such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), serving as a static foundation. SAG then fine-tunes these representations through graph-based enrichment. While domain-specific embeddings can improve convergence speed, our results show that SAG's performance is ultimately robust to the choice of initialization.

4.2 Generating initial graph

We aim to construct a directed graph where nodes represent text paragraphs, and edges represent semantic relationships between them. Initially, we create a fully connected graph where each node corresponds to a paragraph. For every pair of nodes, we calculate a similarity score using cosine similarity between their static embeddings. Edges are only preserved if their similarity exceeds a threshold τ (empirically set to 0.5 in our experiments). The edge weight is computed as:

$$w_{ij} = \frac{(\cos(\vec{p}_i, \vec{p}_j) + 1)/2 - \tau}{1 - \tau} \tag{1}$$

where $\vec{p}_i$ and $\vec{p}_j$ are the paragraph embeddings, and the weight is mapped to the range $[0, 1]$.

To compute paragraph embeddings, we use the Smooth Inverse Frequency (SIF) weighting scheme. SIF reduces the influence of common words and emphasizes the importance of informative words in the paragraph. The paragraph embedding $\vec{p}_i$ is computed as:

$$\vec{p}_i = \frac{1}{|T_i|} \sum_{t \in T_i} \frac{a}{a + f(t)} \cdot \vec{t} \tag{2}$$

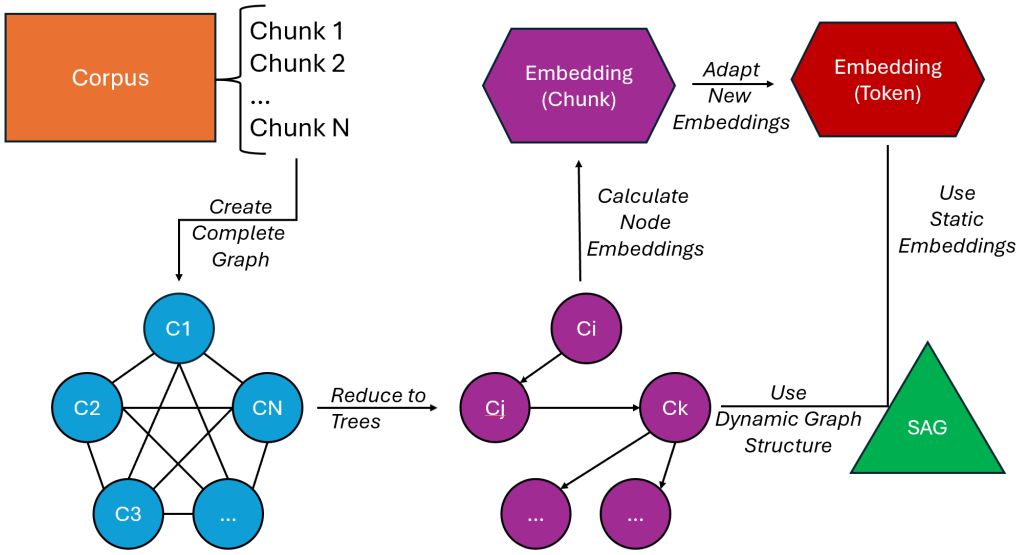

Figure 1: Overview on SAG's pipeline

where T_i is the set of tokens in paragraph i, a is a smoothing parameter (empirically set to 0.0002), $f(t)$ is the frequency of token t in the corpus, and $\vec{t}$ is the token embedding.

In our experimental design, we deliberately selected static word embeddings augmented with SIF for paragraph representations, despite the availability of various sentence and text embedding alternatives. This approach enables us to isolate and highlight the architectural benefits of SAG itself, rather than potentially obscuring its contributions by applying it to already optimized state-of-the-art models, which would make it difficult to distinguish between innate model performance and SAG-derived improvements.

To transform this undirected similarity graph into a directed graph that captures the general-to-specific relationship, we use the SMOG readability index (McLaughlin, 1969), which is particularly effective for medical texts. The SMOG index provides a measure of textual complexity, which we use as a proxy for specificity: higher SMOG scores indicate more specific content, while lower scores suggest more general content. The SMOG index is calculated as:

$$\text{SMOG} = 1.043\sqrt{N_p \left(\frac{30}{N_s}\right)} + 3.1291 \quad (3)$$

where N_p is the number of polysyllabic words (words with three or more syllables) and N_s is the number of sentences.

For each pair of connected nodes (i, j), we compute their respective SMOG scores s_i and s_j, and derive a directionality measure:

$$c_i = \frac{s_i}{s_i + s_j}, \quad c_j = \frac{s_j}{s_i + s_j} = 1 - c_i \quad (4)$$

We then establish a directed edge from the node with the lower c value (more general) to the node with the higher c value (more specific). This process transforms our similarity graph into a directed graph that represents the flow from general to specific information.

4.3 Reducing the graph in size

The fully connected directed graph, while theoretically comprehensive, is computationally inefficient and contains redundant information. Our goal is to reduce this graph to a more manageable structure while preserving the most important semantic relationships. Specifically, we aim to transform the graph into a forest of directed trees, where each tree represents a coherent subdomain of knowledge. We employ a heuristic algorithm that combines topological sorting with dynamic programming to identify optimal subtrees, formally described in Algorithm 1. The distinction between trees naturally emerges from the semantic structure of the data, without requiring explicit domain labels. Each tree captures a semantic hierarchy from general to specific information within its subdomain.

Algorithm 1 Graph Reduction to Forest

procedure REDUCETOFOREST($G = (V, E)$)
    Initialize $visited[node] \leftarrow 0$ for all $node \in V$
    Initialize $stack \leftarrow []$
    function DFS($node$)
        $visited[node] \leftarrow 1$
        for each neighbor v of $node$ in E **do**
            if $visited[v] = 0$ **then**
                DFS(v)
            end if
        end for
        $stack$.append($node$)
    end function
    for each $node \in V$ **do**
        if $visited[node] = 0$ **then**
            DFS($node$)
        end if
    end for
    Reverse $stack$
    Initialize $costs[node] \leftarrow \infty$ for all $node$
    Initialize $oset \leftarrow$ sorted list of $(\infty, node, [])$ for all $node$
    $new\_edges \leftarrow []$
    while $oset$ not empty **do**
        $(node, curr\_cost, edges) \leftarrow$ pop node with minimum cost from $oset$
        if $curr\_cost = \infty$ **then**
            $curr\_cost \leftarrow 0$
        end if
        $costs[node] \leftarrow curr\_cost$
        for each $(prev\_node, edge\_cost)$ in $edges$ **do**
            new_edges.append($(prev\_node, node, \{weight : 1 - edge\_cost\})$)
        end for
        for each outgoing edge $(node, neighbor, weight)$ in E **do**
            if $neighbor$ is still in $oset$ **then**
                if $curr\_cost + weight <$ current cost of $neighbor$ **then**
                    Update $neighbor$'s cost to $curr\_cost + weight$
                    Record path from $node$
                else if $curr\_cost + weight =$ current cost of $neighbor$ **then**
                    Add alternative path from $node$
                end if
            end if
        end for
    end while
    Replace G's edges with new_edges
    return G
end procedure

4.4 Computing updated word embeddings

With the optimized forest structure in place, we now compute enhanced node embeddings that incorporate both the textual content and the graph structure. We use Node2Vec to generate embeddings for each node in the forest, capturing the structural relationships between paragraphs. For individual tokens, we compute enriched embeddings by taking a weighted average of the node embeddings for all paragraphs in which the token appears:

$$\vec{t}_{\text{new}} = \sum_{i \in P(t)} \frac{f_{t,i}}{\sum_{j \in P(t)} f_{t,j}} \cdot \vec{n}_i \tag{5}$$

where $P(t)$ is the set of paragraphs containing token t, $f_{t,i}$ is the frequency of token t in paragraph i, and $\vec{n}_i$ is the node embedding for paragraph i.

Rather than completely replacing the original static embeddings, we merge them with these new embeddings to preserve general semantic information while incorporating domain-specific relationships. The combined embeddings are computed as:

$$\vec{t}_{\text{final}} = [\vec{t}_{\text{static}}; \vec{t}_{\text{new}}] \tag{6}$$

After merging the static and new embeddings, the resulting dimensionality increases. To ensure that the final embeddings match the original static embeddings' dimensionality (for seamless integration with existing systems), we apply dimensionality reduction techniques. Specifically, we use the method described in (Raunak et al., 2019) to reduce the size of the word embeddings. The implementation can be seen in Algorithm 2.

Algorithm 2 Dimensionality Reduction via PCA

Input: Data matrix X_{train}, components O, reduced dims N, desired comps D, words W
Output: Reduced embeddings $\vec{t}_{\text{final}}$
Center data: $X_{\text{train}} \leftarrow X_{\text{train}} - \text{mean}(X_{\text{train}})$
$U_1 \leftarrow \text{PCA}(X_{\text{train}}, O)$ ▷ First PCA: extract main components
for each $x \in X_{\text{train}}$ **do**
    $x \leftarrow x - \sum_{j=1}^{D} \text{dot}(U_1[j]^T, x) \cdot U_1[j]$ ▷ Remove top D components
end for
$X_{\text{train}} \leftarrow X_{\text{train}} - \text{mean}(X_{\text{train}})$
$X_{\text{new}} \leftarrow \text{PCA}(X_{\text{train}}, N)$ ▷ Project to N dimensions
$X_{\text{new}} \leftarrow X_{\text{new}} - \text{mean}(X_{\text{new}})$
$X_{\text{new}} \leftarrow \text{PCA}(X_{\text{new}}, N)$ ▷ Second PCA refinement
for each $i \in 1..W$ **do**
    $\vec{t}_{\text{final}}[i] \leftarrow X_{\text{new}}[i] - \sum_{j=1}^{D} \text{dot}(U_{\text{fit}}[j]^T, X_{\text{new}}[i]) \cdot U_{\text{fit}}[j]$
end for
Return $\vec{t}_{\text{final}}$

4.5 Encoding logic

4.5.1 Single-Tree

The SAG encoding process integrates static and dynamic components to produce embeddings that capture both lexical content and structural semantics. The static component uses a weighted average of token embeddings with SIF weighting to downplay common terms. The dynamic component incorporates domain-specific relationships by simulating a walk through the semantic graph, as described in Algorithm 3. The final embedding is a weighted combination of the two:

$$\vec{e} = p_{\text{SAG}} \cdot \vec{d} + (1 - p_{\text{SAG}}) \cdot \vec{s} \tag{7}$$

where p_{SAG} (typically 0.7) balances the influence of the graph-based and lexical components.

4.5.2 Multiple-Trees

In practice, our graph reduction algorithm typically produces multiple trees, each representing a distinct subdomain within the corpus. For each text to be encoded, we identify the most relevant trees (termed "master trees") based on token overlap. Specifically, for a given text, we identify the top k trees (where $k = \sqrt{|trees|}$ by default) with the highest token frequency overlap.

When comparing two texts, we compute similarity scores for each tree in the union of their respective master tree sets, with trees that appear in both sets receiving double weight. The final similarity score is:

$$\text{sim}(A, B) = \frac{\sum_{t \in M_A \cup M_B} w_t \cdot \cos(\vec{e}_A^t, \vec{e}_B^t)}{\sum_{t \in M_A \cup M_B} w_t} \tag{8}$$

where M_A and M_B are the sets of master trees for texts A and B, $\vec{e}_A^t$ and $\vec{e}_B^t$ are the embeddings of A and B with respect to tree t, and $w_t = 2$ if $t \in M_A \cap M_B$ and $w_t = 1$ otherwise.

Algorithm 3 Dynamic Embedding Computation

procedure COMPUTEDYNAMICEMBEDDING($text, G, nodeEmb, tokenFreq$)
  $tokens \leftarrow$ tokenize($text$)
  $span \leftarrow tokens[0 : \text{SPAN\_SIZE}]$, $last \leftarrow$ SPAN_SIZE
  $sumVec \leftarrow \sum_{t \in span} tokenFreq[t]$
  $currNode \leftarrow \arg\max_n sumVec[n]$, $vecs \leftarrow [nodeEmb[currNode]$
  $temp \leftarrow T_0$ ▷ Initial temperature
  while $last < |tokens|$ **do**
    Update $span$ and $sumVec$ with $tokens[last]$
    $last \leftarrow last + 1$
    $steps \leftarrow$ COMPUTESTEPS($span, tokenFreq$), $tmp \leftarrow []$
    for $1 \ldots steps$ **do**
      With prob. $\propto e^{-1/temp}$, pick random neighbor of $currNode$
      Otherwise, set $currNode \leftarrow \arg\max$ neighbor by $sumVec$
      $temp \leftarrow temp \cdot \lambda$
      tmp.append($nodeEmb[currNode]$)
    end for
    $vecs$.append(tmp)
  end while
  return mean($vecs$)
end procedure

4.6 Inference high-level methods

While the multiple-tree approach provides rich semantic representations, it requires a custom similarity function, which may not integrate seamlessly with existing embedding-based tools and pipelines. To address this, we propose a compact embedding format that encapsulates the multi-tree information within a single vector representation, facilitating compatibility with standard similarity metrics.

4.6.1 Compact Embedding Formulation

Let $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ be the set of all trees in our forest, and $\mathcal{M}_x \subset \mathcal{T}$ be the set of master trees for text x. The compact embedding $\vec{c}_x$ is constructed as a concatenation of three components:

$$\vec{c}_x = [\vec{s}_x^{\text{weighted}}; \vec{e}_x^{\text{specific}}; \vec{e}_x^{\text{general}}] \tag{9}$$

where:

- $\vec{s}_x^{\text{weighted}} = r_{\text{sent}} \cdot \vec{s}_x$ represents the static component weighted by r_{sent}
- $\vec{e}_x^{\text{specific}} = [p_{\text{SAG}} \cdot \vec{e}_x^i \text{ if } T_i \in \mathcal{M}_x \text{ else } \vec{0}]_{i=1}^n$ represents the tree-specific embeddings for master trees
- $\vec{e}_x^{\text{general}} = [p_{\text{SAG}} \cdot \vec{e}_x^i]_{i=1}^n$ represents the general tree embeddings across all trees

The weighting factor r_{sent} is computed as:

$$r_{\text{sent}} = \begin{cases} 2 \cdot |\mathcal{M}_x| \cdot p_{\text{sent}} & \text{if } |\mathcal{M}_x| > 0 \\ p_{\text{sent}} & \text{otherwise} \end{cases} \tag{10}$$

where $p_{\text{sent}} = 1 - p_{\text{SAG}}$.

4.6.2 Theoretical Justification

The compact embedding design ensures that when comparing two texts using standard cosine similarity, the result approximates our weighted multi-tree similarity. Consider the dot product between compact embeddings of texts x and y:

$$\vec{c}_x \cdot \vec{c}_y = r_{\text{sent}}^2 (\vec{s}_x \cdot \vec{s}_y) + p_{\text{SAG}}^2 \sum_{i=1}^{n} \left(1 + I_{T_i \in \mathcal{M}_x \cap \mathcal{M}_y}\right) (\vec{e}_x^i \cdot \vec{e}_y^i) \tag{11}$$

where $I(\cdot)$ is the indicator function and $I(A)I(B) = I(A \cap B)$ was used to merge the two sums.

When normalized by the magnitudes in the cosine similarity calculation, this approximates our weighted multi-tree similarity Formula 8. The compact embedding computation is implemented efficiently as shown in Algorithm 4.

Algorithm 4 Compact Embedding Computation

procedure ENCODE($text, p_{SAG} = 0.7, inference = True$)
  $p_{sent} \leftarrow 1 - p_{SAG}$
  $sent \leftarrow$ ComputeStaticEmbedding($text$)
  $sent \leftarrow \frac{sent}{||sent||}$ ▷ Normalize
  $masters, sag \leftarrow$ ComputeSAGVectors($text$)
  $r_{sent} \leftarrow 2.0 \cdot |masters| \cdot p_{sent}$ if $|masters| > 0$ else p_{sent}
  $sentPart \leftarrow [r_{sent} \cdot sent]$
  $specificPart \leftarrow [p_{SAG} \cdot sag[i]$ if $i \in masters$ else $\vec{0}$ for i in range($|sag|$)$]$
  $generalPart \leftarrow [p_{SAG} \cdot x$ for x in $sag]$
  if inference **then**
    $parts \leftarrow [sentPart, specificPart, generalPart]$
  else
    $parts \leftarrow [sentPart, generalPart, specificPart]$
  end if
  $result \leftarrow []$
  for $part \in parts$ **do**
    for $vector \in part$ **do**
      $result$.extend($vector$.flatten())
    end for
  end for
  return array($result$)
end procedure

4.7 Parameter Discussion

Parameters fall into training and inference categories. Training's key factors are chunk size and graph node count (controlling granularity/scale); similarity threshold mainly reduces computation by limiting edges without affecting structure. For inference, embedding weight between static/dynamic components is crucial and tunable per use case. The 50-dim embeddings balance performance with speed, aligning with typical subtree counts (6-10); larger sizes showed comparable accuracy but slower inference.

5 Experiments

We evaluated SAG on text similarity and RAG-based question answering tasks. To optimize training data quality, we utilized the CRAFT 2.0 dataset (Cohen et al., 2017) containing 67 full-text articles with around 560,000 tokens. Our training employed 1024 chunks of 512 tokens each with final static embeddings of size 50.

5.1 Text Similarity

We evaluated SAG on two biomedical semantic textual similarity datasets: Clinical STS EBMSASS (Hassanzadeh et al., 2019), containing 1,000 expert-annotated clinical evidence pairs with 1-5 similarity scores from biomedical abstracts, and BIOSSES (Soganciogl u et al., 2017), comprising 100 sentence pairs from biomedical articles with 0-4 similarity scores focused on citation relationships. Evaluation employed a domain-specific pretrained Sentence Transformer trained on the PubMed corpus (Wheeler et al., 2002). Table 4 demonstrates SAG's state-of-the-art performance, exceeding both the reference model and previous SOTA on BIOSSES, and marginally outperforming the reference model on EBMSASS.

5.2 Question Answering with RAG

We evaluated SAG on the MIRAGE benchmark while considering two retrieval corpora: LoData (medical textbooks and Statpearls (Publishing, 2025)) and HiData (25% of PubMed). Table 2 demonstrates that RAG augmented with SAG consistently outperforms baseline approaches using Gemini 2.0 Flash. Notably, Table 5 indicates SAG's minimal computational overhead, with reduced inference time and memory requirements in both single-threaded and accelerated environments, confirming its suitability for resource-constrained deployments. SAG enhances standard RAG pipelines to competitive levels with special-

Model	Size	Acc. (%)	Chain-of-Thought	Web-Search
Med-Gemini (Saab et al., 2024)	1800B	91.1	Yes	Yes
GPT-4 (OpenAI, 2023)	1760B	90.2	Yes	Yes
Gemini 2.0 Flash + RAG w/ SAG (HiData)	40B	87.7	Yes	No
Med-PaLM 2 (Singhal et al., 2025)	340B	85.4	Yes	No
Med-PaLM 2 (5-shot) (Singhal et al., 2025)	340B	79.7	Yes	No
AMG-RAG (Wang et al., 2023)	8B	73.9	Yes	Yes
Meerkat (Kim et al., 2024)	7B	74.3	Yes	No
Meditron (Chen et al., 2023)	70B	70.2	Yes	Yes
Flan-PaLM (Singhal et al., 2023)	540B	67.6	Yes	No
LLAMA-2 (Chen et al., 2023)	70B	61.5	Yes	No
Shakti-LLM (Shakhadri et al., 2024)	2.5B	60.3	No	No
BioMedGPT (Luo et al., 2023)	10B	50.4	No	No

Table 1: Recent results on MedQA-US.

Model	BioASQ	MedMCQA	MedQA-US	MMLU-MED	Avg
RAG with SAG (HiData)	74.13	71.88	87.65	90.16	80.95
RAG with MedCPT (HiData)	66.55	69.43	84.23	85.71	76.48
RAG with SAG (LoData)	72.35	67.85	67.45	83.49	72.79
RAG with MedCPT (LoData)	65.12	66.73	65.42	79.96	69.31
No RAG	60.35	64.50	67.01	78.84	67.68

Table 2: Accuracy (%) on Question Answering Datasets using Gemini 2.0 Flash

Model	Acc.
SAG (HiData)	71.88
AMG-RAG (Wang et al., 2023)	66.34
Meditron (Chen et al., 2023)	66.00
Codex 5-shot (Liévin et al., 2024)	59.70
VOD (Liévin et al., 2023)	58.30
PubmedBERT (Gu et al., 2021)	40.00

Table 3: Recent results on MedMCQA

Dataset	SAG	Reference	SOTA
EBMSASS	85.84	85.62	–
BIOSSES	96.87	86.67	93.63

Table 4: Pearson Correlation (%) on biomedical STS datasets

Model	Size	Time S/A	Memory S/A
SAG	124 MB	136/32 ms	1/7 KB
MedCPT	439 MB	350/62 ms	7/23 MB

Table 5: Inference Time and Memory Usage per 1k tokens in (S)ingle-threaded and (A)ccelerated settings

ized medical models, achieving 87.7% on MedQA-USMLE (Table 1) and 71.88% on MedMCQA (Table 3). These results demonstrate SAG's efficacy as a lightweight enhancement module enabling modest LLMs to compete with state-of-the-art systems in complex biomedical reasoning.

6 Conclusion and Future Work

This work presents the Semantic-Augmented Graph (SAG), a new architecture that improves domain-specific embeddings by organizing semantic relationships hierarchically within a directed graph. By structuring information from general to specific concepts, SAG produces context-aware embeddings while remaining more computationally efficient than traditional deep learning methods. Experimental results show that SAG achieves state-of-the-art performance on biomedical semantic similarity benchmarks. When integrated into a Retrieval-Augmented Generation system, it consistently improves biomedical question-answering performance while retaining its efficiency. Future research may explore expanding SAG to general-domain applications, adapting it for symbolic and mathematical data, combining multiple embedding sources through ensemble methods, and extending the model to low-resource languages where efficient learning is critical. We anticipate releasing the code and related artifacts in the future to support reproducibility and further research.

7 Limitations

Despite the promising results, our study is subject to several limitations. First, due to hardware constraints when processing large-scale graphs with Python-based backend libraries, the Semantic-Augmented Graph (SAG) was restricted to a maximum of 1024 nodes during training and inference. Our implementation using NetworkX for graph operations on million-edge structures faced memory and processing bottlenecks, despite running on dual Intel Xeon Silver 4210 CPUs (2.20GHz, 40 total logical cores) and two NVIDIA GPUs (Quadro RTX 6000/8000 and RTX 5000). While the node limitation was sufficient for capturing meaningful hierarchical structures, it may have prevented deeper modeling of larger corpora. Second, during RAG evaluation, only a quarter of the full PubMed dataset was accessible, which may have constrained the retrieval coverage and affected downstream performance. These limitations highlight potential gains from scaling SAG with more efficient graph processing backends and accessing larger retrieval corpora. As with any embedding-based system applied in biomedical contexts, there is a potential risk of encoding latent biases or producing misleading similarity scores that may impact downstream clinical decisions. We recommend caution and further evaluation before deployment in real-world healthcare settings.

8 Acknowledgements

This research was in part sponsored by the NATO Science for Peace and Security Programme under grant id. G8648.

References

Mark Chen, Khaled Saab, Harsha Nori, Aakanksha Chowdhery, Tiffany Kuo, Jure Lee, Joon Lee, Yifan Zhang, Qian Jin, and Raj Patel. 2023. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.00000*.

K. Bretonnel Cohen, Karin Verspoor, Karën Fort, Christopher Funk, Michael Bada, Martha Palmer, and Lawrence E. Hunter. 2017. The colorado richly annotated full text (craft) corpus: Multi-model annotation in the biomedical domain. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 1379–1394. Springer Netherlands, Dordrecht.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Pubmedbert: A domain-specific language model for biomedical text. *arXiv preprint arXiv:2007.15779*.

Hamed Hassanzadeh, Tudor Groza, and Jane Hunter. 2019. Ebmsass: Evidence-based medicine sentence similarity dataset. In *Proceedings of the 10th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 652–657.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Zou, Dawn Lee, Dawn Tang, Dawn Song, Jacob Steinhardt, Justin Gilmer, Erica Ma, Gaurav Sastry, Andy Tran, Xander Wang, Andy Miller, Alexander D'Amour, Andrew Lohn, David Krueger, and 23 others. 2021. Measuring massive multitask language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 23711–23723.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Jinhyuk Kim, Wonjin Kim, Jinhyuk Lee, Joongbo Lee, Kyunghyun Lee, Sunghwan Yoon, and Jaewoo Kang. 2024. Meerkat: A medical reasoning benchmark for large language models. *arXiv preprint arXiv:2402.00000*.

Valentin Liévin, Charlotte E Hother, and Ole Winther. 2023. Vod: Visual open-domain question answering. *arXiv preprint arXiv:2305.00000*.

Valentin Liévin, Charlotte E Hother, and Ole Winther. 2024. Can large language models reason about medical questions? *npj Digital Medicine*.

Ruijie Luo, Yuan Li, Yuxuan He, Yuxian Wang, Xiaodan Zhang, Shijie Wang, Shuo Zhang, Xiaozhi Liu, Zhiyuan Liu, and Maosong Sun. 2023. Biomedgpt: A unified and generalist biomedical generative pretrained transformer for biomedical text, image, and cross-modal tasks. *arXiv preprint arXiv:2306.00000*.

Stefano Marro, Benjamin Molinet, Elena Cabrio, and Serena Villata. 2023. Natural language explanations for clinical case retrieval. In *Artificial Intelligence in Medicine*. Springer.

G. Harry McLaughlin. 1969. Smog grading: A new readability formula. *Journal of Reading*, 12:639–646.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshops*.

OpenAI. 2023. Gpt-4 technical report. ArXiv preprint arXiv:2303.08774.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on Health, Inference, and Learning*, pages 248–260. PMLR.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

StatPearls Publishing. 2025. Statpearls [internet]. Online medical reference, regularly updated. Available from: https://www.statpearls.com/.

Pranav Rajpurkar, Emily Chen, Bridget McCall, Raj Patel, Jenny Liu, Jeremy Irvin, James Zou, Ross Jones, Nikhil Kohli, Tony Duan, and Daisy Ding. 2020. Mirage: A large-scale medical qa benchmark for robustness and generalization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243, Florence, Italy. Association for Computational Linguistics.

Khaled Saab, Harsha Nori, Aakanksha Chowdhery, Tiffany Kuo, Jure Lee, Joon Lee, Yifan Zhang, Qian Jin, Raj Patel, and Percy Liang. 2024. Capabilities of gemini models in medicine. *arXiv preprint arXiv:2404.18416*.

A Shakhadri, S Sahoo, S Saha, A Kumar, S Kumar, A Gupta, and A Sharma. 2024. Shakti-llm: An open-source large language model for indian languages. *arXiv preprint arXiv:2403.00000*.

Karan Singhal, Shekoofeh Azizi, Tong Tu, Soroush Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Hunter Cole, Joon Lee, and Zachary Bradshaw. 2023. Large language models encode clinical knowledge. *Nature*. Doi:10.1038/s41586-023-05881-4.

Karan Singhal, Shekoofeh Azizi, Tong Tu, Soroush Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Hunter Cole, Joon Lee, and Zachary Bradshaw. 2025. Toward expert-level medical question answering with large language models. *Nature*. Doi:10.1038/s41586-024-07366-5.

Gizem Soganciogl u, Hakime Öztürk, and Arzucan Özgür. 2017. Biosses: A semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*, 33(14):i49–i58.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, John Pavlopoulos, Theofanis Karaletsos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Niko Heino, Eric Gaussier, Laura Barrio-Alvers, and 2 others. 2015. Bioasq: A challenge on large-scale biomedical semantic indexing and question answering. *Journal of Biomedical Informatics*, 57:1–7.

Yifan Wang, Linjun Zhang, Yichong Liu, Yichao Zhang, Jimmy Lin, Buzhou Tang, Xiaodan Wang, Hua Xu, Fei Wang, and Yulan He. 2023. Amg-rag: Adaptive medical graphs for retrieval-augmented generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

David L. Wheeler, Deanna M. Church, Scott Federhen, Alex E. Lash, Thomas L. Madden, Joan U. Pontius, Gregory D. Schuler, Lynn M. Schriml, Eduardo Sequeira, Tatiana A. Tatusova, and Lisbeth Wagner. 2002. Pubmed: the bibliographic database. *Nucleic Acids Research*, 30(1):61–65.

Hongyin Yuan, Kai Sun, Qingyu Yu, Buzhou Tang, Xiaolong Wang, Bin Wang, Xiaodan Wang, Yanshan Zhang, Hua Xu, Fei Wang, Yulan He, and Fei Wang. 2022. Medcpt: Contrastive pre-training for biomedical retrieval. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.