LP-RAG: Learning to Retrieve with Link Predictors

Erik J. F. Nascimento
Federal University of Ceará

Jorge Luiz Franco
University of São Paulo & Avra

Amauri H. Souza
Avra & Federal Institute of Ceará

Abstract

Retrieval-augmented generation (RAG) strategies have empowered large language models (LLMs) through integration with external knowledge sources, leading to more accurate, up-to-date, and contextually relevant outputs. Recently, graph-based RAG methods have gained attention for leveraging relational structures that support multi-hop reasoning and retrieval. However, existing approaches remain limited in their ability to explicitly model query-aware relevance signals during retrieval. In this paper, we propose LP-RAG, a graph-based framework for document RAG that casts retrieval as an inductive link prediction problem. In particular, LP-RAG first constructs a graph encoding semantic relationships among chunks (i.e., candidates for retrieval), which is then augmented with chunk-conditioned synthetic queries that emulate potential user questions. This enables self-supervised learning of query-chunk relevance without relying on real query annotations. Retrieval is performed by predicting links between unseen queries and chunk nodes. Notably, LP-RAG is model-agnostic and can leverage a broad class of link prediction methods. To demonstrate its effectiveness, we evaluate LP-RAG across diverse settings and benchmarks. Our empirical results show that LP-RAG consistently outperforms existing methods in both retrieval quality and downstream generation performance, while maintaining competitive efficiency relative to learnable graph-based RAG approaches. Source code at https://github.com/ErikJhones/LP_RAG.

1 Introduction

Large language models (LLMs) [2, 7, 26] have quickly become one of the most transformative technologies of modern era, enabling significant progress across a range of applications, such as conversational agents [44], automated code generation [37], education [38], and healthcare [29, 60]. Yet, their reliance on static corpora prevents access to real-time information, limits coverage of domain-specific knowledge, and hinders reliable encoding of long-tail factual knowledge [25, 40]. To address these limitations, retrieval-augmented generation (RAG) systems [24, 33, 61] couple LLMs with information retrieval strategies, integrating external knowledge into the generation process. Nowadays, RAG constitutes a central component of LLM-based applications, underpinning widely deployed products such as search engines and chat assistants [3].

To identify relevant documents for a given query, early methods primarily relied on lexical retrieval techniques [47] or dense semantic retrievers [22, 27, 33]. However, these approaches typically treat documents independently, limiting their ability to capture global relationships distributed across documents or to exploit dependencies among their textual elements [39, 61]. As a result, they often struggle with complex multi-hop reasoning and summary-level queries that require integrating information from multiple sources [17]. To mitigate these limitations, graph-based RAG [18, 24] have emerged as a promising alternative that organizes textual corpora into graphs, where nodes

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

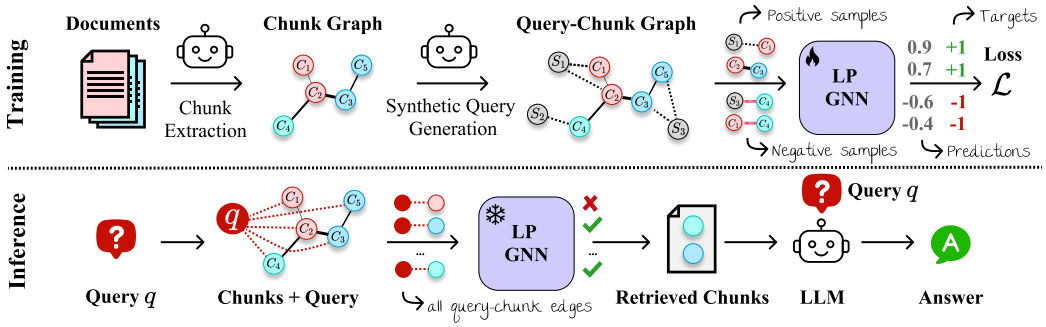


Figure 1: **Overview of LP-RAG**. For training, we leverage LLMs to obtain atomic chunks — from which we build a similarity chunk graph — and chunk-conditioned synthetic queries. These are used to train a GNN for link prediction (LP-GNN). During inference, we feed only the chunk graph (synthetic queries are removed) and the user query to LP-RAG to obtain chunks for generation.

typically represent entities or chunks and edges encode semantic relations. More recent approaches further leverage LLMs to enrich graphs with summary-level information [8, 24], and employ graph neural networks (GNNs) [12, 51] to support reasoning over such structures [20, 40, 41].

Despite these advances, effectively leveraging graph structure to estimate query–chunk relevance remains an important challenge in graph-based RAG. While some approaches incorporate query information into graph-based retrieval, they rely on similarity-based initialization, graph traversal, or learned scoring mechanisms over existing nodes. This motivates exploring alternative formulations that explicitly model query–chunk relevance while exploiting the relational structure of the documents.

In this paper, we propose LP-RAG, a graph-based framework that formulates document retrieval as an inductive link prediction problem. Our key insight is to learn query–chunk relevance by predicting links between incoming queries and candidate chunks, leveraging synthetic queries to provide self-supervised training signals. Importantly, our design enables i) exploiting semantic cues from chunks, and ii) integrating powerful link predictors for retrieval. More specifically, our methodology begins by decomposing documents into semantic units (e.g., chunks). Each such unit is represented as a node, and connections between nodes are established via semantic relationships, e.g., similarity scores from contextual embeddings. Subsequently, synthetic queries are generated by the language model conditioned on the extracted chunks. These chunks, queries, and inter-node connections are then integrated to form a unified graph structure, referred to as the chunk-query graph. Finally, we formulate retrieval as a link prediction task, where the goal is to predict links between an incoming query and chunk nodes. Crucially, the use of synthetic queries provides supervision for training link predictors, enabling LP-RAG to incorporate a wide range of models. Figure 1 shows an overview of LP-RAG.

To evaluate the effectiveness of LP-RAG, we consider ten popular RAG benchmarks. Our results demonstrate that LP-RAG consistently outperforms strong baselines, including NodeRAG [61], HippoRAG [24], LinearRAG [66], and GFM-RAG [40], in open-ended head-to-head evaluations, while maintaining a competitive number of retrieval tokens and latency. We also conduct ablation studies to demonstrate the importance of using synthetic queries, the cross-domain transferability performance of our method, and the impact of different GNNs, chunking methods, and negative sampling schemes.

In sum, our contributions are:

- We formulate document retrieval as an inductive link prediction problem, leveraging synthetic queries to learn query–chunk relevance without requiring annotated query–document pairs;
- We propose LP-RAG, a graph-based retrieval framework that exploits semantic and structural information through general link prediction architectures, including GNN-based predictors;
- We introduce Fast LP-RAG, a scalable variant of LP-RAG that operates on lightweight entity-centric graphs, enabling substantially lower indexing and storage costs;
- We show the effectiveness and efficiency of LP-RAG across several benchmarks, achieving substantial empirical gains over prior graph-based RAG methods.

2 Related Works

Retrieval-Augmented Generation (RAG) [11, 33] provides an effective way to integrate external knowledge into LLMs by retrieving relevant documents to facilitate LLM generation. Early works adopt the pre-trained dense embedding model to encode documents as separate vectors [5, 33, 42], which are then retrieved via similarity scores to the query. Despite efficiency, these methods struggle to capture complex document relationships. Subsequent studies have explored multi-step retrieval, where LLMs guide an iterative process to retrieve and reason over multiple documents [10, 53].

Graph-based RAG [16, 45] leverages graphs to explicitly model the relationships between semantic units/ entities. For instance, GraphRAG [8, 18] and LightRAG [14] incorporate graph structures into text indexing and retrieval, enabling efficient retrieval of entities and their relationships. HippoRAG [24] and HippoRAGv2 [15] enhance multi-hop retrieval by using a personalized PageRank algorithm to locate relevant knowledge. NodeRAG [61] further advances this line of work by introducing heterogeneous graph structures specifically designed to allow a more holistic integration of graph algorithms into the RAG workflow. In contrast to these approaches, LinearRAG [36] revisits the GraphRAG pipeline by avoiding explicit relation extraction and instead constructing a relation-free hierarchical structure through lightweight entity extraction and semantic linking. Efforts to incorporate GNNs into graph-enhanced RAG frameworks [20, 35, 40, 41] have shown strong empirical gains by leveraging message passing to model structured dependencies among documents. More recent approaches further explore different mechanisms for graph-based retrieval. [41] employs message passing over curated knowledge graphs to identify relevant answer entities, while GNN-Ret [35] uses GNNs to refine passage relevance scores over structure- and keyword-relatedness graphs. KG²RAG [65] combines semantic seed retrieval with knowledge graph-guided chunk expansion, whereas KGP relies on an LLM agent to iteratively traverse passage graphs. Despite leveraging graph structures, these methods do not formulate retrieval as inductive link prediction between a newly introduced query node and candidate chunks, as in LP-RAG.

Reasoning on knowledge graphs. Other frameworks focus on constructing and fusing structured knowledge: RAGraph retrieves whole graphs from a library to augment downstream graph learning [23], while Graphusion uses LLMs to extract triplets and fuse them into a global knowledge graph for improved retrieval and reasoning [62]. TOBUGraph [28] instead builds KGs from documents and perform graph-traversal retrieval on them, providing an alternative to text-only similarity retrieval [28].

Synthetic query generation aims to enrich retrieval systems with auxiliary contextual signals by generating plausible queries associated with a document or passage. Early approaches, such as doc2query [13] and Siamese [36], rely on seq2seq architectures [54], particularly models like BART, which are fine-tuned on supervised query-document pairs to map passages into likely user queries. More recent methods, including query2doc [58], QGen [4], and SGenVA [48], leverage prompted LLMs to generate synthetic queries in a zero-shot or few-shot manner, often producing richer contextual supervision signals. These synthetic queries have been used to improve document indexing, retrieval expansion, and retrieval robustness by exposing latent semantic relations that may not be explicitly present in the original text. More closely related to our setting, [46] generate synthetic questions over atomic units and retrieve them through dense similarity matching. Building on this line of work, LP-RAG employs synthetic queries as supervision for learning query–chunk relevance in a graph-based retrieval framework. Rather than retaining generated queries as searchable representations, LP-RAG uses them during training to learn an inductive link predictor that estimates relevance between incoming queries and candidate chunks while exploiting the relational structure of the document graph.

3 Method: LP-RAG

This section introduces LP-RAG, a novel link prediction–based framework for RAG. We begin with a brief overview of the document RAG task, and then Sections 3.1 – 3.4 describe each component of LP-RAG. Moreover, Section 3.5 introduces a fast variant of LP-RAG, which employs a named entity recognition (NER) module to build a compact entity-centric graph (i.e., knowledge graph). An example of the LP-RAG process can be found in Appendix D.

Notation. A corpus is a collection of documents $\mathcal{D} = \{D_1, \ldots, D_{|\mathcal{D}|}\}$, where each document D_i is a sequence of tokens. A *chunk* is commonly defined as a contiguous subsequence of tokens within

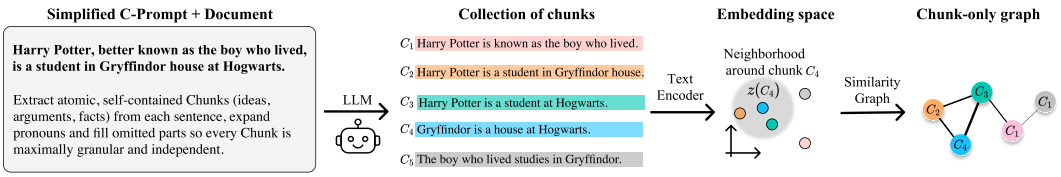


Figure 2: **Chunking**. LP-RAG decomposes a document into chunks via a prompted LLM. Then, we compute embeddings for each chunk node using, e.g., a BERT-based encoder. Next, we calculate pairwise cosine similarity scores between all chunk-node embeddings. Finally, each chunk node is connected to its top-k most similar neighbors, as given by these similarity scores.

a document. Here, we adopt a more general notion that also subsumes named entities. Moreover, we represent an *attributed undirected graph* as a tuple $G = (V, E, z, l)$, where V is a set of vertices, E is a set of unordered pairs of vertices, called edges, $z : V \to \mathbb{R}^d$ is a function that assigns a feature vector $z(v)$ to each vertex $v \in V$, and $l : E \to \mathbb{R}$ assigns edge weights. The set of neighbors of a node v is denoted by $\mathcal{N}^G(v) = \{u \in V : \{v, u\} \in E\}$. Hereafter, we denote the features of v by z_v, and the weight of an edge $\{u, v\}$ as $l_{u,v}$.

Document RAG tasks. Let $\mathcal{C}$ denote the collection of all chunks. Given a user query q, a corpus $\mathcal{D}$, and a pre-trained language model $\mathrm{LLM}(\cdot)$, we want to find a retrieval function RETRIEVE, with

$$\mathcal{C}_q = \textsc{Retrieve}(q, \mathcal{D}) \subseteq \mathcal{C}, \tag{1}$$

such that the retrieved chunks $\mathcal{C}_q$ are relevant to q. The notion of relevance is broad and task-dependent. Here, we are particularly interested in retrieval functions for which the combination of q and $\mathcal{C}_q$, denoted by $\textsc{Combine}(q, \mathcal{C}_q)$, maximizes the expected utility of the LLM answers

$$y_q \sim \mathrm{LLM}\big(\textsc{Combine}(q, \mathcal{C}_q)\big). \tag{2}$$

The notion of utility is task-dependent, e.g., semantic equivalence or exact match for Q&A. Equations 1 and 2 denote the retrieval and generation phases of RAG systems.

3.1 Chunk extraction

We carry out chunking via a prompted LLM. The intuition is that leveraging LLM-based chunking can mitigate truncated content and per-chunk ambiguity while obviating manual tuning of chunk length. It may also alleviate overlap issues commonly observed in heuristic approaches (e.g., TextSplit, SemanticSplit, and RecursiveSplit) [32, 52]. In particular, let C-PROMPT denote a fixed instruction prompt (for details, see Appendix C). We apply an LLM, conditioned on C-PROMPT and the corpus $\mathcal{D}$, to obtain the collection of chunks, i.e., the chunk set is $\mathcal{C} = \mathrm{LLM}\big(\text{C-PROMPT} \,\|\, \mathcal{D}\big)$. An illustration of this procedure is given in Figure 2.

3.2 Synthetic query generation

We prompt an LLM to generate synthetic queries conditioned on extracted chunks/entities. The goal is to produce queries that closely resemble those posed by human users.

We employ a batched generation strategy. Let $\mathcal{B} \subseteq \mathcal{C}$ be a batch of chunks. For each batch, we prompt an LLM to generate synthetic queries whose answers can be derived from the chunks in $\mathcal{B}$. Using a fixed prompt, S-PROMPT (see Appendix C), we generate up to $n_{\mathcal{B}}$ queries per batch:

$$\mathcal{S}_{\mathcal{B}} \sim \mathrm{LLM}(\text{S-PROMPT} \,\|\, \mathcal{B}; n_{\mathcal{B}}). \tag{3}$$

Each synthetic query $S \in \mathcal{S}_{\mathcal{B}}$ is linked with all chunks in its batch $\mathcal{B}$. Let $\mathcal{S}$ denote the set of all generated queries. We encode chunk–query associations via the relation $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{C}$, where $(S, C) \in \mathcal{A}$ iff $S \in \mathcal{S}_{\mathcal{B}}$ and $C \in \mathcal{B}$. Details of the batch construction procedure are provided in Section A.2.

3.3 Graph construction

Given the chunk set $\mathcal{C}$ and the synthetic query set $\mathcal{S}$, we build a graph $G = (V, E, z, l)$ with node set $V = \mathcal{C} \cup \mathcal{S}$, node-feature function z given by a pre-trained contextual encoder (e.g., Contriever) [22], and edge set $E = E_{\mathcal{S}} \cup E_{\mathcal{C}}$ and weighting function l defined as follows.

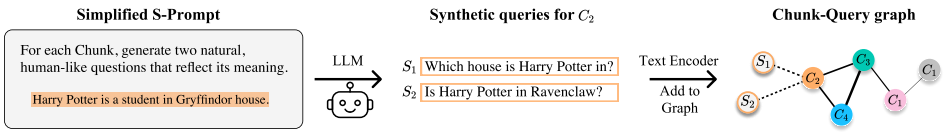


Figure 3: **Synthetic queries and chunk-query graphs**. For each chunk (batch), we create synthetic queries via LLMs. We convert these queries into embeddings using a text encoder (e.g., Contriever) and add them to the chunk graph, connecting them to their respective chunks.

Query-chunk edges $E_{\mathcal{S}}$. Let $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{C}$ denote the query-chunk relation (Section 3.2). We add an edge between a query S and a chunk C iff $(S, C) \in \mathcal{A}$, i.e., $\{S, C\} \in E_{\mathcal{S}} \Longleftrightarrow (S, C) \in \mathcal{A}$.

Chunk-chunk edges $E_{\mathcal{C}}$. Let $\mathrm{sim} : \mathbb{R}^d \times \mathbb{R}^d \to [-1, 1]$ be a similarity metric (e.g., cosine similarity) in the embedding space. Also, for $k \in \mathbb{N}$, define the k-NN set of a chunk C as

$$\mathcal{N}_k(C) \;=\; \arg\operatorname{topk}\{\mathrm{sim}\left(z_{C'}, z_C\right) : C' \in \mathcal{C} \setminus \{C\}\}. \tag{4}$$

Then, we connect two chunks C and C' by *mutual* k-NN:

$$\{C, C'\} \in E_{\mathcal{C}} \iff C \in \mathcal{N}_k(C') \;\wedge\; C' \in \mathcal{N}_k(C). \tag{5}$$

Edge weights. For the edge features, we employ

$$l_{u,v} = \begin{cases} \mathrm{sim}(z_u, z_v), & \text{if } \{u, v\} \in E_{\mathcal{C}}, \\ 1, & \text{if } \{u, v\} \in E_{\mathcal{S}}. \end{cases} \tag{6}$$

Note that $E_{\mathcal{C}}$ and $E_{\mathcal{S}}$ are disjoint sets. Also, the formulation in Equation 6 assigns greater weight to query-chunk edges. This design choice aims to compensate for their relatively lower connectivity and ensure that query–chunk interactions have sufficient influence during message passing, particularly under simple aggregation schemes in large graphs. Figure 3 illustrates the generation of synthetic queries.

3.4 Learning to retrieve

We now formulate retrieval as an inductive link prediction task. Let $G' = (V', E')$ be an (unknown) underlying graph and let $G = (V, E)$ with $V \subseteq V'$ and $E \subseteq E'$ be an observed subgraph. In standard link prediction, the goal is to infer missing edges in $E' \setminus E$ among nodes in V. In the *inductive* setting, the predictor must also infer edges involving unseen nodes in $V' \setminus V$. In our context, the goal is to predict links between incoming user queries (treated as unseen nodes) and chunk nodes.

Box 3.4: GNN for Link Prediction

Most GNNs can be described using the message-passing paradigm [12]. They employ a sequence of message-passing steps, where each node v aggregates messages from its neighbors $\mathcal{N}^G(v)$ and use the resulting vector to update its own embedding.

Let $h_v^{(0)} = z_v$ for all $v \in V$. Then, for all nodes v, message-passing GNNs recursively compute

$$m_v^{(\ell)} = f_{\mathrm{agg}}^{(\ell)}\left(\{\!\{(h_u^{(\ell-1)}, h_v^{(\ell-1)}, l_{u,v}) : u \in \mathcal{N}(v)\}\!\}\right) \tag{7}$$

$$h_v^{(\ell)} = f_{\mathrm{upd}}^{(\ell)}(h_v^{(\ell-1)}, m_v^{(\ell)}), \tag{8}$$

where $\{\!\{\cdot\}\!\}$ denotes multisets, f_{agg} is an order-invariant function and f_{upd} is an arbitrary map (often an MLP). The final GNN embeddings, $\{h_v^{(L)}\}_{v \in V}$, can be further processed using feedforward layers. Finally, the score for a link $\{u, v\}$ is given by

$$\phi(u, v) = \mathrm{MLP} \circ \mathrm{Readout}(\{h_v^{(L)}, h_u^{(L)}\}). \tag{9}$$

Training. We consider a parameterized link predictor of the form $\phi_\theta : V \times V \to \mathbb{R}$, with parameters $\theta \in \mathbb{R}^m$. For undirected graphs, we often require $\phi_\theta(u, v) = \phi_\theta(v, u)\ \forall\ u, v$. The graph used for training consists of the query-chunk graph G in Section 3.3, including both query-chunk and chunk-chunk edges. Although our approach is rather general and accommodates arbitrary models, we briefly illustrate GNNs for link prediction in Box 3.4.

We apply the binary cross-entropy (BCE) loss with negative sampling to learn the parameters θ of our predictor ϕ_θ. Let us define $\hat{y}_{u,v} = \sigma(\phi_\theta(u, v))$ where σ denotes the sigmoid logistic function. Then, we want to minimize the loss

$$\mathcal{L}(\theta) = -\sum_{\{u,v\}\in E} (\log \hat{y}_{u,v} + \mathbb{E}_{v'\sim\mathbb{P}_V(u)}[\log(1 - \hat{y}_{u,v'})]),$$

where $\mathbb{P}_V(u)$ is a negative-sampling distribution over V.

Inference. For an unseen query $q \notin V$, we construct a graph $\tilde{G} = (\mathcal{C} \cup \{q\}, E_\mathcal{C} \cup_{C\in\mathcal{C}} \{q, C\}, \tilde{z}, \tilde{l})$ by adding q as a node to the chunk-only graph (no synthetic queries) and connecting q to all chunk nodes $C \in \mathcal{C}$. These edges serve solely to enable message passing between the query and document chunks and are not treated as observed links. Node and edge features (i.e., $\tilde{z}$, and $\tilde{l}$) are defined exactly as in Section 3.3. Then, retrieval reduces to scoring candidate edges between the query q and chunk nodes $C \in \mathcal{C}$ and selecting those with the highest scores via thresholding, that is

$$\mathcal{C}_q = \{\, C \in \mathcal{C} : \phi_\theta(q, C) \geq \tau \,\}, \tag{10}$$

for a threshold τ. The set $\mathcal{C}_q$ comprises the chunk nodes predicted to form links with q. Note that the query node q in $\tilde{G}$ is discarded after inference, ensuring that no query-specific information is persisted.

Integration into LLMs. The final step of our method consists of combining the input user prompt with the retrieved chunks. Given $\mathcal{C}_q$, we sort the chunks in reverse order by their link–prediction scores w.r.t. q, i.e., we obtain a sequence of chunks $C_1, C_2, \ldots, C_{|\mathcal{C}_q|} \in \mathcal{C}_q$ such that $\phi_\theta(q, C_i) \geq \phi_\theta(q, C_{i+1})$ for all i. We then concatenate the query with the sorted chunks to form the LLM input: $\textsc{Combine}(q, \mathcal{C}_q) = q \,\|\, C_1 \,\|\, C_2 \,\|\, \cdots \,\|\, C_{|\mathcal{C}_q|}$. Appendix C shows the exact prompt sent to the LLM.

3.5 Fast LP-RAG

A potential bottleneck of LP-RAG stems from the use of LLMs to extract atomic chunks from documents, which can increase both indexing time and token usage. To mitigate that, inspired by prior methods [61, 66], here we introduce a lightweight variant of LP-RAG. Rather than constructing chunk-chunk graphs, the idea is to operate on compact entity-centric graphs while preserving the overall LP-RAG framework.

Specifically, Fast LP-RAG employs a BERT-based NER model to extract entities (nodes) and their relations (edges) from documents; identical entity mentions across passages and documents map to one node, which is what makes the graph compact. The synthetic query generation procedure is adapted accordingly: for each entity, we apply Personalized PageRank [19] with randomly sampled neighborhood sizes between 1 and 10 hops to obtain batches of related entities. This stochastic neighborhood expansion captures both local and multi-hop contextual structures. For instance, if a walk returns (Messi, played for, Barcelona) and (Barcelona, located in, Spain), a generated query could be "In which country is the football club Messi played for based?" — the relations are what make such multi-hop questions possible. This is the entity-graph analogue of the chunk batches. The remainder of the pipeline remains unchanged. The resulting query-entity graph (without relation types) is used to train the link predictor, and, during inference, incoming queries are connected to the entity graph for retrieval.

For indexing, we store each entity with its id and graph node, and each relation with the ids it connects; given the retrieved entities we recover the relations among them via those ids, producing (entity, relation, entity) triples, deduplicated so each appears once. In Fast LP-RAG, the LLM context is exactly this triple set plus the query and instruction – a compact relational summary rather than raw passages.

4 Experiments

This section evaluates LP-RAG on widely used benchmarks under several experimental settings. Our evaluation is structured to analyze performance across different LLM settings, data scales, and benchmark protocols. In particular, we consider (i) comparisons against state-of-the-art multi-hop RAG systems using standardized evaluation protocols, (ii) large-scale evaluations with private LLMs on Arena Benchmarks, and (iii) controlled experiments with open-source LLMs. In addition, we conduct ablation studies to quantify the contribution of key design choices, including the choice of link predictor, the use of synthetic queries, the impact of chunking strategy, and different negative sampling strategies. Implementation details are provided in Appendix A, and prompts are reported in Appendix C.

4.1 Experiment I: Comparison against State-of-the-Art Multi-Hop Retrieval Methods

We first evaluate LP-RAG against recent state-of-the-art RAG methods on widely adopted multi-hop QA benchmarks following the protocol of Jimenez Gutierrez et al. [24] and Guo et al. [14]. We consider three multi-hop QA datasets: HotpotQA [63], MuSiQue [56], and 2WikiMultiHopQA (2Wiki) [21]. The evaluation protocol enforces strict data separation to avoid leakage and reports both retrieval and end-to-end QA performance. Retrieval is evaluated using recall@2 (R@2) and recall@5 (R@5), while QA performance is measured using exact match (EM) and F1 scores. We compare LP-RAG against the following baselines: Contriever [22], ColBERTv2 [49], RAPTOR [50], GraphRAG (MS) [8], LightRAG [14], HippoRAG [24], G-retriever [20], GFM-RAG [40], LinearRAG [66] and HippoRAGv2 [15]. We employ `GPT-4o-mini` and adopt the NCN model [59] as the link predictor, and Contriever [22] as the embedding model for all texts.

Table 1: Retrieval performance. LP-RAG outperforms all baselines in four scenarios, and is second best in the remaining ones.

Method	HotpotQA		MuSiQue		2Wiki	
	R@2	R@5	R@2	R@5	R@2	R@5
Contriever	57.2	75.5	34.8	46.6	56.0	57.5
ColBERTv2	64.7	79.3	37.9	49.2	59.2	68.2
RAPTOR	58.1	75.9	35.7	49.3	56.9	63.0
GraphRAG	58.3	76.6	35.4	49.3	61.6	77.3
LightRAG	38.8	54.7	31.4	43.9	51.0	67.9
HippoRAG	60.5	77.9	40.2	53.2	73.5	90.4
G-retriever	53.3	65.5	38.8	45.1	60.8	67.5
GFM-RAG	78.3	87.1	49.1	58.2	**90.8**	**95.6**
HippoRAGv2	80.5	95.7	53.5	74.2	74.6	90.2
LinearRAG	62.9	76.7	42.7	54.1	74.2	90.5
LP-RAG	77.9	89.6	53.6	61.8	89.9	93.2
Fast LP-RAG	**81.7**	**96.6**	**54.9**	**74.9**	90.4	93.8

Results: retrieval performance. As shown in Table 1, Fast LP-RAG achieves the best performance on four scenarios, outperforming HippoRAGv2 by 1.4% in R@2 and 0.7% in R@5 on MuSiQue. Moreover, LP-RAG outperforms all competing baselines by a large margin. Notably, compared to LinearRAG, Fast LP-RAG achieves a significant improvement of 16.2% in R@2 on HotpotQA.

Results: Q&A performance. Table 2 depicts the Q&A results. Notably, LP-RAG achieves state-of-the-art performance against all baselines. In particular, LP-RAG outperforms HippoRAGv2 by 1.6%, 2.4%, 13.3% in EM on HotpotQA, MuSiQue, and 2Wiki, respectively.

Table 2: Results: Q&A performance. Fast LP-RAG is the best method in both metrics for all datasets.

Method	HotpotQA		MuSiQue		2Wiki	
	EM	F1	EM	F1	EM	F1
None	30.4	42.8	12.5	24.1	31.0	39.0
ColBERTv2	43.4	57.7	15.5	26.4	33.4	43.3
GraphRAG	35.3	54.6	13.6	24.9	28.3	38.4
LightRAG	36.8	48.3	18.1	27.5	45.1	49.5
HippoRAG	41.8	55.0	19.2	29.8	46.6	59.5
GFM–RAG	51.6	66.9	30.2	40.4	69.8	77.7
HippoRAGv2	56.3	71.1	35.0	49.3	60.5	69.7
LinearRAG	44.5	61.6	21.3	33.9	46.3	61.8
LP–RAG	53.2	69.1	32.6	43.9	72.6	78.3
Fast LP-RAG	**57.9**	**73.3**	**37.4**	**52.1**	**73.8**	**78.6**

Table 3: Results on the Arena datasets. Fast LP-RAG surpasses the performance of the baselines in 3 out of 4 datasets. Best values are in bold.

Methods	Writing		Tech		Science		Recreation	
	W+T ↑	Token↓	W+T ↑	Token↓	W+T ↑	Token↓	W+T ↑	Token↓
NaiveRAG	0.663	9.4k	0.689	9.1k	0.526	9.0k	0.720	9.1k
HyDe	0.789	9.6k	0.863	9.3k	0.823	9.3k	0.777	9.3k
LightRAG	0.754	6.3k	0.937	6.9k	0.840	7.1k	0.800	6.8k
GraphRAG	0.749	6.4k	0.943	6.7k	0.863	6.7k	0.806	6.8k
NodeRAG	**0.794**	3.3k	0.949	3.8k	0.903	4.2k	0.886	4.3k
HippoRAGv2	0.748	3.7k	0.942	3.4k	0.859	3.5k	0.841	3.6k
LinearRAG	0.759	3.6k	0.914	3.3k	0.876	3.3k	0.866	3.6k
LP-RAG	0.775	4.2k	0.949	4.8k	0.920	4.1k	0.907	4.1k
Fast LP-RAG	0.793	3.2k	**0.953**	3.2k	**0.941**	3.1k	**0.926**	3.1k

4.2 Experiment II: Large-scale Evaluation

We now evaluate LP-RAG in a large-scale setting using `GPT-4o-mini` and the Arena benchmarks [17]. We follow the evaluation setup of Xu et al. [61] and compare against NaiveRAG [33], HyDe [10], GraphRAG [8], LightRAG [14], NodeRAG [61], HippoRAGv2 [15], and LinearRAG [66].

For fairness, we use the exact data splits of Xu et al. [61] and consider four domains: Writing, Recreation, Science, and Tech, which keeps evaluation costs manageable. Except for HippoRAGv2 and LinearRAG, baseline results are taken directly from their work. Consistent with their evaluation, we measure token usage for efficiency and report the win–tie ratio (W+T). Again, for LP-RAG, we again use NCN as link predictor and Contriever as text encoder.

Results. Table 3 reports the results on the Arena benchmarks. Fast LP-RAG attains the highest win/tie ratio in the Tech, Science and Recreation domains, with particularly large gains in Recreation, where it exceeds NodeRAG by 0.04. On Writing, LP-RAG does not surpass NodeRAG, but it still improves over the others baselines while using substantially fewer tokens. Overall, LP-RAG's token usage remains less or comparable to NodeRAG across all domains.

Table 4: Efficiency of Fast LP-RAG (LP), HippoRAGv2 (H) and LinearRAG (L) on Arena datasets.

Dataset	Corpus size	Index (min)			Storage (MB)			Avg Query (s)			Avg tokens (T)		
		H	L	LP	H	L	LP	H	L	LP	H	L	LP
A. Writing	1.82M	54	36	68	234	221	278	2.40	2.63	2.15	3782	3681	3261
A. Tech	1.72M	49	32	65	213	207	236	2.23	2.41	2.43	3419	3316	3216
A. Science	1.43M	46	31	58	208	198	223	2.61	2.67	2.16	3509	3365	3167
A. Recreation	0.93M	34	24	53	174	126	187	2.26	2.42	2.76	3635	3631	3106

Efficiency Analysis. We compare the computational performance of Fast LP-RAG (LP) with the main graph-based methods in Section 4.2, i.e., HippoRAGv2 (H) and LinearRAG (L). Our comparison covers indexing time, storage usage, average query latency, and average number of retrieved tokens (see Table 4). Importantly, all experiments were conducted under the same hardware and setup to ensure a fair comparison. Overall, our method demonstrates equal or superior performance across all four datasets. In terms of query time, it is the fastest method on 2 out of 4 datasets, indicating that the overhead introduced by the GNN-based predictor is offset by the significantly lower number of retrieved tokens. Also, the indexing time is comparable with that of HippoRAGv2.

4.3 Experiment III: Controlled Evaluation with Open-Source LLMs

We now consider smaller versions of five benchmarks: HotpotQA [63], MuSiQue [56], MultiHop-RAG [55], RAG-QA [17], and ICLR Papers [9]. Following Xu et al. [61], we compute macro accuracy, F1 score, and measure the average number of retrieved tokens to evaluate efficiency. Since cost is not a concern here, we only consider the standard variant of our framework (LP-RAG) in this section.

We conduct a controlled experiment using `Llama-3.3 70B` and sampling 20 documents per dataset — except for ICLR Papers, for which we sample 50 documents. To distinguish the reduced datasets from their original counterparts, we append the suffix "-S" to the smaller versions. This reduced setting enabled us to (1) re-run baselines and perform ablation studies, and (2) compute F1 scores (not available in Xu et al. [61]), ensuring a fair and reliable comparison. We use NCN and Contriever as embedding model. For completeness, we also report results using full datasets in Appendix B.

Results. Table 5 shows the results. Across all datasets, LP-RAG significantly outperforms all baselines: for the HotpotQA-S dataset, it achieves a 22% accuracy advantage over NaiveRAG, and a 2.42% over NodeRAG. Furthermore, its cost is lower than that of all baselines across datasets, with particularly large savings on the ICLR dataset. Regarding similarity-based baselines (i.e., NaiveRAG, HyDe, and GraphRAG), they generally achieve lower accuracy and macro F1 scores, even when they use graphs to capture relational structure. This behavior is evidenced in the GraphRAG results where accuracy is lower than NaiveRAG in the MuSiQue-S dataset. These findings suggest that these similarity-based methods struggle to identify the most informative nodes for generation, reinforcing the benefits of LP-RAG's link-prediction-based retrieval mechanism. We hypothesize that LP-RAG achieves the lowest token consumption for two reasons. First, our chunk extractor produces short, atomic chunks. Second, retrieval is based on the probability of an edge existing in a chunk rather than a fixed top-k selection. Together, these factors yield higher-quality information and more efficient retrieval. Overall, our results suggest that link prediction methods can improve retrieval performance in LLM-based question-answering tasks.

Table 5: Q&A results: LP-RAG outperforms all baselines in all datasets while utilizing minimum resources. Bold denotes the best results.

Methods	HotpotQA-S			MuSiQue-S			MultiHop-S			RAG-QA-S			ICLR-S		
	Acc.↑	F1↑	Tok↓	Acc.↑	F1↑	Tok↓	Acc.↑	F1↑	Tok↓	Acc.↑	F1↑	Tok↓	Acc.↑	F1↑	Tok↓
NaiveRAG	67.50	22.50	7.2k	69.43	20.30	7.1k	68.23	21.14	8.1k	67.08	25.50	8.2k	53.90	21.62	7.9k
HyDe	72.17	36.95	7.3k	69.43	34.60	6.7k	73.89	37.59	8.3k	54.96	38.63	8.4k	68.50	35.30	7.2k
GraphRAG	87.24	65.00	6.1k	61.71	42.00	5.8k	72.15	58.62	6.4k	65.96	43.60	6.4k	74.50	42.00	6.9k
LightRAG	79.83	65.00	5.4k	63.14	41.50	5.3k	70.63	48.95	6.7k	65.89	43.62	6.3k	72.40	43.96	6.9k
NodeRAG	86.90	69.50	4.0k	66.00	45.00	5.2k	70.86	54.50	6.0k	67.20	56.86	3.3k	77.92	54.82	6.2k
LP-RAG	**89.32**	**86.29**	**3.2k**	**89.67**	**85.15**	**3.8k**	**89.12**	**86.35**	**3.8k**	**87.89**	**85.26**	**2.2k**	**91.16**	**89.51**	**2.4k**

4.4 Ablation study

We now conduct an ablation study using the same five datasets and evaluation metrics as in Table 5. We begin by evaluating the contribution of the GNN-based retrieval module, then examine the effect of incorporating synthetic queries, and finally assess how different negative sampling strategies influence LP-RAG's performance.

Table 6: Ablation study concerning the performance of different GNNs and the importance of synthetic queries. Here, $*$ denotes models trained using only chunk nodes (no synthetic queries). Overall, NCN is the best performing model, and using synthetic queries is crucial.

GNN	HotpotQA-S			MuSiQue-S			MultiHop-S			RAG-QA-S			ICLR-S		
	Acc.↑	F1.↑	Tok↓	Acc.↑	F1.↑	Tok↓	Acc.↑	F1.↑	Tok↓	Acc.↑	F1.↑	Tok↓	Acc.↑	F1.↑	Tok↓
SEAL	89.28	85.12	3.2k	89.41	84.05	3.5k	89.03	87.62	3.7k	87.73	83.13	2.2k	89.14	88.23	2.5k
PEG	87.01	86.11	3.3k	85.22	83.16	3.5k	87.64	85.63	3.8k	83.14	82.63	2.2k	84.37	82.09	2.6k
VGAE	82.03	80.43	3.2k	84.63	80.50	3.7k	84.07	81.64	3.7k	83.29	80.19	2.4k	86.21	79.53	2.6k
NCN	89.32	86.29	3.2k	89.67	85.15	3.8k	89.12	86.35	3.8k	87.89	85.26	2.2k	91.16	89.51	2.4k
MLP	32.50	12.25	5.3k	43.91	22.50	5.6k	29.80	16.24	5.7k	42.58	16.80	5.7k	48.90	23.78	5.3k
VGAE*	60.50	42.82	3.9k	50.92	41.63	3.9k	40.06	22.16	4.1k	60.15	37.16	2.8k	60.15	48.60	2.8k
NCN*	69.15	42.80	3.9k	68.64	55.20	4.0k	52.15	48.05	3.8k	67.15	45.18	2.4k	72.60	48.95	2.9k

Impact of GNN architecture. We consider four popular GNNs for link prediction: VGAE [31], PEG [57], SEAL [64], and NCN [59]. For completeness, we also include results using an MLP (multilayer perceptron). The results are given in Table 6. Overall, the results confirm that a robust GNN with stronger negative link discrimination is essential for producing higher accuracy. In contrast, an MLP, which lacks the capacity to model relationships between entities, tends to underperform even compared to NaiveRAG. Another notable result is that VGAE performs comparably to NodeRAG, even though VGAE is a much less simpler model than NCN. This suggests that even simpler GNNs can achieve high-quality chunk retrieval when trained using our approach.

Impact of synthetic queries. Our second experiment (also reported in Table 6) evaluates the impact of synthetic queries on retrieval quality. Here, the GNNs architectures were trained on graphs containing only interconnected chunks, without synthetic queries. In this setting, even the strongest GNN (i.e., NCN) performs similarly to NaiveRAG, confirming the importance of augmenting graph information with synthetic queries in our framework.

Impact of negative sampling. Recent works [1] have discussed issues related uniform negative sampling in link prediction. To assess the impact of negative sampling, we compare LP-RAG using uniform vs. degree-based negative sampling [1], under the setup of EXPERIMENT III. As we observe, modern negative-sampling strategies improve the performance of LP-RAG in some cases. However, despite these improvements, their overall impact is considerably smaller than the effect of the GNN architecture. We hypothesize that this is likely due to the approximately regular structure of our query–chunk graph, which results from the use of a k-NN construction.

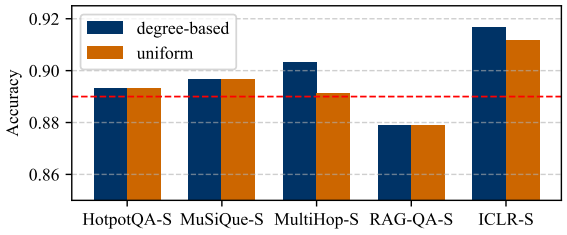


Figure 4: Impact of negative sampling on Q&A results.

Additional results. In Appendix B, we report further experiments regarding i) the impact of chunking scheme (Section B.2); ii) the generalization of LP-RAG to unseen documents and cross-domain

transferability (Section B.4); iii) the effectiveness of LP-RAG with open-source LLM on full datasets (Section B.5); and iv) the impact of smaller models for generating chunks and queries (Section B.7).

5 Conclusions

In this paper, we propose a novel graph-based RAG framework based on link prediction. Crucially, our method relies on synthetic queries to obtain supervision signals for the link prediction GNN. Intuitively, these queries enrich the graph with fine-grained semantic cues, guiding retrieval toward passages that are not only structurally connected but also semantically relevant. Retrieval tasks are formulated as inductive link prediction over a chunk–query graph. Through extensive experiments on multiple datasets, we demonstrate that LP-RAG not only achieves a significant improvement in accuracy and F1-score compared to multiple baselines but also operates with little to moderate resources. Overall, LP-RAG demonstrates that integrating synthetic queries into RAG pipelines can significantly boost the quality of information retrieval and performance on question-answering tasks.

Limitations. Due to financial and computational constraints, in this work, we do not evaluate LP-RAG as a large-scale pretrained (foundation) approach. Nevertheless, we provide evidence that LP-RAG can outperform graph foundation models (i.e., GFM-RAG) when trained on the same data (Table 14). In addition, we note that the effectiveness of LP-RAG depends on the quality of the synthetic query generation and chunking prompts, which may introduce sensitivity to prompt design and LLM generation quality.

Societal Impact. We foresee no immediate societal or ethical concerns arising from this work.

6 Acknowledgements

This work was carried out with the support of FUNCAP through Call for Proposals No. 08/2023 – FUNCAP – Internationalization, Project ITR-0214-00224.01.00/26. We also acknowledge the support of the Núcleo de Visão Computacional e Engenharia (NUVEN), the Polo de Inovação at Instituto Federal do Ceará (Polo IFCE), the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (grants 408974/2025-7, 312068/2025-5), the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and the Empresa Brasileira de Pesquisa e Inovação Industrial (Embrapii).

References

[1] Rachith Aiyappa, Xin Wang, Munjung Kim, Ozgur Can Seckin, Jisung Yoon, Yong-Yeol Ahn, and Sadamori Kojaku. Implicit degree bias in the link prediction task. In *International Conference on Machine Learning*, 2025.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Tianchi Cai, Zhiwen Tan, Xierui Song, Tao Sun, Jiyan Jiang, Yunqi Xu, Yinger Zhang, and Jinjie Gu. Forag: Factuality-optimized retrieval augmented generation for web-enhanced long-form question answering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 199–210, 2024.

[4] Aditi Chaudhary, Karthik Raman, and Michael Bendersky. It's all relative!–a synthetic query generation approach for improving zero-shot relevance prediction. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1645–1664, 2024.

[5] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

[6] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense x retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, 2024.

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[8] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[9] Tao Feng, Yihang Sun, and Jiaxuan You. Grapheval: A lightweight graph-based llm framework for idea evaluation. In *International Conference on Learning Representations*, volume 2025, pages 99181–99209, 2025.

[10] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, 2023.

[11] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[13] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. Doc2query–: when less is more. In *European Conference on Information Retrieval*, pages 414–422. Springer, 2023.

[14] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. In *The Thirteenth Conference on Empirical Methods in Natural Language Processing [EMNLP 2025]*, 2025.

[15] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning*, pages 21497–21515. PMLR, 2025.

[16] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

[17] Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. Rag-qa arena: Evaluating domain robustness for long-form retrieval augmented question answering. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 4354–4374, 2024.

[18] Zhen Han, Anand Muralidhar, and Aditya Degala. Graphrag: Reasoning on graphs with retrieval-augmented llms. In *NeurIPS Workshop on New Frontiers in Graph Learning (GLFrontiers)*, 2023.

[19] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526, 2002.

[20] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907, 2024.

[21] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[22] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022.

[23] Xinke Jiang, Rihong Qiu, Yongxin Xu, Wentao Zhang, Yichen Zhu, Ruizhe Zhang, Yuchen Fang, Xu Chu, Junfeng Zhao, and Yasha Wang. Ragraph: A general retrieval-augmented graph learning framework. In *NeurIPS 2024*, 2024.

[24] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 2024.

[25] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International conference on machine learning*, pages 15696–15707. PMLR, 2023.

[26] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[27] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, November 2020. Association for Computational Linguistics.

[28] Savini Kashmira, Jayanaka L. Dantanarayana, Joshua Brodsky, Ashish Mahendra, Yiping Kang, Krisztian Flautner, Lingjia Tang, and Jason Mars. Tobugraph: Knowledge graph-based retrieval for enhanced llm performance beyond rag. *arXiv preprint arXiv:2412.05447*, 2024.

[29] Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae Won Park. Mdagents: an adaptive collaboration of llms for medical decision-making. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[31] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

[32] Urho Lensu. Impact of chunking granularity on accuracy and token consumption in retrieval-augmented generation for question-answering. *Master's thesis Aalto Fi*, 2025.

[33] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[34] Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[35] Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. Graph neural network enhanced retrieval for question answering of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6612–6633, 2025.

[36] Davis Liang, Peng Xu, Siamak Shakeri, Cicero Nogueira dos Santos, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. Embedding-based zero-shot retrieval through query generation. *arXiv preprint arXiv:2009.10270*, 2020.

[37] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Conference on Neural Information Processing Systems*, 2023.

[38] Jiayu Liu, Zhenya Huang, Tong Xiao, Jing Sha, Jinze Wu, Qi Liu, Shijin Wang, and Enhong Chen. Socraticlm: Exploring socratic personalized teaching with large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[39] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024.

[40] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. Gfm-rag: graph foundation model for retrieval augmented generation. In *The thirty-ninth Conference on Neural Information Processing Systems*, 2025.

[41] Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699, 2025.

[42] Gabriel de Souza P Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. Nv-retriever: Improving text embedding models with effective hard-negative mining. *arXiv preprint arXiv:2407.15831*, 2024.

[43] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, 2022.

[44] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. Secom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Representation Learning*, 2025.

[45] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024.

[46] Vatsal Raina and Mark Gales. Question-based retrieval using atomic units for enterprise rag. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 219–233, 2024.

[47] Stephen E Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer, 1994.

[48] Sonal Sannigrahi, Thiago Fraga-Silva, Youssef Oualil, and Christophe Van Gysel. Synthetic query generation using large language models for virtual assistants. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2837–2841, 2024.

[49] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, 2022.

[50] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024.

[51] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[52] Maximilian Stäbler, Steffen Turnbull, Tobias Müller, Chris Langdon, Jorge Marx-Goméz, and Frank Köster. The impact of chunking strategies on domain-specific information retrieval in rag systems. In *2025 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6, 2025. doi: 10.1109/COINS65080.2025.11125724.

[53] W Su, Y Tang, Q Ai, Z Wu, and Y Liu. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. arxiv 2024. *arXiv preprint arXiv:2403.10081*, 2024.

[54] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[55] Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*, 2024.

[56] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[57] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022.

[58] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, 2023.

[59] Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[60] Zhenbang Wu, Anant Dadu, Michael Nalls, Faraz Faghri, and Jimeng Sun. Instruction tuning large language models to understand electronic health records. In *Advances in Neural Information Processing Systems 37: Datasets and Benchmarks Track*, 2024.

[61] Tianyang Xu, Haojie Zheng, Chengze Li, Haoxiang Chen, Yixin Liu, Ruoxi Chen, and Lichao Sun. Noderag: Structuring graph-based rag with heterogeneous nodes. *arXiv preprint arXiv:2504.11544*, 2025.

[62] Rui Yang, Boming Yang, Xinjie Zhao, Fan Gao, Aosong Feng, Sixun Ouyang, Moritz Blum, Tianwei She, Yuang Jiang, Freddy Lecue, Jinghui Lu, and Irene Li. Graphusion: A rag framework for scientific knowledge graph construction with a global perspective. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, page 2579–2588, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713316.

[63] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[64] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[65] Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8912–8924, 2025.

[66] Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. In *International Conference on Learning Representations*, volume 2026, pages 147053–147075, 2026.

A Experimental details

A.1 Datasets and baselines

Datasets. We consider seven benchmarks:

- *HotpotQA* [63] is a multi-hop question-answering dataset where each question requires combining information from multiple documents to find the correct answer.
- *MuSiQue* [56] is also a multi-hop question-answering dataset that challenges models to combine information across multiple documents in a structured, step-by-step manner.
- *MultiHop-RAG* [55] is a multi-hop question-answering dataset that includes four distinct question types: comparison query, null query, inference query, and temporal query.
- *Arena benchmark* [17] builds on Long-form RobustQA (LFRQA), a dataset of 26K queries across 7 domains. Here, we consider the Writing, Tech, Science, and Recreation domains.
- *RAG-QA* [17] is a shortened version of the Arena Benchmark, where we selected samples from 7 domains.
- *ICLR Papers* [9] is formed by abstracts and review decisions from paper submissions to the ICLR conferences between 2021 and 2023. We use an LLM to generate questions and answers to tailor it to the Q&A task.
- *2WikiMultiHopQA (2Wiki)* [21] is a multi-hop QA dataset that requires reasoning over multiple Wikipedia articles to answer questions. The dataset consists of 192k questions, which are designed to be answerable using information from 2 or 4 articles.

Baselines. In the following, we provide a brief overview of the baselines considered in the paper.

- *NaiveRAG* [33] represents a simple baseline for RAG systems. It first divides documents into several text chunks and encode them into a vector space utilizing text embeddings. Then retrieve related text chunks based on similarity of query representations.
- *HyDe* [10] improves upon traditional RAG systems. In particular, HyDe uses an LLM to generate "synthetic answers" from the input query. These "synthetic answers" are transformed into embeddings and used to search for the most similar chunks.
- *GraphRAG* [8] segments the input document into chunks and extracts the entities and their relationships to create a graph. This graph is then divided into multiple communities at different levels. At query time, GraphRAG identifies the relevant entities from the question and synthesizes answers by referencing these corresponding community summaries.

- *LightRAG* [14] is an improved approach based on GraphRAG, designed to minimize computational overhead while enhancing the comprehensiveness of retrieved information through dual-level retrieval. This leads to more efficient retrieval and a better balance between effectiveness and speed compared to GraphRAG.
- *NodeRAG* [61] is a graph-centric approach that leverages heterogeneous graphs.
- *BM25* [47] is a classic information retrieval method based on the probabilistic model that ranks a set of documents based on the query terms frequency appearing in each document.
- *Contriever* [22] trains a dense retriever with contrastive learning on a large-scale corpus to retrieve relevant documents for a given query.
- *GTR* [43] develops a scale-up T5-based dense retriever that could generalize across different datasets and domains.
- *ColBERTv2* [49] is a state-of-the-art dense retriever that couples an aggressive residual compression mechanism with a denoised supervision strategy to simultaneously improve the retrieval quality.
- *RAPTOR* [50] is an LLM-augmented retriever that recursively embeds, clusters, and summarizes chunks of text, constructing a tree with differing levels of summarization to enable accurate retrieval.
- *Proposition* [6] enhances the performance of dense retrievers by leveraging LLMs to generate a natural language proposition that captures the essential information of the document.
- *HippoRAG* [24] is a state-of-the-art, training-free graph-enhanced retriever that uses the Personalized PageRank algorithm to assess entity relevance to a query and performs multihop retrieval on a document-based knowledge graph. It can be directly applied to various datasets.
- *GFM-RAG* [40] uses a Graph Foundation Model (GFM) to reason over a knowledge-graph index built from documents. The model is pretrained at scale (60 KGs, 14M triples + 700k documents).
- *SubgraphRAG* [34] retrieves relevant subgraphs from a knowledge graph (KG) using a lightweight multilayer perceptron and a parallel triple-scoring mechanism, while encoding directional structural distances to improve retrieval effectiveness. Then, those subgraphs (triples) are fed into an LLM, which reasons over them to generate grounded and explainable answers.
- *G-retriever* [20] works by: (1) indexing graph nodes and edges using embeddings; (2) retrieving the most relevant elements based on similarity when given a query; (3) constructing a relevant subgraph by formulating it as a Steiner Tree problem to avoid context explosion; and (4) using an LLM to generate an answer with soft prompt tuning based on that subgraph.
- *LinearRAG* [66] builds a lightweight, relation-free hierarchical structure called a Tri-Graph using only entity extraction and semantic linking, which scales linearly and avoids additional token costs. During retrieval, it employs a two-stage process: first activating relevant entities through local semantic connections, and then selecting passages via global importance aggregation.
- HippoRAGv2 [15] is designed to better approximate human-like long-term memory by overcoming limitations of standard vector-based retrieval. It enhances prior graph-augmented approaches by integrating deeper passage-level information and leveraging more effective online use of LLMs, while building on a Personalized PageRank mechanism for knowledge propagation.

A.2 Batch generation

We generate batches of chunks based on the chunk-only graph (see Section 3.3). Specifically, for each target node $v \in V$, we first uniformly sample the batch size $k_v \sim \text{Uniform}(\{0, 1, 2, \ldots, |\mathcal{N}(v)|\})$, and then uniformly select k_v (connected) nodes from $\mathcal{N}(v)$. For instance, if $k_v = 2$ and u, w are sampled from $\mathcal{N}(v)$, then we form the batch $\mathcal{B} = \{v, u, w\}$. Note that we repeat this procedure for all target nodes v (chunks) in the chunk graph.

In addition, in this work, we employ $n_{\mathcal{B}} = 2$ — i.e., we generate two synthetic queries per batch.

A.3 Details on Experiment I

In this section, we provide more details about the evaluation setup for the experiments reported in Tables [1, and 2], and about the hyperparameters used in all GNNs. To ensure the reproducibility of our work, we will release the code publicly after the review process is complete.

LP-RAG was implemented using PyTorch and PyTorch Geometric, with all experiments conducted on a machine with 32GB RAM, Intel Core I7-13700 30mb 2.5ghz, and a single NVIDIA RTX 2060 super with 8GB.

For EXPERIMENT I, we use NCN as link predictor with uniform negative sampling. In addition, we use `GPT-4o-mini` with temperature 0 and API calls from OpenAI API. We use the same LLM during the indexing phase, i.e., for chunk extraction and synthetic query generation. Regarding the construction of chunk-chunk edges, we apply mutual k-nearest neighbors with $k = 5$.

Table 7: Hyperparameters for NCN used in Experiment I.

dataset	LR	WD	layers	epochs	h-dim
MuSiQue	0.0043	10^{-8}	1	1000	256
HotpotQA	0.0043	10^{-8}	3	1000	256
2wiki	0.0085	10^{-8}	3	1000	32

MuSiQue, HotpotQA, and 2wiki have 6119, 9221, 11656 documents, respectively. Each dataset contains 1000 documents for test, Table 7 shows the hyperparameters for each dataset. To train LP-RAG, we use the same test split reported from Luo et al. [40]. The baseline results reported in Table 1 and Table 2 were collected from [40].

Table 8: Hyperparameters used for the link prediction methods.

GNN	LR	WD	layers	epochs	h-dim
VGAE	10^{-2}	10^{-4}	2	400	32
PEG	10^{-2}	10^{-4}	2	400	32
NCN	0.0043	10^{-8}	1	100	256
SEAL	10^{-4}	10^{-8}	3	50	32
MLP*	10^{-3}	$5 * 10^{-4}$	2	1000	1000

Regarding the configuration of each GNN, Table 8 shows the used hyperparameters. To train the GNNs, we employ the Adam optimizer [30]. We considered chunks with a confidence level greater than 50% ($\tau = 0.5$) assigned by the GNN to generate the response. Finally, For all baselines that have been run, we use the official dataset divisions as chunks, with top-5 retriever. All other settings are those provided by the respective authors.

A.4 Details on Experiment II

For EXPERIMENT II, we also use NCN with the same hyperparameter choice reported in Table 8. We also use the same negative sampling, Adam optimizer, and chunk selection criteria from EXPERIMENT I. Also, we use `GPT-4o-mini` with temperature 0 as LLM with calls to OpenAI API.

For Arena-Writing, we consider 11,250 documents for chunking, and 2,693 questions. For Arena-Science, we consider 5,283 documents and 1,403 questions. For Arena-Tech, we consider 8,438 documents and 2,063 questions. Finally, for Arena-Recreation, we consider 4,341 documents and 2,089 questions. The baseline results reported in Table 3 were collected from [61].

A.5 Details on Experiment III

For EXPERIMENT III, we use `Llama-3.3-70B` with temperature 0 and API calls from Groq API. We use the same LLM during the indexing phase, i.e., for for chunk extraction and synthetic query generation. Regarding the construction of chunk-chunk edges, we apply mutual k-nearest neighbors with $k = 5$.

For MuSiQue, we use the “ans" version. For the other datasets, we use the version available in their respective repositories. We randomly sampled 20 documents from each dataset, with the exception of the ICLR papers, where we sampled 50 documents since it is much smaller.

B Additional experiments

B.1 Efficiency Analysis

We compare the computational performance of LP-RAG with the main graph-based methods in Section 4.3. Our comparison covers indexing time, storage usage, average query latency, and average number of retrieved tokens (see Table 9). Importantly, all experiments were conducted under the same hardware and setup to ensure a fair comparison.

Overall, our method demonstrates equal or superior performance across all five datasets. In terms of query time, it is the fastest method on 4 out of 5 datasets, indicating that the overhead introduced by the GNN-based predictor is offset by the significantly lower number of retrieved tokens. Also, the indexing time is comparable with that of NodeRAG. These results underscore the efficiency of our framework when compared with existing graph-based approaches.

Table 9: LP-RAG performance. Corpus size in total tokens, indexing time in minutes, storage usage in MegaBytes (MB), query time in seconds (s). Light refers to LightRAG and Node refers to NodeRAG.

Dataset	Size	Indexing time (min)			Storage usage (MB)			Avg query time (s)			Avg retrieved tokens		
		Light	Node	LP-RAG	Light	Node	LP-RAG	Light	Node	LP-RAG	Light	Node	LP-RAG
HotpotQA-S	62K	7.9	5.1	4.7	21.2	10.7	10.3	5.8	4.2	3.6	5462	4019	3208
MuSiQue-S	58K	5.6	5.2	5.8	21.6	9.8	10.1	6.2	5.6	5.1	5312	5269	3849
MultiHop-S	49K	5.9	5.3	4.8	19.7	10.4	9.7	7.4	5.9	5.1	6752	6032	3816
RAG-QA-S	54K	6.1	5.8	6.2	26.8	14.0	12.1	5.6	4.8	4.2	6325	3370	2279
ICLR-S	67K	10.3	8.6	7.9	38.7	26.9	15.7	7.9	5.4	5.7	6946	6214	2428

B.2 Impact of chunking strategy

The chunking strategy can be considered a data preprocessing step and therefore can impact the method's performance. In this experiment, we analyzed whether our chunk strategy could benefit other baselines. We used the chunks extracted from the samples used in EXPERIMENT III to feed all the baselines and we report the macro accuracy found.

Table 10: Macro accuracy using our chunk strategy as data preprocessing for the baselines. LP-RAG results are from Table 5.

Method	HotpotQA-S	MuSiQue-S	MultiHop-S	RAG-QA-S	ICLR-S
NaiveRAG	18.33	15.56	16.67	12.78	11.11
HyDe	25.46	21.19	36.50	32.67	37.69
LightRAG	71.25	60.62	64.25	61.20	66.62
NodeRAG	83.31	62.29	70.64	67.05	74.43
LP-RAG	**89.32**	**89.67**	**89.12**	**87.89**	**91.16**

As we can see in Table 10, our chunk strategy does not show a significant gain for the baselines; in fact, it's the opposite. All baselines showed lower accuracy than those reported in Table 5. The most impacted baseline was NaiveRAG, with all results falling below 20%. We believe that our short chunks with atomic information are too small for similarity-based methods to correctly connect them with a query. Larger chunks provide more information, and traditional RAG models can take advantage of this to generate a more accurate response, which is not the case when they use our chunking strategy.

The Table 11 shows the impact of different chunkenizers on LP-RAG. We can see a clear gain in indexing time; however, the quality of the responses is directly and negatively affected. We believe the main driver is the chunk size generated by these traditional methods. With larger chunks, the synthetic query generator tends to generate vaguer queries, since it will try to use all the factual information at once, which degrades the GNN's ability to identify correct passages.

B.3 Generalization to unseen documents

In this section, we evaluate the generalizability of LP-RAG by conducting domain-specific generalization on data not seen during training. For each domain-specific LP-RAG acquired during

Table 11: Analysis of the impact of different chunk methods on LP-RAG using the EXPERIMENT I setup.

chunk	HotpotQA			2Wiki		
	index (min)	R@2	R@5	index (min)	R@2	R@5
recursive	37	59.5	76.1	32	71.4	75.9
semantic	112	65.3	79.8	98	82.6	89.7
Our	287	77.9	89.6	248	89.9	93.2
NER*	35	81.7	96.6	31	90.4	93.8

EXPERIMENT III, we select 50 completely new samples. For this new data, we only construct the chunk graph, without fine-tuning the model. We also re-evaluated the baselines with the same 50 samples. In the end, we calculated the macro accuracy.

Table 12: Generalization to unseen documents.

Method	HotpotQA-S	MuSiQue-S	MultiHop-S	RAG-QA-S	ICLR-S
NaiveRAG	52.00	42.00	38.00	54.00	64.00
HyDe	64.00	68.00	66.00	62.00	70.00
LightRAG	74.00	62.00	68.00	66.00	72.00
NodeRAG	84.00	66.00	74.00	68.00	72.00
LP-RAG	**86.00**	**82.00**	**84.00**	**82.00**	**84.00**

As can be seen in Table 12, LP-RAG is able to generalize to new documents not seen during training. In fact, macro accuracy has a significant decay compared to the results in Table 5, but our method is still superior in all datasets, especially MuSiQue, which had a 16% gain compared to NodeRAG.

B.4 Cross-domain transferability

To demonstrate the transferability of LP-RAG, we consider the retrieval performance (R@2 and R@5) of LP-RAG on two datasets following the setup of Experiment I. In particular, We now take the LP-RAG model trained on the MuSiQue dataset and evaluate it on HotpotQA and 2Wiki.

Table 13: Retrieval performance comparison of LP-RAG trained only using the MuSiQue dataset.

Method	HotpotQA		2Wiki	
	R@2	R@5	R@2	R@5
BM25	55.4	72.2	51.8	61.9
Contriever	57.2	75.5	56.0	57.5
GTR	59.4	73.3	60.2	67.0
ColBERTv2	64.7	79.3	59.2	68.2
RAPTOR	58.1	75.9	56.9	63.0
Proposition	58.7	71.1	56.4	63.4
GraphRAG (MS)	58.3	76.6	61.6	77.3
LightRAG	38.8	54.7	51.0	67.9
HippoRAG (Contriever)	59.0	76.2	71.5	89.5
HippoRAG (ColBERTv2)	60.5	77.9	73.5	90.4
SubgraphRAG	61.5	73.0	63.3	71.7
G-retriever	53.3	65.5	60.8	67.5
GFM-RAG	**68.8**	**81.8**	**84.4**	89.6
LP-RAG	65.4	78.3	75.9	**91.6**

As shown in Table 13, LP-RAG is the best model in 1/4 cases and the second-best in the remaining ones — surpassing models like HippoRAG by a large margin. It is important to note that GFM-RAG is pre-trained using a larger number of documents.

Table 14: Retrieval performance when both methods are trained with all 3 datasets together.

dataset	HotpotQA		MuSiQue		2Wiki	
	R@2	R@5	R@2	R@5	R@2	R@5
GFM-RAG	79.3	87.1	**49.1**	58.2	**90.8**	95.6
LP-RAG	**79.5**	**89.3**	48.3	**59.2**	89.6	**96.7**

The results for LP-RAG and GFM-RAG when pretrained using the training sets of the 3 datasets together and inferred using the separate test sets are shown in Table 14. In these experiments, the test chunks were simply added to the graph, without any new adjustment of the GNN predictor. We can see that LP-RAG is superior to GFM-RAG in 5/6 scenarios. And, recalling Table 1, we can see that LP-RAG has comparable effectiveness to SOTA HippoRAGv2, being superior in the inference of the 2wiki dataset. These results show the superior ability of LP-RAG to generalize chunk searches for new queries in pretraining scenarios.

B.5 Full QA with Open-Source LLM

Our next experiment is a complete version of the experiment reported in Table 5. The baseline results are collected from the NodeRAG [61] paper. In this experiment, LP-RAG follows the same setup as EXPERIMENT III, but now trained with the full version of the datasets.

Table 15: Q&A results: LP-RAG outperforms all baselines in all datasets while utilizing minimum resources. Bold denotes the best results.

Methods	HotpotQA		MuSiQue		MultiHop		RAG-QA	
	Acc.↑	Tok↓	Acc.↑	Tok↓	Acc.↑	Tok↓	Acc.↑	Tok↓
NaiveRAG	87.50	9.8k	39.43	9.6k	56.00	8.9k	92.60	9.1k
HyDe	73.00	10.0k	33.14	9.8k	53.00	9.4k	94.90	9.3k
GraphRAG	89.00	6.6k	41.71	6.6k	53.00	7.4k	96.00	6.8k
LightRAG	79.00	7.1k	36.00	7.4k	50.00	7.9k	93.70	7.7k
NodeRAG	89.50	5.0k	46.29	5.9k	57.00	6.1k	97.70	3.4k
LP-RAG	**92.50**	**4.7k**	**68.13**	**4.1k**	**68.00**	**4.7k**	**98.90**	**2.8k**

In Table 15 we can see that LP-RAG is superior in all scenarios, with accuracy gains of 21.84, 11.0, 3.0, and 1.2 on MuSiQue, Multihop, HotpotQA, and Arena-QA, respectively.

B.6 Ablation: varying the number of synthetic queries generated

In this experiment, we demonstrate how LP-RAG behaves when we vary the number of synthetic queries generated for each chunk.

Table 16: Recall@k and index time by varying the number of synthetic queries created.

$n_{\mathcal{B}}$	HotpotQA			2Wiki		
	index (min)	R@2	R@5	index (min)	R@2	R@5
2 (Our)	**287**	**77.9**	89.6	**248**	89.9	93.2
4	329	77.9	89.6	316	89.9	93.2
6	458	78.1	**89.7**	445	**90.3**	**94.7**

As we can see in Table 16, as $n_{\mathcal{B}}$ increases, indexing time grows. However, the gain in recall@k does not offset the cost. We believe that as $n_{\mathcal{B}}$ increases, the diversity of generated queries drops. This directly addresses the diversity-vs-noise trade-off: $n_{\mathcal{B}} = 2$ captures diverse signal while avoiding the noise/redundancy regime.

B.7 Ablation: varying the LLM used to generate chunks and synthetic queries

In this experiment, we evaluate the impact of using different Small Language Models (SLMs) on chunk extraction and synthetic query generation, considering their influence on the effectiveness and efficiency of LP-RAG.

Table 17: Recall@k and index time by varying the LLM. NER* refers to fast variant using qwen-3.5-4b as synthetic query generator.

model	HotpotQA			2Wiki		
	index (min)	R@2	R@5	index (min)	R@2	R@5
qwen-3.5-0.8b	45	56.1	73.9	36	59.8	86.9
qwen-3.5-2b	51	75.9	82.1	45	74.3	89.4
qwen-3.5-4b	56	77.2	88.6	49	89.4	92.7
gpt-4o-mini	287	77.9	89.6	248	89.9	93.2
NER*	37	80.2	94.7	36	89.7	93.4

In Table 17 we can see that using a 4B-parameter SLM causes a minimal negative impact on the recall@k of LP-RAG, while noticeably improving indexing time. We believe this is mainly due to the fact that these models already have the ability to extract atomic chunks and generate synthetic queries. In the NER version, the impacts are even more evident, obtaining the best results in terms of time and recall.

B.8 Sensitivity analysis of k-NN graph

Table 18: Sensitivity of LP-RAG to the mutual-k-NN graph parameter.

k	HotpotQA		MuSiQue	
	R@2	R@5	R@2	R@5
2	59.1	75.2	37.2	48.6
5 (LP-RAG)	77.9	89.6	53.6	61.8
8	75.3	84.2	48.5	56.9

We evaluate the sensitivity of LP-RAG to the mutual-k-NN graph parameter k. Smaller values produce sparser graphs, limiting the paths available for information propagation, whereas larger values increase computational costs and generate larger chunk batches for synthetic-query generation. We hypothesize that overly large batches make it more difficult for the generator to produce focused queries that depend on all included chunks. As shown in Table 18, performance is substantially lower at $k = 2$, peaks at $k = 5$, and decreases moderately at $k = 8$, indicating that overly sparse graphs hinder retrieval, while increased connectivity beyond $k = 5$ provides no additional benefit despite higher graph construction and query-generation costs.

B.9 LP-RAG with top-5 chunks

Table 19: Evaluation of LP-RAG with context limited to its five highest-scoring chunks.

Method	HotpotQA		MuSiQue		2Wiki	
	EM	tokens	EM	tokens	EM	tokens
LP-RAG	51.6	2.7k	30.9	3.1k	69.8	2.6k

In Table 19 we evaluate LP-RAG with its context limited to the five highest-scoring chunks. As expected, recall remains unchanged on both benchmarks, since the top-5 chunks are already included in the original retrieval results. However, this restriction reduces token usage by 16% and 18%, on Hotpot and MuSiQue, with only modest EM drops of 1.6 and 1.7 percentage points, demonstrating a favorable trade-off between answer quality and context efficiency.

B.10 Candidate pruning analysis

We further investigate explicit candidate pruning using a dense similarity-based retriever to shortlist 100 candidate chunks. We consider two variants: (1) performing message passing over the full chunk graph while connecting the query only to the shortlisted candidates, and (2) restricting message

Table 20: Evaluation of explicit candidate pruning using a dense (similarity-based) retriever to select 100 candidate chunks.

Method	HotpotQA		MuSiQue	
	R@2	R@5	R@2	R@5
LP-RAG	77.9	89.6	53.6	61.8
Type 1	77.4	89.1	53.2	60.7
Type 2	71.4	82.5	45.8	54.7

passing to the pruned candidate subgraph, with the query connected to all its nodes. As shown in Table 20, Type 1 preserves nearly all retrieval performance, with R@2 decreases of only 0.5 and 0.4 percentage points on HotpotQA and MuSiQue, respectively. This reduces query-specific edges and candidate scoring from (|C|) to 100, while retaining message passing over the full document graph. In contrast, Type 2 achieves stronger graph-level pruning but incurs larger performance drops, suggesting that NCN benefits from structural information beyond the initial candidate shortlist.

B.11 Scalability and Inference Efficiency

Table 21: Per-query inference latency breakdown (in seconds) across the evaluated datasets, separating query embedding, graph insertion, GNN retrieval, and LLM generation.

Dataset	Query embedding	Graph insertion	GNN retrieva	LLM generation
MuSiQue	0.041	0.052	0.016	0.499
HotpotQA	0.045	0.034	0.012	0.592

LP-RAG's exhaustive inference scales linearly with the corpus size. Specifically, for a chunk graph with $|C|$ nodes, the k-NN construction introduces at most $O(k|C|)$ edges, while query insertion adds up to $|C|$ candidate query–chunk connections. For an L-layer GNN with hidden dimension d, the forward-pass complexity is approximately $O(L(|C|d^2 + |E|d))$, where $|E|$ denotes the number of edges in the chunk graph. Despite this scaling, GNN inference is not the dominant source of online latency in our evaluated settings. Table 21 reports the per-query latency breakdown, showing that GNN retrieval accounts for only 0.016s on MuSiQue and 0.012s on HotpotQA, compared with 0.499s and 0.592s for LLM generation, respectively. These results indicate that the graph-based retrieval component introduces relatively low inference overhead compared with the generation stage.

C Used Prompts

In this section, we provide all the prompts used during the LP-RAG experiments. Note that, overall, we adopted a few-shot example instruction prompting strategy. This allowed us to have more control over the LLM output, making the task of retrieving information easier via Regex. For example, by specifying in the prompt that we want the output of the chunk extraction step to be [chunk], we can force LLM to add the characters [and] to each chunk, which allows for very easy string regex searching. One drawback of using this technique is that it significantly increases LLM's input token consumption, which can result in higher additional costs for users of paid APIs, such as openAI. However, openAI itself is already addressing this issue by implementing the cached-prompt strategy.

Box C.1: Chunk extraction prompt

—Role: System—
You are required to act as an AI annotator and extract the Chunks embedded in the sentences of the provided paragraph text. Below, you will be given a paragraph from a text. You need to break it down sentence by sentence and extract the Chunks embedded in each sentence. The extracted Chunks can be an idea, argument, or fact. Each sentence may contain one or more Chunks to be extracted. The extracted Chunks should be as granular as possible to ensure they cannot be further broken down.
When extracting Chunks from a sentence, pay attention to the context within the paragraph. Replace pronouns with the nouns they represent and complete any omitted sentence components to ensure the independence of the Chunks is not compromised. This means that each extracted Chunk should not contain pronouns whose referents cannot be found within that Chunk.

Below is an example interaction that can serve as a reference for format and method of extracting Chunks:

System's Input:

[The Start of Paragraph text]
Pakistan Super League (Urdu: PSL) is a Twenty20 cricket league, founded in Lahore on 9 September 2015 with five teams and now comprises six teams. Instead of operating as an association of independently owned teams, the league is a single entity in which each franchise is owned and controlled by investors.
[The End of Paragraph text]

Your Answer:

[Sentence 1]
Pakistan Super League (Urdu: PSL) is a Twenty20 cricket league, founded in Lahore on 9 September 2015 with five teams and now comprises six teams.

[Extracted Chunks in Sentence 1]
[Pakistan Super League (PSL) is a Twenty20 cricket league.]
[Pakistan Super League (PSL) was founded in Lahore on 9 September 2015.]
[Pakistan Super League (PSL) was founded with five teams.]
[Pakistan Super League (PSL) now comprises six teams.]

[Sentence 2]
Instead of operating as an association of independently owned teams, the league is a single entity in which each franchise is owned and controlled by investors.

[Extracted Chunks in Sentence 2]
[Pakistan Super League (PSL) does not operate as an association of independently owned teams.]
[Pakistan Super League (PSL) operates as a single entity.]
[Each franchise in the Pakistan Super League (PSL) is owned by investors.]
[Each franchise in the Pakistan Super League (PSL) is controlled by investors.]

—Role: User—
[The Start of Paragraph text]
{Your Text Here}
[The End of Paragraph text]

Box C.2: Query response prompt

—Role: System—
You are a through assistant responding to questions based on retrieved information.
Below, you'll find a bag of chunks and a query. Your job is to answer the query using only the information contained in the chunks. If the answer isn't in the chunks, simply answer: "I don't know."

—Role: User—
[Bag of chunks]
{Retrieved chunks}

[Query]
{Your query}

Box C.3: Synthetic query generation prompt

—Role: System—
You are required to act as an AI annotator and generate questions in the provided paragraph chunks. Below, you will be given a bag of chunks from a paragraph. You need to generate two questions about it for each chunk. The generated two questions need to be like a human made question and each chunk may contain two questions. The question generated should be as similar as possible to a human question.
When generating the two questions from a chunk, pay attention to the context within the chunk.

Below is an example interaction that can serve as a reference for format and method of generate questions:

System's Input:

[The Start bag of chunks]
[Chunk 1]
We propose a model-based reinforcement learning (MBRL) approach for continuous action spaces.
[Chunk 2]
Bthardamz is only accessible during the quest The Only Cure.
[The End bag of chunks]

Your Answer:

[Questions Chunk 1]
[What type of reinforcement learning approach was proposed?]
[What type of action spaces does the proposed approach target?]
[Questions Chunk 2]
[When can Bthardamz be accessed?]
[What quest must be completed to access Bthardamz?]

—Role: User—
[The Start bag of chunks]
{Your chunks here}
[The End bag of chunks]

D LP-RAG Pipeline Examples

In this section, we show a simple example of the chunks returned by LP-RAG and the LLM response for a given query. The pipeline followed in this experiment was the same as that reported in Table 5. In Box C.1, we highlight in red the main chunks used by LLM to generate the answer.

Box D.1: Hit case from MuSiQue

Query:
When was the publisher of Crux launched?

LP-RAG retrieved chunks with probabilities:
"Crux Ansata is a book subtitled 'An Indictment of the Roman Catholic Church' by H. G. Wells.", 0.94
"Crux Ansata is a wartime book.", 0.87
"Crux Ansata has 96 pages.", 0.92
"Crux Ansata was first published in 1943.", 0.95
"Crux Ansata was published by Penguin Books.", 0.91
"Crux Ansata was published in Hammonsworth, Great Britain.", 0.88
"Crux Ansata is classified as Penguin Special No. 129.", 0.90
"The U.S. edition of Crux Ansata was copyrighted.", 0.89
"The U.S. edition of Crux Ansata was published in 1944.", 0.93
"The U.S. edition of Crux Ansata was published by Agora Publishing Company.", 0.86
"The U.S. edition of Crux Ansata was published in New York.", 0.92
"The U.S. edition of Crux Ansata includes a portrait frontispiece.", 0.90
"The U.S. edition of Crux Ansata includes an appendix.", 0.87
"The appendix of the U.S. edition of Crux Ansata contains an interview with H. G. Wells.", 0.94
"The interview with H. G. Wells in the U.S. edition of Crux Ansata was recorded by John Rowland.", 0.91
"The U.S. edition of Crux Ansata has 144 pages.", 0.88
"The U.S. edition of Crux Ansata went into a third printing.", 0.93
"The third printing of the U.S. edition of Crux Ansata occurred in August 1946.", 0.89
"CrossGen was an American comic book publisher.", 0.95
"CrossGen was an entertainment company.", 0.88
"CrossGen operated from 1998 to 2004.", 0.95
"The Walt Disney Company acquired CrossGen's assets in 2004.", 0.94
"CrossGen's assets were designated to Disney Publishing Worldwide.", 0.90
"Disney re-established the CrossGen brand in July 2010.", 0.87
"Disney re-established the CrossGen brand through Marvel Comics.", 0.91
"Marvel Comics announced plans to revive CrossGen titles.", 0.89
"Crux is an American comic book.", 0.86
"Crux was published by CrossGen Entertainment.", 0.98
"Crux was published from May 2001 to February 2004.", 0.88
"Crux was cancelled in 2004.", 0.92
"Crux was cancelled due to bankruptcy.", 0.87
"Crux was one of the later titles of Crossgen.", 0.91
"Crux came first in a second wave of Crossgen titles.", 0.89
"The second wave of Crossgen titles included Sojourn.", 0.85
"The second wave of Crossgen titles included Brath.", 0.88
"Crux detailed the exploits of six Atlanteans.", 0.93
"The six Atlanteans were awoken 100,000 years later.", 0.94

LLM response:
The publisher of Crux was launched in 1998 by CrossGen that operated from 1998 to 2004.

When does it work? During the chunk-node graph generation phase, LP-RAG interconnected most of the returned chunks in this example. In particular, the chunks ["CrossGen operated from 1998 to 2004.", "Crux was published by CrossGen Entertainment.", "CrossGen was an American comic book

publisher.", “Crux is an American comic book."] were interconnected. During the synthetic query generation phase for the chunks, our model had the following queries added to the LP-RAG training graph: [“Which American comic book publisher, active between 1998 and 2004, published Crux?", “Which company published American comic books between 1998 and 2004, including Crux?", “What is Crux and who published it?"]. This combination of synthetic queries and interconnected chunks during the training stage provided LP-RAG with a high power of interpretation of the query in question in the example, thus returning the correct answer.

When does it fail? In the Blood Street case, LPRAG retrieved the necessary evidence (Blood Street $\rightarrow$ Leo Fong and Leo Fong $\rightarrow$ Chinese American) but also many high-probability distractor director–nationality edges, producing a noisy subgraph in which GPT-4o-mini failed to preserve the correct entity chain and answered American. This case suggest that LP-RAG may learn strong lexical or local semantic associations but, sometimes, not always generalize to the relational and compositional reasoning required to build a complete and minimally distracting retrieval path.

Box D.2: Failure case from 2wikimultihop

Query:
What nationality is the director of film Blood Street?

LP-RAG retrieved chunks with probabilities:
“Interview with a Hitman is a 2012 British action film.", 0.82
“Interview with a Hitman was written by Perry Bhandal.", 0.54
“Interview with a Hitman was directed by Perry Bhandal.", 0.91
“Interview with a Hitman tells the story of Viktor.", 0.67
“Viktor is played by Luke Goss.", 0.76
“Viktor is a professional Romanian hitman.", 0.61
“Viktor agrees to tell Viktor’s story to a disgraced film director.", 0.88
“The disgraced film director is desperate to discover a unique story.", 0.53
“A unique story will help the disgraced film director rebuild the disgraced film director’s career.", 0.74
“Interview with a Hitman was produced by Kirlian Pictures.", 0.59
“Interview with a Hitman was produced by Scanner Rhodes.", 0.95
“Interview with a Hitman was produced with the assistance of Northern Film e Media.", 0.63
“Michael Govan was born in 1963.", 0.51
“Michael Govan has been the director of the Los Angeles County Museum of Art since 2006.", 0.85
“Prior to 2006, Michael Govan worked as the director of the Dia Art Foundation in New York City.", 0.79
“Blood Street is a 1988 film.", 0.65
“Blood Street was co-directed by Leo Fong.", 0.97
“Jackie Kong is an American screenwriter.", 0.83
“Jackie Kong is a film producer.", 0.66
“Jackie Kong is a film director.", 0.94
“Jackie Kong is known for the cult horror film Blood Diner.", 0.78
“Ian Barry is an Australian director of film.", 0.62
“Ian Barry is an Australian director of television.", 0.87
“Dana Blankstein-Cohen was born on March 3, 1981.", 0.57
“Dana Blankstein-Cohen is the director of the Israeli Academy of Film and Television.", 0.90
“Dana Blankstein-Cohen is a film director.", 0.73
“Dana Blankstein-Cohen is an Israeli culture entrepreneur.", 0.69
“Charles Leavitt was born in 1953.", 0.50
“Charles Leavitt is an American screenwriter.", 0.86
“Charles Leavitt is best known for writing the 2006 film Blood Diamond.", 0.96
“Leo Fong was born on November 23, 1928.", 0.64
“Leo Fong is a Chinese American actor.", 0.80
“Leo Fong is a martial artist.", 0.75
“Leo Fong is a boxer.", 0.53
“Leo Fong is a former Methodist minister.", 0.91
“Leo Fong has been making films, acting, and directing since the early 1970s.", 0.88
“Leo Fong’s latest film role is in Challenge of Five Gauntlets.", 0.67
“Ralph L. Thomas was born on September 8, 1939.", 0.52
“Ralph L. Thomas is a Brazilian-born Canadian film director.", 0.81
“Ralph L. Thomas is a Canadian screenwriter.", 0.63
“Ralph L. Thomas was born in São Luís, Maranhão to Canadian Baptist missionary parents.", 0.87
“Ralph L. Thomas grew up in São Luís, Maranhão and in Canada.", 0.69
“Ralph L. Thomas attended the University of Toronto for two years.", 0.58
“Ralph L. Thomas began to write for the entertainment pages of the Toronto Star in 1963.", 0.75

LLM response:
Chinese.
Gold response:
American.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions outlined in both the abstract and the introduction are developed in the main text (Section 3 and 4).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Section 5 (Conclusions).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: No theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide full implementation details in Appendices A and D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We plan to release the code via a public repository once the paper is accepted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: All implementation details are given in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: For computational and cost reasons, we follow related works and do not report error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are described in Appendix A.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss it after Section 5 (Impact Statement).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers related to all datasets/toolboxes we used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new significant assets

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were employed solely for grammar and style corrections throughout the manuscript.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.